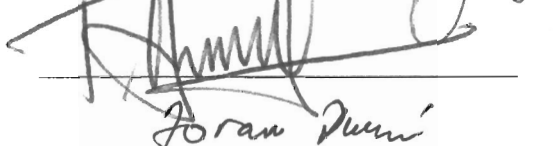


AN ANALYSIS OF ISLAND MODELS IN EVOLUTIONARY COMPUTATION

by

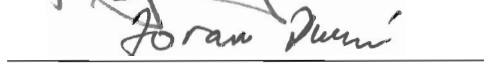
Zbigniew Maciej Skolicki  
A Dissertation  
Submitted to the  
Graduate Faculty  
of  
George Mason University  
in Partial Fulfillment of the  
the Requirements for the Degree  
of  
Doctor of Philosophy  
Computer Science

Committee:

  
\_\_\_\_\_  
  
\_\_\_\_\_

Dr. Kenneth A. De Jong, Dissertation  
Director

Dr. Tomasz Arciszewski, Committee Member

  
\_\_\_\_\_

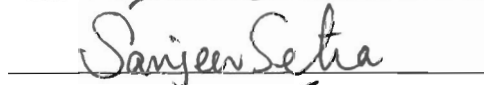
Dr. Zoran Durić, Committee Member

  
\_\_\_\_\_

Dr. John J. Grefenstette, Committee Member

  
\_\_\_\_\_

Dr. Sean Luke, Committee Member

  
\_\_\_\_\_

Dr. Sanjeev Setia, Department Chairperson

  
\_\_\_\_\_

Dr. Lloyd Griffiths, Dean, The Volgenau School  
of Information Technology and Engineering

Date: 12/6/07

Summer Semester 2007  
George Mason University  
Fairfax, VA

An Analysis of Island Models in Evolutionary Computation

A dissertation submitted in partial fulfillment of the requirements for the degree of  
Doctor of Philosophy at George Mason University

By

Zbigniew Maciej Skolicki  
Master of Science  
Jagiellonian University, 2000

Director: Dr. Kenneth A. De Jong, Professor  
Department of Computer Science

Fall Semester 2007  
George Mason University  
Fairfax, VA

Copyright © 2007 by Zbigniew Maciej Skolicki  
All Rights Reserved

# Dedication

To Pucia and My Parents

## Acknowledgments

Writing this dissertation took me much longer than I initially estimated, and required countless sacrifices. When I first started, writing lengthy *acknowledgments* seemed pretentious to me. However, as time passed, I became indebted to many people, and I am happy to be able to acknowledge them now. It is customary to thank your committee members first, then mention all people associated with the dissertation and finish with your beloved one. I am not following this order. While I understand that this dissertation is an academic achievement, the whole process of studying was not only a scientific experience, but rather a life experience, in which personal aspects played a major role.

Therefore, thanks go first to Iwona (aka Pucia), my wife. I admire how patient, believing and forgiving you were throughout this time. If we are together, it is thanks to you. Thank you for loving me.

I thank my parents — first for bringing me into this world. I find life a very interesting experience! Second, more importantly, for bringing me up and giving me all you have. It is intimidating to see how little you leave for yourselves. Together with my parents I need to thank my aunt Krystyna (aka Ciocia Krysia) for being my second mom.

Switching to the academic side, I would like to thank Dr. Kenneth De Jong, my advisor, for keeping me as a student and tolerating my “last minute attitude.” I appreciate your unconditional logic and precise criticism, which were intended to make me a better scientist. I hope I have improved at least a bit. Special thanks go to Dr. Tomasz Arciszewski. While we might have not always agreed, I think I became a more mature person thanks to you. I also appreciate times when you helped me in common-day life — thank you for caring about me, and last, but not least, for providing financial support. It is altogether fitting and proper that I should thank Dr. Sean Luke — for being passionate about every matter I had a chance to talk with you about, starting from technical subtleties, through science, to American culture and Asian food. Thanks for your detailed help with my final presentation and dissertation. I also want to thank John Grefenstette and Zoran Durić for your help with transforming a random collection of thoughts into this hopefully more consistent dissertation. In addition, I would also like to acknowledge Dr. Henry Hamburger for serving on my committee prior to your retirement.

It was always a pleasure to work with Dr. Mark Houck, and interesting to learn about water networks — thanks! I want to thank Dr. Mohan Venigalla for the traffic network data used for experiments in this dissertation. I would also like to thank

Associate Dean, Dr. Daniel Menascé for granting me multiple times The Volgenau School of IT&E fellowships.

My life in the past few years would be much more boring, if not for my roommates. Thanks to Raul and Marite Neira, and Anibal Barrantes for the first “Peruvian” years of my stay in the USA. Then, I owe a big “thank you” to Marcel (and Maria) Barbulescu for all the fun we had! So many episodes of our stay together come to my mind that I would need a separate chapter to describe them all! Ehm, maybe better not. I would also like to thank another roommate of mine, Joaquin Arteaga-Gomez, who kept introducing me (albeit not very successfully) to every possible TV series there is. More importantly, thanks for hosting me in your apartment towards the end of my studies — which thanks I also address to Isabel Rodriguez Tejedo, Yyannu Cruz Aguayo and Anand Radhakrishnan. Thanks also go to Sachiko Eguchi, and Nat Parry, the only truly American roommate I had!

I want to thank many friends for your company, which I enjoyed, and for many occasions on which you helped me. Let me mention Iwona and Rafał Kicingier (thanks for Easters!), Liviu Panait, Sepideh Mirza, Gabriel Cătălin Bălan, Elena Popovici, Joey Harrison (thanks for checking my English!), Keith Sullivan, Alexei Samsonovich, Emerson Farrugia, Denitsa Apostolova, Safaa Nhairy, Jarek Pietrzykowski, Janusz Wojtusiak, Urszula Kuczma, Tomohito Kanaizuka (who made me wade in a lake and enjoy it as part of orienteering), Jeff and Hideko Bassett (thanks for storing my motorcycle!), Adrian Grăjdeanu and Paul Wiegand. Special thanks go to Guido Cervone for all discussions we had and advice you gave me. Guido, Jacek Radzikowski, Veronica Grasso, Ruggero Scorcioni and Camelia Lică — I must say it was very enjoyable to share motorcycle rides with you — thanks a lot! Finally, I stayed in touch with a few close friends from Poland, initially remotely, but recently again in person.

Michelle Svoboda deserves a separate paragraph not only for being a friend, but also for meticulously proofreading the whole dissertation, and responding with more remarks than I could incorporate. While you may hate evolutionary computation by now, you did a superb job. Thank you!

Special thanks also go to Niklas Zennström and Janus Friis, the creators of *Skype*, because it enabled me to finish my dissertation remotely and to call my advisor to discuss my progress (if any). The difficult time of writing my dissertation was often relieved by reading the hilarious *phdcomics.com* by Jorge Cham. It gave me not only laughter, but also hope that maybe it is normal to struggle. Thanks! And, finally, let me also thank the *Google* team for changing my starting date so many times. Now I only hope that I will not have to cheer myself up by reading *Dilbert* too often!

# Table of Contents

|   | Page  |
|---|-------|
| List of Tables . . . . .                                | xi    |
| List of Figures . . . . .                               | xii   |
| List of Abbreviations . . . . .                         | xxi   |
| List of Symbols . . . . .                               | xxii  |
| Abstract . . . . .                                      | xxiii |
| 1 Introduction . . . . .                                | 1     |
| 1.1 Evolutionary algorithms . . . . .                   | 1     |
| 1.2 Island models . . . . .                             | 3     |
| 1.2.1 Good performance . . . . .                        | 4     |
| 1.2.2 Parallelization . . . . .                         | 6     |
| 1.2.3 Open problems . . . . .                           | 7     |
| 1.3 Thesis and contributions . . . . .                  | 8     |
| 1.4 Dissertation layout . . . . .                       | 10    |
| 2 Background . . . . .                                  | 12    |
| 2.1 Short history of evolutionary computation . . . . . | 12    |
| 2.2 Evolutionary algorithms . . . . .                   | 14    |
| 2.2.1 Representation . . . . .                          | 15    |
| 2.2.2 Genetic operators . . . . .                       | 16    |
| 2.2.3 Fitness . . . . .                                 | 18    |
| 2.2.4 Selection . . . . .                               | 18    |
| 2.2.5 Exploration vs. exploitation . . . . .            | 19    |
| 2.2.6 Research methodologies . . . . .                  | 20    |
| 2.2.7 Schema theory and building blocks . . . . .       | 21    |
| 2.2.8 Incremental and compositional evolution . . . . . | 23    |
| 2.3 Island models . . . . .                             | 24    |
| 2.3.1 Parameters of island models . . . . .             | 25    |
| 2.3.2 Diversity . . . . .                               | 27    |

|       |  |    |
|-------|--|----|
| 2.3.3 | Role of migration . . . . .                                | 28 |
| 2.3.4 | Advanced island models . . . . .                           | 30 |
| 2.4   | Other parallel models of evolutionary algorithms . . . . . | 34 |
| 2.4.1 | Master-slave model . . . . .                               | 34 |
| 2.4.2 | Neighborhood model . . . . .                               | 34 |
| 2.4.3 | Tags . . . . .   | 34 |
| 2.4.4 | Co-evolution . . . . .                                     | 35 |
| 2.5   | Related studies in evolutionary biology . . . . .          | 36 |
| 2.6   | Applications in engineering design . . . . .               | 37 |
| 3     | Methodology . . . . .                                      | 41 |
| 3.1   | The island model . . . . .                                 | 41 |
| 3.2   | Names, notation and conventions . . . . .                  | 44 |
| 3.3   | Experimental setup . . . . .                               | 45 |
| 3.3.1 | Statistical significance . . . . .                         | 45 |
| 3.3.2 | EA-1 and EA-2 . . . . .                                    | 46 |
| 3.3.3 | Migration mode . . . . .                                   | 47 |
| 3.4   | Tools for island model analysis . . . . .                  | 50 |
| 3.4.1 | Measures . . . . .   | 51 |
| 3.4.2 | Test functions . . . . .                                   | 52 |
| 4     | Two-level evolution . . . . .                              | 53 |
| 4.1   | Inter- and intra-island evolution . . . . .                | 54 |
| 4.2   | Oscillating convergence . . . . .                          | 55 |
| 4.2.1 | Gene independence in island models . . . . .               | 56 |
| 4.2.2 | Fixation forming building blocks . . . . .                 | 61 |
| 4.3   | Compositional evolution . . . . .                          | 64 |
| 4.3.1 | The IM6 function . . . . .                                 | 64 |
| 4.3.2 | The H-IFF function . . . . .                               | 65 |
| 4.4   | Parameter analysis . . . . .                               | 66 |
| 4.4.1 | Number of islands . . . . .                                | 68 |
| 4.4.2 | Island size . . . . .                                      | 72 |
| 4.4.3 | Large number of small islands . . . . .                    | 76 |
| 4.4.4 | Migration size . . . . .                                   | 82 |
| 4.4.5 | Migration interval . . . . .                               | 88 |
| 4.4.6 | Migration size and interval interaction . . . . .          | 91 |



|        |  |     |
|--------|--|-----|
| 4.4.7  | Policy . . . . .   | 93  |
| 4.4.8  | Topology . . . . .   | 104 |
| 4.4.9  | Joint migration parameters analysis . . . . .  | 107 |
| 4.4.10 | Increase of evolvability . . . . .   | 117 |
| 4.5    | Pitfalls . . . . .   | 123 |
| 4.5.1  | Deception at the inter-island level . . . . .  | 123 |
| 4.5.2  | Losing inter-island diversity . . . . .  | 126 |
| 4.6    | Summary . . . . .  | 128 |
| 5      | Migration as a genetic operator . . . . .  | 134 |
| 5.1    | Migration . . . . .  | 134 |
| 5.1.1  | Bidirectional random-random migration . . . . .  | 136 |
| 5.1.2  | Unidirectional random-random migration . . . . .   | 138 |
| 5.1.3  | Unidirectional selective migration . . . . .   | 145 |
| 5.2    | Selection . . . . .  | 146 |
| 5.2.1  | Uniform stochastic selection . . . . .   | 148 |
| 5.2.2  | Tournament selection . . . . .   | 148 |
| 5.3    | Selection with migration . . . . .   | 150 |
| 5.3.1  | Uniform stochastic selection . . . . .   | 150 |
| 5.3.2  | Binary tournament selection . . . . .  | 152 |
| 5.4    | Recombination . . . . .  | 154 |
| 5.4.1  | Two-children deterministic uniform recombination . . . . .                                     | 155 |
| 5.5    | Recombination with migration . . . . .   | 156 |
| 5.5.1  | Two-children deterministic uniform recombination and bidirectional random migration . . . . .  | 157 |
| 5.5.2  | Two-children deterministic uniform recombination and unidirectional random migration . . . . . | 160 |
| 5.6    | Recombination with random selection . . . . .  | 163 |
| 5.7    | Migration-recombination-selection scenario . . . . .   | 165 |
| 5.8    | Mutation . . . . .   | 170 |
| 5.9    | Summary . . . . .  | 177 |
| 6      | Dynamics of after-migration evolution . . . . .  | 179 |
| 6.1    | Allele survivability . . . . .   | 180 |
| 6.2    | After-migration dynamics models . . . . .  | 183 |
| 6.2.1  | Building blocks competing . . . . .  | 186 |

|       |   |     |
|-------|---|-----|
| 6.2.2 | Combining building blocks . . . . .                                     | 198 |
| 6.2.3 | Concatenating building blocks . . . . .                                 | 206 |
| 6.2.4 | General model . . . . .   | 210 |
| 6.3   | Improving migrations . . . . .  | 215 |
| 6.3.1 | Redesigning recombination operator . . . . .                            | 218 |
| 6.3.2 | Selection pressure . . . . .  | 220 |
| 6.3.3 | Repeated migrations . . . . .   | 221 |
| 6.4   | Inter-island interpretation of scenarios . . . . .                      | 235 |
| 6.5   | Summary . . . . .   | 238 |
| 7     | Heterogeneity in island models . . . . .                                | 242 |
| 7.1   | Representation heterogeneity . . . . .                                  | 243 |
| 7.1.1 | Heterogeneity by using exclusive or . . . . .                           | 244 |
| 7.1.2 | The F and F2 functions . . . . .  | 247 |
| 7.1.3 | Better representation dominating . . . . .                              | 250 |
| 7.1.4 | Representations cooperating . . . . .                                   | 253 |
| 7.2   | Discussion and conclusions . . . . .                                    | 259 |
| 8     | Exemplary problem analysis . . . . .                                    | 264 |
| 8.1   | Problems well suited for island models . . . . .                        | 265 |
| 8.2   | Simple functions . . . . .  | 266 |
| 8.2.1 | Asymmetric quadratic function . . . . .                                 | 266 |
| 8.2.2 | The IM6 function . . . . .  | 273 |
| 8.2.3 | The H-IFF function . . . . .  | 289 |
| 8.2.4 | The Deceptive function . . . . .  | 297 |
| 8.3   | Complex domains . . . . .   | 304 |
| 8.3.1 | Custom recombination operators . . . . .                                | 304 |
| 8.3.2 | Cellular automata and the majority problem . . . . .                    | 309 |
| 8.3.3 | Traffic congestion and optimizing street lanes . . . . .                | 319 |
| 8.4   | Summary . . . . .   | 327 |
| 9     | Conclusions . . . . .   | 331 |
| 9.1   | Contributions . . . . .   | 332 |
| 9.1.1 | Interaction between the local and global evolution . . . . .            | 333 |
| 9.1.2 | Tools for island models . . . . .                                       | 336 |
| 9.2   | Future work . . . . .   | 336 |
| A     | Island models as a remedy for evolutionary algorithm problems . . . . . | 339 |

|   |   |     |
|---|---|-----|
| B | Measures . . . . .                                  | 356 |
| C | Test suite . . . . .                                | 376 |
| D | Recombination revisited . . . . .                   | 393 |
| E | Convergence to migration equilibrium . . . . .      | 401 |
| F | Notes on experimenting with island models . . . . . | 404 |
|   | Bibliography . . . . .                              | 412 |

## List of Tables

| Table  | Page |
|--|------|
| 4.1 Parameters and their values. . . . .                                     | 109  |
| 6.1 Local dynamics creates inter-island evolution. . . . .                   | 238  |
| 7.1 Parameters used. . . . .   | 257  |
| 7.2 Changing EA parameters. . . . .  | 258  |
| 7.3 Changing migration parameters. . . . .                                   | 259  |
| 8.1 Selected best CAs found for the majority classification problem. . . . . | 314  |
| A.1 EA problems. . . . .   | 340  |

## List of Figures

| Figure   | Page |
|--|------|
| 1.1 Island model obtains much better results than both single-population and isolated multi-population models, on function IM6 (95% confidence level). . . . . | 5    |
| 4.1 Slower gradual fixation in IMs compared to fast simultaneous fixation in a single-population EA. . . . .   | 56   |
| 4.2 Diversity of each locus in an exemplary single run. . . . .  | 57   |
| 4.3 Average $x$ (first gene) value, the IM1' function. . . . .   | 59   |
| 4.4 Fitness, the IM1'' function. . . . .   | 61   |
| 4.5 Average trajectories of gene values, the IM1'' function, EA-2. . . . .   | 62   |
| 4.6 Island model obtains much better result than both single-population and isolated multi-population models, on the H-IFF function. . . . .                   | 66   |
| 4.7 Island models converge to the better peak due to restricted selection.   | 70   |
| 4.8 System-best fitness curves for different number of islands ( $N$ ), the IM6 function. . . . .  | 73   |
| 4.9 Best-in-run fitness values for different number of islands ( $N$ ), the IM6 function. . . . .  | 74   |
| 4.10 Diversity for different number of islands ( $N$ ), the IM6 function, EA-1.  | 75   |
| 4.11 System-best fitness curves for different island sizes ( $M$ ), the IM6 function.  | 77   |
| 4.12 Best-in-run fitness values for different island sizes ( $M$ ), the IM6 function.  | 78   |
| 4.13 Diversity for different island sizes ( $M$ ), the IM6 function, EA-1. . . . .   | 79   |
| 4.14 System-best fitness curves when keeping $N \cdot M$ constant, the IM6 function. . . . .   | 81   |
| 4.15 Best-in-run fitness values when keeping $N \cdot M$ constant, the IM6 function.   | 82   |
| 4.16 Best-in-run fitness values, when keeping $N \cdot M$ constant, various test functions. . . . .  | 83   |

|      |   |     |
|------|---|-----|
| 4.17 | System-best fitness curves in runs with different migration sizes ( $\alpha$ ), interval = 10, the IM6 function. . . . .  | 84  |
| 4.18 | Best-in-run fitness values in runs with different migration sizes ( $\alpha$ ), interval = 10, the IM6 function. . . . .  | 85  |
| 4.19 | Diversity in runs with different migration sizes ( $\alpha$ ), interval = 10, the IM6 function. . . . .   | 86  |
| 4.20 | System-best fitness curves in runs with different migration intervals $i$ , the IM2 function. . . . .   | 90  |
| 4.21 | Best-in-run fitness values in runs with different migration intervals $i$ . Larger migration interval helps islands to converge to different optima and recombine later to create the optimal solution, the IM2 function. | 91  |
| 4.22 | System-best fitness curves in runs with different migration intervals ( $i$ ), alpha = 0.1, the IM6 function. . . . .   | 92  |
| 4.23 | Best-in-run fitness values in runs with different migration intervals ( $i$ ), alpha = 0.1, the IM6 function. . . . .   | 93  |
| 4.24 | Diversity in runs with different migration intervals ( $i$ ), alpha = 0.1, the IM6 function. . . . .  | 94  |
| 4.25 | System-best fitness curves in runs with the same average number of migrants per generation on the IM6 function. . . . .   | 96  |
| 4.26 | Best-in-run fitness values in runs with the same average number of migrants per generation on the IM6 function. . . . .   | 97  |
| 4.27 | Diversity in runs with the same average number of migrants per generation on the IM6 function. . . . .  | 98  |
| 4.28 | System-best fitness curves in runs with different migration policy, the IM6 function. . . . .   | 100 |
| 4.29 | Best-in-run fitness values in runs with different migration policy, the IM6 function. . . . .   | 101 |
| 4.30 | Diversity in runs with different migration policy, EA-1, the IM6 function.  | 102 |
| 4.31 | With very weak (a uniform stochastic) selection, strong (best-random) policy together with a high level of migration results in additional selection pressure. $N = 20$ , $M = 20$ . . . . .                              | 105 |
| 4.32 | System-best fitness curves in runs with different migration topology, the IM6 function. . . . .   | 108 |

|      |  |     |
|------|--|-----|
| 4.33 | Best-in-run fitness values in runs with different migration topology, the IM6 function. . . . .  | 109 |
| 4.34 | Diversity in runs with different migration topology, the IM6 function.   | 110 |
| 4.35 | Maximal fitness for various parameters, EA-1 (with isolated islands $\approx 0.019$ , with panmictic $\approx 0.651$ ). . . . .  | 113 |
| 4.36 | Maximal fitness for various parameters, EA-1 with 5-tournament parent selection (with isolated islands $\approx 0.017$ , with panmictic $\approx 0.355$ ).114  |     |
| 4.37 | Maximal fitness for various parameters, EA-2 (with isolated islands $\approx 0.025$ , with panmictic $\approx 0.851$ ). . . . .  | 115 |
| 4.38 | Fitness compared. Small migrations boost evolution. . . . .  | 119 |
| 4.39 | Selection intensity compared. Small migrations boost evolution. . . .  | 121 |
| 4.40 | Diversity compared. Small migrations boost evolution. . . . .  | 122 |
| 4.41 | Allele survivability compared. Small migrations boost evolution. . . .   | 124 |
| 4.42 | Fitness for different number of islands ( $N$ ), the IM-Trap function. . .   | 127 |
| 4.43 | System-best fitness curves in runs with different island sizes, $M$ , the IM2' function. . . . .   | 129 |
| 4.44 | Best-in-run fitness values in runs with different island sizes, the IM2' function. Too big islands prevent islands from converging to different optima to recombine and create the optimal solution. . . . .                                       | 130 |
| 5.1  | Bidirectional migrations mix individuals from different islands. . . . .   | 137 |
| 5.2  | Local diversity decreases slower than global diversity. Unidirectional random-random migrations, for different $W$ . . . . .   | 140 |
| 5.3  | A sample simulated random walk of the number of copies of an individual, assuming it has reached $p_k = 0.1$ with $W = 1000$ . . . . .   | 142 |
| 5.4  | The expected $w^*(t)$ , and observed $w(t)$ , number of unique individuals compared. Unidirectional random-random migrations, $N = 20$ , $M = 50$ .143   |     |
| 5.5  | Unidirectional random-random migrations first mix individuals, then cause a drift. . . . .   | 144 |
| 5.6  | Strong (binary tournament-random) migration policy causes a fast loss of both global and local diversities (in absence of other operators). Unidirectional selective migrations shown, for different number/sizes of islands, $W = 1000$ . . . . . | 147 |

|      |  |     |
|------|--|-----|
| 5.7  | Selection causes fast local convergence, but preserves inter-island diversity. Experiments for different number/sizes of islands, $W = 1000$ .   | 149 |
| 5.8  | The percentage of unique individuals ( $\bar{w}(t)$ ). Uniform stochastic selection and unidirectional migration for different numbers/sizes of islands, $\alpha = 0.01$ and $i = 10$ .      | 151 |
| 5.9  | Uniform stochastic selection and unidirectional migration act opposite to each other. Interval of 30.  | 153 |
| 5.10 | The number of locally unique individuals ( $m(t)$ ), with different $N$ and constant $M = 10$ . Uniform stochastic selection and unidirectional migration, $\alpha = 0.1$ .                  | 154 |
| 5.11 | Global and local percentage of unique individuals ( $\bar{w}(t)$ and $\bar{m}(t)$ ), binary tournament selection and unidirectional binary tournament-random migration, with interval of 30. | 155 |
| 5.12 | The number of represented individuals per individual ( $R_{II}$ ), using two-children deterministic recombination, for various genome lengths. Single population.                            | 157 |
| 5.13 | Two-children recombination and bidirectional migrations mix alleles from all islands, for different genome lengths.  | 158 |
| 5.14 | Distances from island equilibrium. Two-children recombination and bidirectional migrations, for different genome lengths.  | 161 |
| 5.15 | Two-children recombination counteracts drift from a unidirectional migration (initial generations), interval of 30.  | 162 |
| 5.16 | Drift due to unidirectional migration occurs in later generations even though two-children recombination counteracts it.   | 164 |
| 5.17 | Uniform stochastic selection of parents, and one-child recombination results in a drift. The percentage of locally unique individuals ( $\bar{m}(t)$ ), using one-child recombination.       | 166 |
| 5.18 | The number of represented individuals per individual ( $R_{II}$ ) using one-child recombination, for different genome lengths.   | 167 |



|      |  |     |
|------|--|-----|
| 5.19 | Migration plus recombination slows down drift from selection, but cannot compensate it. One-child stochastic recombination and unidirectional migrations, different genome lengths, migration interval=1. . . . .  | 169 |
| 5.20 | Recombination increases the positive impact of migration on local diversity, but soon selection (or just stochastic sampling here) brings the diversity down, globally producing a slow drift. One-child stochastic recombination and unidirectional migrations, interval of 10. . . . . | 171 |
| 5.21 | Island-wide measurements converge to individual-wide measurements, but long after the mixing inside individuals took place. One-child stochastic recombination and unidirectional migrations, for different genome lengths. . . . .  | 173 |
| 5.22 | Island distance from migration equilibrium converges to average individual distance from the equilibrium. One-child stochastic recombination and unidirectional migrations, for different genome lengths. . . . .  | 175 |
| 6.1  | The number of represented populations in an individual ( <i>RPI</i> ), when migrants are accepted, neutral or rejected. . . . .  | 182 |
| 6.2  | Global diversity when migrants are accepted, neutral or rejected. . . . .  | 184 |
| 6.3  | Convergence to alpha individuals. . . . .  | 190 |
| 6.4  | Different $\alpha_0$ (percentage of best individuals) lead to the domination of different individuals. Results from the model for $L = \infty$ . . . . .   | 192 |
| 6.5  | The $\alpha$ - $\beta$ vector field, with paths of the three cases converging to different individuals. Results from the model for $L = \infty$ . . . . .  | 193 |
| 6.6  | Convergence map for the model with alpha= <b>10</b> , beta= <b>01</b> , gamma= <b>**</b> . . . . .   | 195 |
| 6.7  | Short genomes prevent gamma individuals from dominating. Results from the model for $L = 5$ . . . . .  | 197 |
| 6.8  | Different individuals dominate the population in different runs, for $\alpha_0 = 0.25$ and $L = 12$ . Results from a simulation. . . . .   | 199 |
| 6.9  | Convergence map for a simulation with alpha= <b>10</b> , beta= <b>01</b> , gamma= <b>**</b> . . . . .  | 200 |
| 6.10 | <i>Delta</i> individuals take over, with alpha= <b>10</b> , beta= <b>01</b> , gamma= <b>**</b> , delta= <b>11</b> , for $\alpha_0 = 0.2$ and $L = 5$ . . . . .   | 204 |
| 6.11 | Convergence maps with alpha= <b>10</b> , beta= <b>01</b> , gamma= <b>**</b> , delta= <b>11</b> . . . . .   | 205 |

|      |   |     |
|------|---|-----|
| 6.12 | Convergence maps with $\alpha=1^*$ , $\beta=1$ , $\gamma=**$ , $\delta=11$ .  | 209 |
| 6.13 | Different $\alpha_0$ and $L$ lead to the domination of different individuals.<br>Results from the general model, $p_s = (0.5)^{L/2}$ , $p_c = (0.5)^L$ (uniform recombination). | 213 |
| 6.14 | Convergence map with the general model, $p_s = (0.5)^{L/2}$ , $p_c = (0.5)^L$ (uniform recombination).  | 215 |
| 6.15 | Average fitness after migration for different convergence scenarios.<br>Results from the general model, $p_s = (0.5)^{L/2}$ , $p_c = (0.5)^L$ (uniform recombination).          | 216 |
| 6.16 | Convergence maps with a redesigned recombination (one-point crossover).   | 219 |
| 6.17 | Convergence maps with an increased selection pressure (a 4-tournament parent selection).  | 222 |
| 6.18 | Convergence maps with both increased selection pressure (4-tournament parent selection) and a redesigned recombination (one-point crossover)                                    | 223 |
| 6.19 | <i>Delta</i> individuals dominate for $\alpha = 0.2$ , $L = 10$ . Results from the model with $\alpha=1^*$ , $\beta=1$ , $\gamma=**$ , $\delta=11$ .                            | 225 |
| 6.20 | <i>Alpha</i> individuals dominate for $\alpha = 0.4$ , $L = 20$ . Results from the model with $\alpha=1^*$ , $\beta=1$ , $\gamma=**$ , $\delta=11$ .                            | 226 |
| 6.21 | <i>Beta</i> individuals dominate for $\alpha = 0.05$ , $L = 20$ . Results from the model with $\alpha=1^*$ , $\beta=1$ , $\gamma=**$ , $\delta=11$ .                            | 227 |
| 6.22 | Inconsistency between the model and a simulation for $\alpha = 0.25$ , $L = 30$ . Results from the model with $\alpha=1^*$ , $\beta=1$ , $\gamma=**$ , $\delta=11$ .            | 229 |
| 6.23 | The growth of alpha BB genes, while maintaining beta BB, $\alpha = 0.1$ , $L = 20$ .  | 230 |
| 6.24 | Delta individuals appear, $\alpha = 0.1$ , $L = 20$ , $i = 10$ .  | 232 |
| 6.25 | Delta individuals appear for $L = 30$ and $L = 40$ .  | 234 |
| 6.26 | Convergence maps with repeated migrations, $i = 20$ .   | 236 |

|      |  |     |
|------|--|-----|
| 7.1  | Performance when gradually changing representations (for both axes 0=binary, 99=Gray). . . . .   | 251 |
| 7.2  | A typical single run with single encodings on the function F. EA got trapped in local minima. . . . .  | 255 |
| 7.3  | A single run with both encodings on the function F, with different migration intervals (each line corresponds to one island). EA finds the global optimum. . . . . | 256 |
| 7.4  | Average results from 50 runs on standard multimodal functions. . . . .   | 260 |
| 8.1  | Asymmetrical quadratic function. Best fitness, real-valued representation. . . . .   | 268 |
| 8.2  | Asymmetrical quadratic function. Best fitness, binary representation. . . . .  | 269 |
| 8.3  | Asymmetrical quadratic function. Fitness grows even after global diversity drops significantly. EA-1. . . . .  | 270 |
| 8.4  | Asymmetrical quadratic function. Migrant alleles survivability differs between representations. EA-1. . . . .  | 274 |
| 8.5  | Asymmetrical quadratic function. Selection intensity confirms rejection of migrants for binary representation. EA-1. . . . .                                       | 275 |
| 8.6  | The IM6 function is more difficult for mutation than the asymmetric quadratic function, here shown for five dimensions. . . . .                                    | 277 |
| 8.7  | IMs behavior depends on representation and recombination, the IM6 function. . . . .  | 280 |
| 8.8  | Average gene value, the IM6 function. . . . .  | 281 |
| 8.9  | Extent to which individuals are the effect of mixing alleles between individuals and between islands; the IM6 function. . . . .                                    | 284 |
| 8.10 | Selection intensity, the IM6 function. . . . .   | 285 |
| 8.11 | Allele survivability, the IM6 function. . . . .  | 287 |
| 8.12 | A change in average fitness when alpha, beta or gamma/delta dominate (note a different scale on the y-axis), the IM6 function. . . . .                             | 290 |
| 8.13 | Two roles of migrations — additional selection pressure, or exchanging genetic material, the H-IFF function (note a different axis layout). . . . .                | 293 |
| 8.14 | Fitness in runs for different migration size and interval. Policy=2, topology=2, the H-IFF function. . . . .   | 295 |

|      |  |     |
|------|--|-----|
| 8.15 | Selection intensity in runs for different migration size and interval.<br>Policy=2, topology=2, the H-IFF function. . . . .  | 296 |
| 8.16 | Global diversity in runs for different migration size and intervals.<br>Policy=2, topology=2, the H-IFF function. . . . .  | 298 |
| 8.17 | Alleles survivability in runs for different migration size and interval.<br>Policy=2, topology=2, the H-IFF function. . . . .  | 299 |
| 8.18 | Two roles of migrations for Deceptive function — additional selection<br>pressure, or exchanging genetic material, the Deceptive function (note<br>a different axis layout). . . . .   | 300 |
| 8.19 | Fitness in runs for different migration size and interval. Policy=2,<br>topology=2, the Deceptive function. . . . .  | 302 |
| 8.20 | Selection intensity in a run for different migration size and interval.<br>Policy=2, topology=2, the Deceptive function. . . . .   | 303 |
| 8.21 | Global diversity in runs for different migration size and intervals.<br>Policy=2, topology=2, the Deceptive function. . . . .  | 305 |
| 8.22 | Alleles survivability in runs for different migration size and interval.<br>Policy=2, topology=2, the Deceptive function. . . . .  | 306 |
| 8.23 | Part of a graph of rules neighboring each other (let-hand sides for only<br>selected rules shown, close to $0^k$ and $1^k$ ). . . . .  | 311 |
| 8.24 | Fitness curves for CA domain results. . . . .  | 316 |
| 8.25 | Fitness changes in a configuration suggested by previous experiments<br>for the CA domain (50x20 IM, $\alpha=0.02$ , $i=20$ , 2-neighbor topology,<br>3-tournament selection, dynamic crossover, mutation=0.008, elitism). . . . . | 318 |
| 8.26 | Washington DC traffic network used in experiments. Circles represent<br>zones, and lines represent streets. . . . .  | 322 |
| 8.27 | Fitness curves for traffic domain results. . . . .   | 325 |
| 8.28 | Fitness changes in a configuration suggested by previous experiments<br>for the traffic domain (20x5 IM, $\alpha=0.1$ , interval=10, selection=2, static<br>crossover(20). . . . .   | 328 |
| C.1  | Function IM1. . . . .  | 378 |
| C.2  | Function IM1', $a = 1$ . . . . .   | 379 |
| C.3  | Gene contributions for the IM1'' function. . . . .   | 380 |

|      |   |     |
|------|---|-----|
| C.4  | Function IM2. . . . .                                     | 381 |
| C.5  | Function IM2'. . . . .                                    | 382 |
| C.6  | Quadratic function (two-dimensional case). . . . .        | 383 |
| C.7  | Function IM3 (two-dimensional case). . . . .              | 385 |
| C.8  | Function IM6 (two-dimensional case). . . . .              | 386 |
| C.9  | Possible configurations of a 4-bit trap function. . . . . | 387 |
| C.10 | Rosenbrock function. . . . .                              | 388 |
| C.11 | Schwefel function. . . . .                                | 389 |
| C.12 | Rastrigin function. . . . .                               | 390 |
| C.13 | Griewank function. . . . .                                | 391 |

## List of Abbreviations

|     |  |
|-----|--|
| BB  | building block                                     |
| EA  | evolutionary algorithm                             |
| EP  | evolutionary programming                           |
| ES  | evolution strategy                                 |
| GA  | genetic algorithm                                  |
| GP  | genetic programming                                |
| IM  | island model                                       |
| RPP | “represented populations in a population” measure  |
| RPI | “represented populations in an individual” measure |
| RIP | “represented individuals in a population” measure  |
| RII | “represented individuals in an individual” measure |

## List of Symbols

|                   |  |
|-------------------|--|
| $t$               | generation   |
| $N$               | number of islands  |
| $M$               | island size  |
| $W$               | $N \cdot M$ , total number of individuals                        |
| $m/\bar{m}$       | number/percentage of distinct individuals in an island           |
| $w/\bar{w}$       | number/percentage of distinct individuals in the system          |
| $d$               | distance from population migration-linkage equilibrium           |
| $d'$              | distance from individual migration-linkage equilibrium           |
| $s/\alpha$        | number/percentage of migrants (migration size), $\alpha = s/M$   |
| $p$               | probability for an individual to migrate                         |
| $i$               | migration interval   |
| $\alpha_0$        | initial percentage of migrants in an island (after migration)    |
| $\alpha/\alpha_t$ | percentage of migrants in an island (after generation $t$ )      |
| $\beta/\beta_t$   | percentage of locals in an island (after generation $t$ )        |
| $\gamma/\gamma_t$ | percentage of unfit hybrids in an island (after generation $t$ ) |
| $\delta/\delta_t$ | percentage of fit hybrids in an island (after generation $t$ )   |
| $\oplus$          | exclusive or (xor)   |
| $\circ$           | function composition, $(f \circ g)(x) = g(f(x))$                 |
| $\Omega$          | solution search space  |
| $G_L$             | graph of encodings of length $L$ , with neighborhood relation    |
| $\mathbb{R}$      | real numbers   |

# Abstract

AN ANALYSIS OF ISLAND MODELS IN EVOLUTIONARY COMPUTATION

Zbigniew Maciej Skolicki, PhD

George Mason University, 2007

Dissertation Director: Dr. Kenneth A. De Jong

Island models (IMs) are a class of distributed evolutionary algorithms (EAs) in which the population is split into multiple sub-populations called islands. Separate EAs run independently on each island, but they interact by means of migrating individuals. Therefore, IMs are different both from a single-population standard EA, as well as from a set of multiple isolated EAs.

IMs are interesting for several reasons. They have been reported to yield better results than standard EAs. IMs are also advantageous when computational tasks must be distributed across multiple machines because their structure is easy to parallelize. However, despite many studies, no comprehensive theory describing their behavior has been developed. Due to the lack of theory and a complex architecture with many control parameters, setting up IMs has been a trial-and-error process, guided mostly by “rules of thumb.”

In this dissertation, I adopt a two-level (intra- and inter-island) view of IMs and show how this approach makes understanding their dynamics easier. They behave very differently than standard EAs, and in order to take full advantage of this, I



propose a better utilization of the inter-island level of evolution. In particular, I argue for setups with many relatively small islands, and I also show that compositional evolution may scale to the inter-island level.

The two levels of evolution influence each other, and I analyze this interaction more deeply. Migrations profoundly change the local dynamics and stimulate evolution, which often ultimately results in better performance. I study the role of genetic operators in this behavior and also create mathematical models of after-migration dynamics. This analysis gives us a better understanding of mixing and the survival of genes locally, and these processes in turn determine the type and level of interaction between islands globally. Further, using island heterogeneity enhances the inter-island evolution. Following the study, I analyze IM behavior on a range of test problems, including two complex domains.

This dissertation improves our understanding of the dynamics of IMs and suggests a qualitative change in the way we think about them. This perspective offers new guidelines for configuring IM parameters and opens new directions for future work.

# Chapter 1: Introduction

Complex problems with non-linear interaction of parameters and a huge space of possible solutions require special solving methods. Such methods cannot exhaustively inspect all possible solutions, because of their number. Therefore they have to select a subset of solutions to analyze, based on some *heuristics* predicting the usefulness of a given solution for finding the optimum. The algorithms that use heuristics to guide the search are called *metaheuristic algorithms* (Gustafson, 2004).

## 1.1 Evolutionary algorithms

*Evolutionary algorithms* (EAs) are metaheuristic algorithms, which are based on the principles of natural evolution. In EAs a population of solutions is iterated through a number of generations, and if used as problem solvers, EAs gradually find better and better solutions. EAs have been developed and studied for over 40 years (Fogel *et al.*, 1966; Holland, 1962; Rechenberg, 1965) and they have been successfully applied to various domains.

An EA is not a perfect method. Since EAs discover new solutions without human interference, it is tempting to treat an EA as a magic instrument capable of quickly reaching anything that is achievable in the representation used. This is rarely true and it's important to realize it. From a common user point of view, there are two main problems with EAs.

- they are computationally expensive, and we are not sure when to stop them.
- they often do not find the optimal solution because they become stalled.

The two problems are very much complementary. One could let the algorithm behave more randomly and explore more, which would prevent it from getting stuck — but could take it even longer to find the optimum. Or, one could try to find an optimum faster (more greedy approach) and risk converging to a local optima. Even if in theory there is always a non-zero probability of escaping from a local optimum, in practice the algorithm may stall for unacceptable long periods of time. This issue is particularly “dangerous” for complex domains, where a path to the optimal solution leads through many suboptimal ones. We are trading off time for the quality of the result and we are never sure when the next improvement will come.

Even if there is enough computational resources, EAs must meet seemingly conflicting goals. For example, EAs are expected to find peaks in the fitness landscape, but omit deceptive ones, to efficiently construct building blocks and to compose them later, but maintain diversity in case something goes wrong. Additionally, quite often we have a situation when a domain requires solutions to evolve at different levels of generality. More general structures should be discovered earlier and the details should be optimized later (although evolution in most implementations tends to put more pressure on details than on general design). In open-ended representations EAs are supposed to gradually increase solution complexity.

There exist a range of EAs. Some specialized EAs try to push the limits in more than one aspect — returning solutions in a reasonable time and of an acceptable quality. The price we pay for this is that they are less general — and whereas they

excel at some tasks, there might be others at which they perform quite poorly. It is good to understand what problems an algorithm is specialized for, so that we use it appropriately.

A relatively simple framework of standard EAs may be insufficient to solve more complex, engineering problems. Such types of problems very often have modular solutions, possibly with several levels of modularity. To achieve all the goals listed above, usually a very uniform, homogeneous EA is used, with repeated simple cycles of reproduction and selection. Perhaps a more “granular” structure of the EA itself would help in solving such problems. Although EA improvements may be achieved by various tweaks of EA components, a more fundamental change is possible — namely a setup with multiple EAs.

## 1.2 Island models

*Island models* (IMs) are models of EAs with multiple populations (*islands*), which evolve simultaneously and interact with each other by means of migrations. Separation of islands’ evolutionary processes make IMs different from standard EAs. At the same time, migrations make them very different from isolated runs of EAs.

There are at least two main reasons to study IMs, good performance and ease of implementation on multiple machines, which are described in the following subsections. The first feature is especially interesting to us, since for many problems we would like to see IMs not as a necessity, but rather as a method of choice. Being a model of an algorithm and not implementation, IMs can very well be implemented on a single machine as well.

### 1.2.1 Good performance

An IM is conceptually a rather simple enhancement to a standard EA and requires little additional computation. Yet it turns out very powerful and may outperform standard EAs. The simplest technique to improve the problem solving ability of an EA, assuming there are enough resources, would probably be to restart the whole process whenever stagnation is detected, and take the best result of all the runs. Other approaches try to make EAs “smarter” and run it fewer times. IMs join both the approaches mentioned above, by starting multiple evolutionary processes enhanced by interaction between them.

In Fig. 1.1, we see an example of an IM obtaining much better results than both a single-population model, and a model of isolated populations, on the IM6 function.<sup>1</sup> In this dissertation I will try to understand the reasons underlying this intriguing behavior.

What are the reasons for a better performance of IMs? IMs behave qualitatively different from standard EAs. First, IMs restrict the flow of information between individuals. They limit mating and they compare an individual’s fitness only with those on the same island. Thus, they create natural irregularities and borders, which may function as seeds for differentiating the evolutionary process and making it non-uniform, as suggested earlier. The results of this may be two-fold. On one hand it may be undesired — for example it can prevent successful mixing, which would lead to constructing a novel solution. On the other hand, the results may be positive

---

<sup>1</sup>Migration: size = 10% ( $\alpha=0.1$ ), interval = 10, policy = random, topology = full, EA: binary tournament parent selection, uniform crossover rate 0.7, 10 real-value genes, gaussian mutation rate 0.1 and  $\sigma=0.01$ , elitism, non-overlapping generations. See appendix C for definition of the IM6 function.

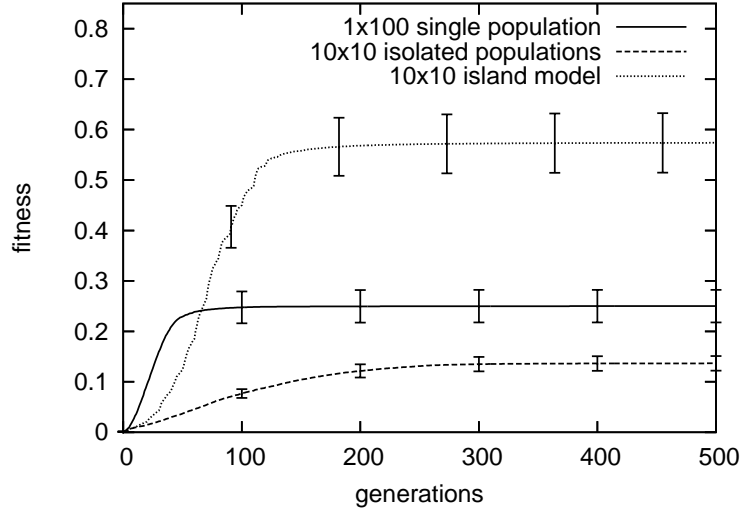


Figure 1.1: Island model obtains much better results than both single-population and isolated multi-population models, on function IM6 (95% confidence level).

— separation may stop the temporarily best solution from dominating the whole population and allow different building blocks to originate in separate regions. In the context of engineering design and creativity, the latter effect may be much more important, since it allows for the mixing of different subsolutions.

IMs intuitively should allow the improvement of solutions without direct competition, confronting them later with each other. Using multiple islands gives chances to seemingly weak individuals, which after a phase of development in their own niche (island) evolve into ultimately better solutions. This behavior should help to evolve solutions to deceptive problems, in which local improvements are globally harmful. Migrations exchange solutions that are optimized in different environments, and thus may help obtain a better general structure. In this context, IMs would be a good approach to enable evolution at different (at least two) levels of generality. This is in contrast to the behavior of a standard EA, which would usually decide on a

global structure early (due to convergence and fixation), and evolve the details later. IMs should be a good option for handling semi-decomposable problems.

### 1.2.2 Parallelization

IMs are easy to distribute. This is important, because EAs generally require long computations. If a single evaluation time is in terms of minutes or hours, running times of even several months are not unheard of. Similarly, large populations may require significant computational resources. It is a common practice to use multiple machines, to speed up EAs. Although it is possible to run a single-population EA on multiple machines, an IM is better suited for this task. A distributed IM does not need to communicate as often as an EA, which is important when the communication through a network is several orders of magnitude slower than within a single machine.

Existing hardware solutions support the usage of IMs. Each island can reside on a different machine, and an arbitrary topology of communication can be maintained using a network to connect the computers. Many companies sell *clusters* of machines, where hardware integration is higher than in the case of separate computers connected through a network. Clusters are a good platform for running IMs and spreading the load evenly. Furthermore, there is recently an interest in so-called *grid computing*, in which heterogeneous networks of loosely connected computers are used for computational purposes. The interest in grids is also a good reason for studying heterogeneous IMs. Finally, IMs are robust because in case of machines getting added or removed from the cluster (due to a failure, maintenance etc.), it is conceptually easy to simply add or delete islands.

### 1.2.3 Open problems

IMs come “in all shapes and colors,” and there is no good theory for them all. The whole domain is still young and even asking the right questions is sometimes difficult.

A good question is, what problems are IMs good for? What kind of problems are better solvable with IMs? Since we know that IMs cannot be good for everything, it would be desirable to identify a subclass of problems for which IMs generally return better results. A second aspect of the same issue is the following: Given that we decided to use an IM, for arbitrary reasons, how good will the results be when comparing an IM to a single-population EA? Can we expect comparable results? Another related issue is what are the dangers in using IMs? Can we significantly worsen the performance by using IMs?

Standard EAs suffer from multiple issues. An IM is a quite fundamental change to an EA, and therefore, it affects many of these issues. It is good to understand this influence and use it to better solve problems. See Appendix A for an analysis of possible EA issues and a discussion of how IMs may help with them.

Many researchers would like to use IMs, however, they feel discouraged because of how many parameters there are. Although we have some intuition about the behavior of an IM coming from a lot of experimental studies, nobody fully understands how to setup and manage IMs to increase problem solving ability. There is a need for a better understanding of properties of IMs.

To achieve the above goals, the underlying reasons for IM behavior must be understood. Although it is known that migration plays a central role in IM dynamics, it is not clear why and how this happens. We know the symptoms, but we do not know the reasons. Understanding the dynamics of IMs obviously simplifies setting



up experiments.

Identifying useful aspects of IMs makes it possible to enhance them. In this context a study of IM extensions could open a whole new area of research. Furthermore, it seems that the ability to explore various higher level structures of solutions depends on how differently evolution proceeds in islands. In addition, it is known that for example fluctuating the selection pressure, or sporadically increasing the mutation level are in general good ways out of stagnation. Differentiating islands (making the model *heterogeneous*) extends the desired features of island models. Instead of changing the behavior in run-time or counting on the diversification of evolution due to a stochastic nature of the process, one can set up an IM and *enforce* different islands. This requires different parameters for each island. The “roles” played by islands could then differ to a much greater extent than in a standard IM. Again, understanding the dynamics of these heterogeneous IMs would be important.

It seems that to a great extent most EAs depend on the initial “impetus,” which is given to them by carefully choosing parameters and possibly initial parents; this impetus is often lost before a good solution is found. Understanding the ways to maintain evolvability, while pinpointing good intermediate solutions as they appear is generally the *Holy Grail* of evolutionary computation. Are island models bringing us closer toward this goal?

### **1.3 Thesis and contributions**

A common understanding of IMs separates the migration process from evolution in islands, assuming that the latter does not differ much from evolution in single-population EAs. Following such an understanding, the separation of islands would

serve just as a way to maintain diversity, and successful migrations would result in “jumps” from one evolutionary path to another. A standard setup would consist of relatively few large islands, exchanging the best individuals, and using IMs as a necessary method to avoid high communication costs over network.

I propose a new way of thinking about IMs. As a result, they can become a method of choice, even if implemented on a single machine. In this dissertation, I make the following claims, based on a deeper analysis of the actual dynamics of IMs:

- IM dynamics can be understood as occurring at two levels. Evolution inside islands forms the local level, and the inter-island interaction creates a higher, global level of evolution. Both levels differ from an evolutionary process in a standard EA.
- The two levels have a deep impact on each other, through migration. Migrations dramatically change local dynamics inside islands. On the other hand, the level of mixing of migrants with other individuals determines the type of interaction between islands, creating the inter-island evolution.
- Migrations, rather than just spreading good solutions *after* evolving locally, may take an active role *during* inter-island evolution. For selected problems this may scale up the process of combining partial solutions to the inter-island level, resulting in *inter-island compositional evolution* causing islands to cooperate rather than compete.
- To take full advantage of both levels of evolution, I propose appropriate parameter settings and enhancements. A typical setup would consist of many

smaller islands and exchanging random individuals through rather small and infrequent migrations.

To achieve the goals of this dissertation I followed the methodology described below:

- I identified important IM aspects: oscillating convergence, gene independence, a potential support for compositional evolution.
- I performed a comprehensive IM parameter analysis.
- I analyzed migration influence on an island EA dynamics in the context of genetic operators.
- I created a genomic-level model of migration and analyzed the after-migration dynamics.
- I proposed heterogeneity as an extension enhancing IM properties.
- I analyzed IM behavior on a set of problems with growing difficulty.

Additionally, I created a “toolbox” of (either adapted or new) measures and compiled a set of functions useful for IM analysis.

## 1.4 Dissertation layout

The dissertation is organized along the methodology outlined above. After this introductory chapter, the next one (Chapter 2) presents the state of the art in the field. Then, in Chapter 3, I introduce various technical aspects, provide implementation specifics and briefly describe a set of problems and measures for analyzing IMs.

Chapter 4 introduces the two-level evolution and presents results of a standard parameter analysis of IMs. Although this is partially a review of what has already been known, the chapter also highlights the important aspects of IMs and sets the groundwork for further studies. What follows in Chapter 5 is an island-level analysis of migrations, explaining the dynamics of local evolution. Next, I take an even deeper approach studying after-migration dynamics at a gene level, in Chapter 6. The results show how local evolution in fact determines the inter-island evolution. A study of heterogeneous IMs follows in Chapter 7. I confirm the validity of the research by showing results of running IMs on various problems and explaining the results in the light of what we have learned in this dissertation, in Chapter 8. Finally, a summary and conclusions from the research are given in Chapter 9.

The appendices discuss various supplemental issues, not critical to the main body of the dissertation. Appendix A explore a wide list of EA problems, shortly analyzing them in the context of IMs. Appendix B provides a list of measures used for IM analysis in this dissertation. In Appendix C, I present a test suite composed from problems used in the dissertation. Appendix D provides a discussion of recombination operators and perfect allele mixing. Appendix E contains a proof for convergence of infinite islands to a migration equilibrium, as stated in Chapter 5. In the last Appendix F, I gather notes on designing, implementing and running experiments with IMs. Enjoy!

## Chapter 2: Background

In this chapter I present the background of the areas studied in the dissertation. I start by presenting history and basics of evolutionary computation (EC). Next, I devote a section to distributed models of EAs, in particular island models (IMs). I shortly review related studies in evolutionary biology (which field to much extent precedes EC, but takes a somewhat different approach). Finally I look at complex problems of engineering design and the reasons why they need more sophisticated algorithms.

The field of evolutionary computation has its roots in biology and the theory of natural evolution (Darwin, 1859). The development of computer technologies enabled researchers to simulate a simplified version of evolution on machines. There are several reasons to do so, among which the most important are the possibility of understanding complex evolutionary systems (which is important for evolutionary biologists or artificial life researchers) and the ability to use evolution as a search tool to find solutions to a given problem (as in the case of engineering design) (De Jong, 2006).

### 2.1 Short history of evolutionary computation

Although some ideas underlying research in evolutionary computation could be traced to the first half of the 20th century, the true beginning of the field should be placed

in the 1960s. One can differentiate several algorithmic approaches, resulting from historical divisions originating at this time. All three subgroups later influenced each other and since the 1990s one can see a uniform approach which is called *evolutionary computation*.

In 1965 Rechenberg and Schwefel started to solve real-valued parameter optimization problems with the use of evolution (Rechenberg, 1965). This approach was named *evolution strategies* (ES) and is effectively used for function optimization. Evolution strategy didn't use any crossover and so a relatively high level of mutation was used for searching. Notably, the experiments with self-adaptive evolution resulted in the famous 1/5 rule specifying the appropriate mutation level.

Fogel, Owens and Walsh began to evolve finite state machines, which gave birth to so-called *evolutionary programming* (EP) (Fogel *et al.*, 1966). Evolutionary programming used a more flexible representation and was applied to various problems. Although the recombination operator proved to be difficult in implementation, asexual reproduction (cloning) together with mutation was found successful in many domains.

Holland experimented with adaptive systems, which marks the beginning of *genetic algorithms* (GAs) (Holland, 1962). This approach introduced the crossover operator and used binary strings as representation. It was discovered that the choice of encoding for solutions can considerably affect the evolutionary process, and therefore, should be done with caution. Later research by Holland resulted in schema theory, which tries to explain how the evolution progresses from creating simple “building blocks” at the first stage to finding the solutions composed from them in the later stages.

EC is not a simple union of the above mentioned approaches, as there is much

research which is difficult to be classified as specific to any one of these approaches. A big subclass of EC research is done by the *genetic programming* (GP) community, which studies evolving objects of a complex structure and representation (traditionally in the form of trees) (Koza, 1992). The evolved objects in most cases can be executed in some way (often they are just computer programs), and fitness is assigned as a result of their behavior. Although genetic programming is close to evolutionary programming, the research crosses the borders of above defined categories.

## 2.2 Evolutionary algorithms

Evolution starts with a population of initial (often random) individuals representing some solutions to the problem that we want to solve and iterates through many generations using some rules that specify how to create the next generation. In the search for the best solution, evolution tries to gradually improve the quality of individuals. An important feature of evolution is its ability to produce novel and original solutions as the outcome of partially random, yet controlled changes.

The solutions from every generation play a major role in further evolution, and are the basis for individuals in the next generation. At first, *parents* are chosen in the *parental selection* process. They are later used to produce individuals for the next generation, by applying genetic *operators* to them. After the offspring are created, they are evaluated and a *survival selection* is applied to determine which children survive into the next generation.

### 2.2.1 Representation

To operate on solutions to a given problem, each solution must be encoded using some method of *representation*. Depending on the method, this encoding can be more or less direct. In the most direct representation the search will occur in phenotypic space, as in ES. More indirect encodings, for example, will include different binary representations in GAs, or programs and grammatical representation in GP. If a gene has a fixed position in a genome, its location is called a *locus*. Genes may have different gene values, called *alleles*.

Representation may be a *string* of values, or it may use some more complex structure (*tree, graph*) A string can be either *fixed-length* or *variable-length*, depending on whether all genotypes are of the same length or not. The simplest case to use to find a solution is obviously a fixed-length string.

A choice of representation may result in so-called *epistasis*. Epistasis means that genes don't influence the phenotype independently, but rather are linked (hence also the name *linkage*) and different configurations of them have different effects. Epistasis may result from the choice of representation, but may also be an internal characteristic of a given problem.

In evolutionary design, five requirements related to representation are considered to be important (Kicinger *et al.*, 2005; Gen and Cheng, 2000):

- 1-to-1 mapping between genotypes and phenotypes (I partially object to this in the light of theory of neutrality, see further).
- Legality of solutions (any possible permutation encodes a solution).
- Completeness (any solution can be represented).



- Lamarckian property (the meaning of alleles is not context dependent).
- Causality/Continuity (small variations in genotypes cause small variations in phenotypes).

### 2.2.2 Genetic operators

The two standard genetic operators are *mutation* and *recombination*. Genetic operators must be chosen appropriately based on the representation used.

Mutation makes small changes to the representation of a parent individual in the hope of adjusting the solution. There are multiple possible types of mutation. For binary representation, the most common mutation switches selected gene values between 0 and 1. For real number representations, mutation can either produce a new random value of a gene, or it can try to slightly disturb the existing value, possibly by adding a small number (most often taken from a Gaussian distribution). The latter is sometimes called *creep* as opposed to mutation (Davis, 1989). For other representations, mutations usually try to “tweak” the solution only slightly, in hopes that the phenotype, and thus, the fitness, also changes just a bit. Some rare exceptions are *hypermutation* (Cobb, 1990), mutation of a large subtree in a tree representation (Koza, 1992), or power-law based mutation (Krink *et al.*, 2000)

Recombination (also called *crossover*) uses two parents to create a child from parts of their representations, to combine features from both parents. The three most common used crossover types are *one-point crossover*, *two-point crossover* and *uniform crossover* (De Jong, 1975). One-point crossover cuts genomes of both parents in the *crossover point* and swaps the remaining parts to create two children (if only one is required, it is chosen randomly). Two-point crossover uses two points in which the

parent is switched when creating a child. Uniform crossover for each genome chooses independently from which parent to copy it. The choice may be done with some probability preferring one parent over the other parent (*parameterized crossover*). For more complex representation, much more complex recombination operators are used. Very often such operators are designed using knowledge about the problem domain. In GP, where most often a genotype is in the form of a tree, recombination exchanges subtrees.

Recombination may be more conservative, moving blocks of genes together, or may be less conservative, and mix, splice and join genomes more aggressively. A choice of a recombination operator may result in certain genes being copied together more often, which is referred to as genes that are *physically linked* (Watson, 2006). It is usually desirable that physical linkage follows epistasis. For more analysis of recombination and in particular its linkage preserving property, see Appendix D.

There are other possible operators, although they are not as common. *Inversion* and *swap* are mostly used in applications in which a given allele may appear a fixed number of times in a genotype and where the neighborhood relation between alleles in a genome should be preserved (in permutation problems). Inversion reverses some part of a genome. Swap exchanges the positions of two genes. An example of such a problem is a Traveling Salesman Problem, for which many new operators were designed (Freisleben and Merz, 1996; Mathias and Whitley, 1992; Mitchell *et al.*, 2000).

### 2.2.3 Fitness

The quality of individuals is measured by the so-called *fitness* of individuals. Fitness represents how good an individual is in the context of a particular problem and impacts which individuals survive and which are killed. The computation of the fitness can be very expensive and can dominate the costs of evolution. Traditionally, fitness is represented as a single real number, often with the requirement to be non-negative. Recently, there is an increasing interest in multi-objective evolution, where fitness is represented by a list of values. Multi-valued fitnesses can be compared using a *dominance* relation, which creates a partial order (Deb, 2001). Depending on the problem and the meaning of fitness, the fitness should be either minimized or maximized.

The evolutionary process is often viewed as searching in the space of all possible solutions. Fitness values of all solutions form a *fitness landscape*, which is a way of visualizing the search space. Fitness landscape shows how the fitness of individuals change when the genotype changes. Individuals having a similar genotype will have their fitness plotted close to each other. From the fitness landscape perspective, mutation produces an individual located in the neighborhood of the parent, and crossover produces an individual located somewhere between the parents (where exactly, it depends on the particular type of crossover operator).

### 2.2.4 Selection

Parental and survival selection are forces that limit the diversity of population and guide evolution toward regions of better fitness. Selection operators may have stronger or weaker selection pressure and they should properly counterbalance the exploration

inducted by genetic operators. When selection is weak, stochastic effects cause a gradual loss of alleles resulting in genetic *drift*. The most well-known selection strategies are (from the strongest to the weakest) (De Jong, 2006):

- *Truncation* — only the best individuals are chosen.
- *Tournament* — the winner of a  $k$ -size tournament is chosen.
- *Ranking* — individuals are chosen with probability depending on their ranking, which strategy is theoretically equivalent to a 2-size tournament.
- *Fitness proportional* — individuals are chosen with probability depending on the their fitness.
- *Uniform* — individuals are chosen at random.

### 2.2.5 Exploration vs. exploitation

In search algorithms two tendencies are confronted with each other. One is exploration, by which term one understands searching for new unseen possibilities, and the second is exploitation, meaning focusing on and improving so-far good solutions.

Standard evolutionary operators (selection and reproduction) are well understood in single-population models. They are traditionally shown as opposing each other, since selection causes exploitation of the search space, and reproduction operators create new individuals, exploring the search space.

It is believed that the success of an EA is very much dependent on the balance between exploration and exploitation. It is known that selection settings must match

other operator settings. A weaker selection matches well with a little mutation and non-destructive recombination (like one- or two-point crossover), which is responsible for most of the explorative work (typical GA setting). Stronger selection requires using much more mutation (ES setting), or a more aggressive recombination.

The balance between exploration and exploitation should also reflect a given problem domain. Strategies allowing for more exploration of the search space may move the population toward regions of worse fitness but also produce individuals in so far unexplored areas. Extreme settings result in a random search. Other strategies take a more exploitative approach, focusing on the faster convergence to the best (possibly suboptimal) solution in an already identified region. For some particular settings evolutionary algorithms can behave similar to the hill-climbing technique. With much simplification, evolutionary computation can be viewed to be between random search and hill-climbing, although it is a much more sophisticated technique.

### **2.2.6 Research methodologies**

Many researchers simplify the evolutionary model, so that they can use rigorous mathematical formalism to analyze it. One method of analysis of EAs is to “turn off” everything that is not needed in the algorithm and leave only the features on which the researcher wants to focus. An example of such a situation is the analysis of genetic drift in the presence of neutral selection (Rogers and Pruegel-Bennett, 1999). Another example would be an analysis of *takeover time* measuring how fast the best individual spreads in the population (Deb and Goldberg, 1991; Rudolph, 2000a; 2000b). Takeover time is measured without any mutation or crossover. An infinite

size of population (Vose, 1999), or a 1+1 model with a population consisting of a single individual is also sometimes assumed (Droste *et al.*, 2002).

The analysis of EA behavior can be of different detail level. *Stochastic* analysis is less accurate and measures some properties of a population in each generation (like the average fitness value or the average number of the best individuals) (Mühlenbein and Schlierkamp-Voosen, 1993; Blickle and Thiele, 1996). The method, although easier to perform, makes strong assumptions, in particular about probability distributions (Shapiro *et al.*, 1994; Popovici and De Jong, 2003). On the other hand, the next generation of solutions in EAs depends only on the previous generation, and therefore we can model evolution as a *Markov process* (De Jong *et al.*, 1995; Rudolph, 1998; 2000a; 2000b). Although much more complex, this method gives precise answers and needs less assumptions.

The *No Free Lunch* theorem (Wolpert and Macready, 1995; 1997) states that on average each search algorithm is equally good, if we consider all possible problems. This means that we can only improve an EA on some class of problems.

Researchers using EAs for solving a given problem may customize their algorithm for a particular task, often significantly increasing the complexity of the model. Using a complex model often allows only for an empirical analysis.

### **2.2.7 Schema theory and building blocks**

A very influential theory about how EAs work, which was developed for GAs, is the already mentioned schema theory (Holland, 1975; Goldberg, 1989; Poli and Langdon, 1998). Schemas are defined as sets of genomes with certain genes fixed, with fitness being an average fitness value over such set. Schema theory provides inequalities for

percentage of schemas in the population and claims that schemas with higher fitness will survive in population leading towards optimal solution as more and more genes are fixed. Whether it is a good explanation of how evolution works still remains controversial.

Schema theory was an attempt to formalize a more general building block hypothesis. The hypothesis explains how the final solution is achieved by first hierarchically finding intermediate steps, so-called *building blocks* (BB). BBs are blocks of alleles that put together make fitness better — or, assuming that a crossover operator is likely to copy adjacent genes together, BBs are short schemas of relatively high fitness. Smaller BBs are combined to create higher-level BBs, and the process continues through multiple levels. According to the BB hypothesis, mutation is mainly responsible for finding new alleles, whereas recombination is responsible for mixing different BBs, which together may create an even more fit BB, and ultimately find the optimal solution. The schema theory and building block hypothesis is based on proper crossover operators. They must preserve existing BBs and be able to create new ones.

Unfortunately BBs can be *deceptive* in at least two ways. First, single alleles that constitute a BB may cause a decrease in fitness if they are not within a BB. Second, although a BB may cause a relatively good fitness, finding it may lead an EA to a suboptimal solution because the optimal solution doesn't necessarily need to contain this particular BB. In fact these are two sides of the same problem. Deception means that good lower-level BBs may form a bad higher-level BB, and bad lower-level BBs may form good higher-level BBs.

Another problem occurs with one BB dominating others. If a BB with high fitness

is found, there is a risk that individuals containing this block will quickly dominate the whole population, disabling discovery of other required BBs. Such a situation was observed in the Royal Road functions (Mitchell *et al.*, 1992).

### 2.2.8 Incremental and compositional evolution

Two types of evolution have been identified: incremental and compositional (Watson, 2006). Incremental evolution refers to evolution that occurs through small changes, created e.g. by mutation. Compositional evolution refers to evolution which reaches final good solutions through the mixing of individuals by recombination. A common understanding of evolution tends to favor the first type, whereas the second type may be much more powerful and scalable.

Compositional evolution is an example of evolution progressing mainly due to recombination. As such, compositional evolution must to much extent follow the building block hypothesis. To maintain such evolution, modifications to the plain genetic algorithm may be necessary. One example of such modification can be the SEAM algorithm (Watson, 2006). Other examples include various linkage learning techniques (Goldberg *et al.*, 1989; Harik and Goldberg, 1997). In this dissertation I show why also island models may help compositional evolution.

Of course, not only a proper recombination operator must be used, but problems also must be of proper nature for compositional evolution to occur. What functions are suitable for compositional evolution? These are functions for which the optimum can be found by composing subsolutions. Watson gives a distinction between separability and modularity in his book on compositional evolution (Watson, 2006). A separable function would be one for which a solution consists from independent



modules. In fact, such modules could be solved one by one, but using an IM would support combining them. A modular function is a function in which modules are also identified, but they are not independent. Nevertheless, their interaction can be analyzed at the level of whole modules, i.e. additionally to *intra-dependence* between smaller blocks within each module, there exists an *inter-dependence* between modules for which the details inside modules are irrelevant and the module itself can be considered a unit. The inter-dependence relation may be as strong as the intra-dependence, and for some domains their strengths can be regulated independently. If such modular structure occurs at several levels, in which case bigger modules can be repetitively built from smaller ones, we have a hierarchical function. For all those functions (separable, modular and hierarchical), a composition of modules from different solutions is likely to improve next solutions. Even for non-hierarchical functions, it still may take multiple steps to compose the final solution, although composition takes place in an undefined order.

## 2.3 Island models

One of many possible extensions to the standard EA is to distribute individuals and restrict their interaction. Models restricting interaction between individuals are sometimes referred to as *spatial* models because individuals are not equally distant from each other anymore. Also, all models with restricted mating are sometimes referred to as *non-panmictic* models, as opposed to *panmictic*, single-population models (Sprave, 1999).

Spatially restricted models not only are easier to distribute, due to mostly local interactions, but they also show different behavior than standard EA models. For

example, spatially distributed EA models naturally slow down the flow of information inside the model and as a result, prevent any individual from quickly dominating the whole population. Such delay supports the diversity of solutions and may lead to better results than those achieved by using a single uniform population (for example, a high fitness BB in a Royal Road problem would not quickly dominate the whole population). Of course, distributing individuals requires answering many implementation questions.

As far back as 1932 an evolutionary theoretician, Sewell Wright, not only introduced the concept of a fitness landscape and nicely described the balance between selection and mutation, but also predicted interesting properties of what we would now call an island model (Wright, 1932). Wright claimed that a species with a subdivided population (into islands) has a better ability to move such smaller groups around the landscape, not even necessarily by adaptation, but rather by a random drift. Once they find a peak, they grow in size and by “crossbreeding” with other islands, push their evolutionary processes toward the peak. Wright characterizes “intergroup selection” as “enormously more effective” than “intragroup selection.” The description seems right, since evolution inside islands may quickly converge and is more local in nature, whereas the diversity between islands stays at much higher level and allows for exchange of optimized parts of solutions. This viewpoint matches the two-level (intra- and inter-island) understanding of IM dynamics.

### **2.3.1 Parameters of island models**

The simplest IM assumes no difference in setup of different islands and migrations, which means that there are some global parameters for islands and there are some

global parameters for migrations. In generational EAs, migrations usually take place between generations and often every fixed number of generations. Traditionally researchers distinguish the following parameters of IMs:

- For islands:
  - Number of islands.
  - Size of islands.
  - EA used.
  
- For migrations:
  - Size of migration.
  - Interval between migrations.
  - Policy (which includes):
    - \* Emigration, which defines which solutions to migrate — e.g. best, worst or random, and also whether to remove them from the source island, or just copy them.
    - \* Immigration, which defines how to include the migrants into the target island — e.g. add, replace worst, replace random, etc.
  - Topology — e.g. ring, torus, random, scale-free model (nodes' degrees obey power law).

Researchers don't fully understand how these parameters together influence the behavior of the algorithm. The existing research focuses on studying single properties as opposed to a study of deeper reasons for IM behavior. Alba and Troya studied

the asynchronism of IMs (Alba and Troya, 1999). Distributed computations and limited network bandwidths require only partial interaction between individuals. Full interaction would require sending all individuals over the network after each generation, considerably slowing down the algorithm. Cantú-Paz investigated communication time influence (Cantú-Paz, 1999). Diversity of individuals was studied by Ursem (Ursem, 2002) and Gustafson (Gustafson, 2004). Theoretical studies of locality and takeover times were done by Sprave (Sprave, 1999) and Rudolph (Rudolph, 2000a). Number and size of islands were studied by Whitley et al. (Whitley *et al.*, 1999) and Cantú-Paz (Cantú-Paz, 1999; Cantú-Paz and Goldberg, 2003). Analysis of migrations can be found in works by Sprave (Sprave, 1999) and Cantú-Paz (Cantú-Paz, 2001). The latter author describes the influence of migration policy on the selection pressure in IMs.

### **2.3.2 Diversity**

Losing genetic diversity is one of the main problems of longer evolutionary runs. Separate islands let the diversity remain high, and therefore, the interest in separate islands was always present in the EC community (even before it was named this way). In 1975, De Jong predicted the usefulness of introducing separate “species,” explaining that “This [...] has direct bearing on the problem of allele loss allowing, for example, exponential exploitation of a local minimum without a problem of complete population dominance” (De Jong, 1975).

De Jong also introduced the idea of *crowding*. This technique was intended for a single population, and its main idea is to remove from the population individuals most similar genetically to the new offspring, thus making place for the offspring. This was

later extended to deterministic crowding (Mahfoud, 1992), where each child created by recombination replaces the genetically closer parent (if it is better than this parent). Watson noticed that we may interpret each individual in deterministic crowding as a “highly converged subpopulation” in an IM (Watson, 2006). This is because low migration subpopulations will converge to “genetically related” individuals. In this interpretation each recombination corresponds to individuals migrating to another island, mixing with its contents and converging to another, better individual. What is different, however, is that in deterministic crowding, this entire process is done by a single recombination (which therefore must create better individuals instantly), whereas in IMs a local evolution takes care of the mixing and can extend through multiple generations (but at the cost of many more computations).

Many techniques maintaining the diversity of population, like crowding (De Jong, 1975), fitness sharing (Goldberg and Richardson, 1987) or Eshelman’s CHC algorithm for incest prevention (Eshelman, 1991), don’t allow individuals to survive or mate too closely, thereby “pushing them away” from each other. This prevents the algorithm from focusing on a single region. In IM, different islands converge to different regions initially due to stochasticity, and then due to selection picking different regions. Therefore one could say that each island “pulls” individuals toward different goals *actively* maintaining diversity.

### **2.3.3 Role of migration**

Intuitively, frequent and big migrations make islands to become very similar, since they start to share the same gene pool. Migrations that are rare and small may not exchange enough genetic material and the behavior of such a model may in turn

be similar to multiple independent runs. Therefore it seems that the right balance in terms of migrations' parameters is a key to achieve better results. The situation resembles seeking the right balance between exploration and exploitation, and it may very well be the case that the best results are achieved in a narrow range of parameters properly adjusted to each other.

Because the process of choosing which solutions to migrate occurs in the same way as choosing the parents for reproduction and individuals to survive, standard selection strategies (like truncation, tournament selection, rank selection) may be used for this purpose. Similarly, selection (or negative selection, i.e. selecting the worst individuals) strategies may be used for identifying the individuals to be replaced in the target island.

Migrating good individuals results in the increase of the selection pressure (Cantú-Paz, 2001). Therefore, if all the other parameters remain the same, we should expect increased exploitation and faster convergence from migrating good individuals (with a possible loss of the optimal solution). However, even with a stronger selection, the results obtained may still be better than in a standard EA (Whitley *et al.*, 1999). This is because the separation of islands serves as a natural way to support the diversity and may guide the evolution in many directions simultaneously.

In an asynchronous IM, islands do not synchronize the generations between them, which enables them to maximally use computational resources of each CPU (Alba and Troya, 1999). Each island EA operates independently and the *migrants* (migrated solutions) wait in special buffers until target islands are ready to accept them.

Changing the level of migration in run-time may be beneficial. Evolution inside islands is probably more important at the beginning of the run to build the basic BBs.

Migrations should be more important at the later stage, to increase the interaction between islands and mix the solutions. With the two-level view of dynamics this would correspond to putting emphasis on local evolution first, and switching it later to the global level evolution. A good behavior of migration policy starting with sparse migrations and increasing their level during the run was observed experimentally (Branke *et al.*, 2004). However, a decrease in the migration in favor of mating was also found to be beneficial in another study (Krink *et al.*, 1999). It is also worth noting that a lower level of migration in the first phase would favor incremental evolution inside islands and could disfavor compositional evolution between islands (although we will see that the second part is not necessarily true, because migration impact can be increased by evolution).

### 2.3.4 Advanced island models

In addition to standard IMs, with a number of identical islands connected by a regular topology, there exist models where islands differ from each other and may serve different roles in the whole system. Multiple islands make it possible to experiment with different representations, operators, evolutionary algorithms etc. within one system. Such a setup results in different evolutionary dynamics in each island. Understanding the impact of evolutionary framework design decisions in heterogeneous setups, even at the experimental level, is very much needed, as there is still no consistent theory covering these issues.

Heterogeneous, possibly tunable, evolutionary setups were shown advantageous when it comes to real applications. They may overcome artificial limits imposed by a choice of the particular type of evolutionary algorithm and representation —

which often prevents the search from reaching the true problem constraints. The multi-perspective evolution should avoid flaws of one set of chosen parameters, one representation or one fitness calculation. Diversification of islands has been observed often to have a positive impact on IM solving abilities. Different representations, even if seemingly redundant, may increase the evolvability, and ultimately lead to a more effective search (as suggested by the neutrality theory (Toussaint, 2001; Toussaint and Igel, 2002)). Varying the parameters of particular islands allows them to be assigned to different goals within one global search algorithm (Tsutsui and Fujimoto, 1993; Tsutsui *et al.*, 1997; Oppacher and Wineberg, 1999).

For conceptual design and high-level engineering problems migrating between islands representing different stages of design, could allow an automatic change of focus during evolution, resulting in a natural shift between different detail levels. Repeated migrations of an individual and its offspring from one island to another could expose them to a richer genotypic pool and possibly test them more thoroughly in varied environments, hopefully avoiding the dead ends encountered in the process of evolving solutions.

An interesting application of a heterogeneous setup is an Injection Island GA (Eby *et al.*, 1999). In the model there are several islands with different precisions of representation and different fitnesses. Some islands have simpler representation and faster fitness evaluation. Their task is to evolve the first, coarse approximation of the final solution. The best solutions migrate to the islands with more detailed representations and more expensive evaluations. There might be several levels of islands and the final solution is taken from the most complex one. All the solutions move one way, from simpler islands to more complex ones. The model is very efficient,



however, there is a price to pay for the speed-up in the discovery of good solutions, which is that some building blocks essential for the final solution may not be visible, and thus omitted, at the coarser representation level.

Another example of separating organisms into several pools with different goals is the Hierarchical Fair Competition (Hu *et al.*, 2005). Although not categorized specifically as an IM, this model separates organisms into different layers depending on the organisms' fitness.

When migrations between islands with different representations are allowed, proper translation procedures must be attached, so that after migration, a genotype still represents the same or similar phenotype. If the new representation only adds additional details, as in the case of Injection Island GA, then the translation is straight forward, just by representing the coarser individual in the new more detailed representation (Eby *et al.*, 1999). If the representations are somehow equivalent, as in the case of binary and Gray coding, then there exist procedures for translating from one to the other (Skolicki and De Jong, 2004). However, it is not obvious how to translate if the representations were deeply different.

Gen and Chen (Gen and Cheng, 2000; Kicinger *et al.*, 2005) argue for *non-redundancy* in representation, which means that there is a 1-to-1 mapping between encodings and solutions. This property would mean that there is also a 1-to-1 correspondence between different representations (at least on the set of solutions being represented in both representations). However, it seems that there is no absolute necessity for the translation mechanisms to maintain the 1-to-1 property. Of course, translating many solutions into one can cause the loss of some genetic information (especially in the case of many translations back and forth). However, one can also

imagine one-to-many stochastic translations, which may introduce some new genes. Moreover, having different genotypical representation for the same phenotype is known as beneficial in studies on *neutrality*. Switching the representation changes the exploration distribution around a given solution, and opens up a possibility for genetic operators to discover a new region of the search space. It also regulates how much a given solution (or part of solution) is susceptible for mutations (Toussaint, 2001; Toussaint and Igel, 2002).

Multiple islands allow for many criteria guiding the evolution and in fact, IMs can be used for multi-objective problems. Xiao and Armstrong studied island models in which each island could take only a subset of objectives into consideration (Xiao and Armstrong, 2003). Our preliminary studies also showed that different linear combinations of objectives in islands may perform relatively well, although worse than the SPEA2 algorithm.<sup>1</sup>

Instead of fixing the number of islands before a run, one can allow for a process of speciation, when populations are created when needed. Very often the decision about creating a new population is based on some heuristics. Speciation was studied by Ursem (Ursem, 1999; 2000), Bessaou et al. (Bessaou *et al.*, 2000) and Ronnewinkel and Martinez (Ronnewinkel and Martinez, 2001).

Designing multi-population EAs is not an easy task, simply because we have to deal with many more possible adjustments than in the case of a standard EA. In addition, heterogeneity of IMs requires further choices and adjustments.

---

<sup>1</sup>Together with Dr. Sean Luke, we compared multi-objective IMs with the SPEA2 algorithm (one of the best ones known). A few islands had different, single objectives and others, located in between the first ones, used weighted sums, gradually changing from one objective to another. Our algorithm found the true Pareto front on a set of difficult functions, without the need for a complex fitness assignment as was used in SPEA2, but SPEA2 was better with regard to the distribution of solutions on the front.

## 2.4 Other parallel models of evolutionary algorithms

Many related models to IMs were suggested, in which either implementation, or algorithm, or both, were distributed.

### 2.4.1 Master-slave model

The master-slave model distributes only the evaluation process (which is usually the most time-consuming part of EA) and performs all genetical operations on one big population. Although often viewed as a distributed EA model, conceptually it does not differ from a single-population EA. The only difference is the distributed implementation and not the algorithm itself.

### 2.4.2 Neighborhood model

One of the approaches to distribute evolution is a *neighborhood model* (also called a *fine-grained*<sup>2</sup> or *diffusion* model). Individuals form a topological structure, usually a grid, and are allowed to mate only within some neighborhood. The shape and size of the neighborhood may be different, but the radius of such neighborhood is the most important factor influencing the propagation of good individuals, affecting takeover times (Sarma, 1998).

### 2.4.3 Tags

One of the simplest ways to implement distributed models on a single CPU is to use *tags* (Spears, 1994; Deb and Spears, 1997). Tags are short sequences of genes attached to individuals and decide which individuals can mate with each other. Therefore,

---

<sup>2</sup>As opposed to IMs, which are sometimes called a *coarse-grained* model.

individuals with the same tags form a *species*. Tags can get mutated and in this way individuals can “migrate” from one species to another. Additionally, the size of a species can change in run-time, depending on how many individuals share the same tag. Tagging was primarily developed to enable EAs find many peaks in multi-modal landscapes.

However, there are several disadvantages of tagging. The method is not easily distributable on many CPUs. Depending on the actual tags and mutation operator, some “migrations” will occur more often than others, unless special steps are taken to treat tags specially — and introducing an arbitrary topology may be a problem as well. Because all individuals are selected against each other, one species could quickly start dominating the others and ultimately become the only one remaining. Therefore fitness sharing is used, so that too many individuals do not converge to the same region. Whereas in IMs existence of species is guaranteed, when using tags this existence is due to a delicate balance.

#### **2.4.4 Co-evolution**

In *cooperative co-evolution* solutions evolved in separate subpopulations may represent parts of the problem and the final solution is then constructed by putting together representatives from every subpopulation (Potter, 1997). Such a setup creates very complex dynamics (Popovici, 2006).

It would be possible to have individuals representing whole solutions, as in IMs, but with different parts of solutions being evolved in particular islands. In this case, the parts being evolved could overlap (same part of genotype could be evolved in several islands), leading to a non-disjunctive cooperative co-evolution.

## 2.5 Related studies in evolutionary biology

Biologists have studied evolution both in theory and practice long before computer scientists became interested in the subject. A field of biology which is closely related to the research in evolutionary computation is *population genetics*. Population genetics studies the changes in allele frequencies in a population. Similarly to the EC community, biologists study several operators that are crucial for evolution: mutation, recombination, drift (not an operator actually, but an effect of stochasticity), selection, and migration (called *gene flow*). Diversity of genetic material in a population is one of the main concerns of population genetics (Klug and Cummings, 1997). Models proposed by biologists were analyzed mostly in the context of probability theory.

Although biological models differ from standard EC models, there are many analogies. Biologists use diploid representations, as opposed to a standard haploid representation used in EC and put relatively less focus on linkage between genes. Fitness function is often dynamic and location dependent. Fitness also has a slightly different meaning. Rather than being a feature that decides about chance of an individual to survive and produce offspring, it is a measure of how successful an individual was at these tasks. Therefore, selection cannot be defined as “increasing fitness.” because it is rather fitness that is defined as “reproducing and being selected.” Of course some individuals are more adapted to the environment, survive longer and produce more offspring, and some other die. Thus, the level of adaptation corresponds to what we usually call fitness in EC.

Some studies in biology correspond to distributed EAs. A *stepping stone model*

corresponds to a neighborhood model in EC. Recently, Cox and Durrett achieved new results extending the model to multiple individuals in a location (Cox and Durrett, 2002). In this situation, each location corresponds to an island in EC approach.

There exist interesting results in the biology field with regard to small subpopulations, in which drift is considerable. There are at least two phenomena, which are of interest, which I address in Chapter 6. The first is *gene swamping* (Alleaume-Benharira *et al.*, 2006) which occurs when migrations have a very strong effect on the target individuals. In such cases the genes in the target environment are lost due to an overwhelming inflow of migrants' genes. Another phenomenon, much subtler, is called the *genetic rescue effect* and occurs when migration helps to maintain diversity in the target environment, even if this migration is still potentially harmful by unifying the overall range of solutions. Migration cannot be too strong, so that the new genes do not dominate the target gene pool. Under genetic rescue, migrants rather should be smoothly combined with the locals, in which case the gene flow may prevent fixation to maladapted alleles.

## 2.6 Applications in engineering design

Design is a process that starts with a realization of a particular need, and through sequential reformulation and specification of the initial conceptual solution proceeds to the detailed stage. Whereas in later stages a solution is usually specified by a set of well-defined parameters and a combination of values fully describes an artifact, it is not so clear how to represent a solution and what algorithms to use in the beginning stages. The design process is often divided into 3 stages: (Kicinger *et al.*, 2005; Pahl and Beitz, 1996): *conceptual design* (tries to produce the initial design), *embodiment*

*design* (uses both qualitative and quantitative criteria, specifying elements of design) and *detailed design* (responsible for final design with quantitative description).

The role of machines in design changed over time. Recently, they not only provide the designer with visualization and construction support (CAD systems), but they take part in the design process itself. Researchers try to devise algorithms usable as early as possible in the design process, to support both defining and solving a problem. Therefore new algorithms search for new concepts, rather than fine-tune existing solutions. Although the latter is obviously important, there already exist methods to achieve optimization. It is more desired that a good engineering design tool produce solutions *qualitatively* different from each other. Even if a given solution is not perfect in all aspects, it may be desired due to a new concept.

EAs applied to search in the space of ideas, concepts, plans or designs, may be perceived as an inventory process (Arciszewski and De Jong, 2001; Goldberg, 2002). In fact, because new solutions are created from previous individuals, the process resembles discovering a new idea by examining existing ones. Selection and propagation of novel ideas seems to be a key to success in engineering design. Even a basic EA occurs useful for global optimization, especially for problems of non-linear, stochastic, temporal or chaotic characteristics (Kicinger *et al.*, 2005).

Design has always been dependent on creativity. The following qualities may characterize a creative person: (Oler, 2004)

- Curiosity and tolerance of the unknown.
- Openness to new experiences.
- Willingness to take risks.

- Ability to both observe the details and see the whole picture.
- No fear of problems.
- Ability to concentrate and focus on the problem until it is solved.

An EA used for conceptual design may require a number of extensions compared to a standard EA, which requires both understanding of engineering and some knowledge about the processes of inventing. The list above can be taken as guidelines for creating a good EA. Curiosity, openness, risk tolerance, no fear of problems and the ability to see the global picture correspond to exploration of search spaces (maybe more than one), and details observing and focusing correspond to exploitation.

A detailed survey of the application of EAs to engineering is given in (Kicinger *et al.*, 2005). EAs have mostly been applied to structural optimization, with further division into three approaches corresponding to the different stages mentioned above:

- *Topology optimization* — most general search for the general layout of the design.
- *Shape optimization* — assumes a fixed topology.
- *Size optimization* — assumes a fixed topology and shape.

Size optimization was not surprisingly the first area to apply an EA, with later research moving toward more abstract areas of shape optimization and topology optimization.

Real-life problems are usually constrained in many ways; it is difficult to come up with one perfect representation since several objectives must be achieved, or at



least addressed, and the requirements may change as the research progresses. For more complex multi-modal and deceptive problems, simpler methods often fail and non-standard EA models are useful.

In these problems, not only is the fitness landscape non-trivial and the search space huge, but it is often the case that the fitness evaluation takes a considerable amount of time. Therefore, an effective EA is truly needed. Furthermore, very often even the search space is not precisely defined, there are many engineering objectives and it is difficult to propose a strict framework for a problem.

It is common knowledge that creative solutions to technical problems are not solved by individuals, but by teams of people representing different technical backgrounds who bring different perspectives to the problem. Non-panmictic EAs may be perceived as representing these different perspectives because various evolutionary mechanisms affect different parts of the model.

## Chapter 3: Methodology

This chapter gives an extended introduction into experimenting with IMs. It discusses the model, technical aspects of my experiments with IMs, and gives an outline of possible measurements used for the analysis. A complete test suite of functions used in the dissertation is given in Appendix C, and some descriptions are repeated in the main body of the dissertation.

### 3.1 The island model

Island models can be seen as a set of constraints on operators. They can be described in the following way:

- selection is restricted to groups (islands).
- recombination is restricted to the same groups.
- migration regroups individuals (and duplicates some).
- mutation operates as in standard models.

Theoretically, a panmictic model can achieve everything that an island model can achieve, since we can imagine that by chance groups of individuals happen not to be recombined with other groups, and individuals of the high fitness in a given group happen to be selected for survival/reproduction as if they did not compete with higher

fit individuals from other groups (at least if we give each individual some chance of surviving). This is, however, extremely unlikely and I am interested in the average behavior of an IM.

Restrictions on selection and recombination are two different things. In theory, groups for selection could be different than groups for recombination (although implementation might be more difficult, and the system would probably quickly lose diversity). We could also create models with just one of the two operators restricted. If only selection was restricted, then panmictic recombination would quickly unify gene pools of each island, and selection would choose the same or similar best individuals. If only recombination was restricted, then global selection would quickly eliminate the worst “branches” of individuals, leaving only the best one. The surviving island would then have to grow in size and/or split into multiple islands. These issues are beyond my scope of research and I focus on a case with local selection and local recombination, restricted to the same groups, i.e. islands.

Migration is an operator counteracting the restriction resulting from separating individuals into islands. When migration moves individuals between islands, it can be seen as regrouping individuals. When migration copies individuals from one island to another, its function is a bit less intuitive — it includes migrants into new groups of recombination and selection, thereby overwriting some previous individuals. In any case migration causes migrants to be evolved in a new context. It is clear that the interaction between selection, recombination and migration, in the context of restriction to islands, is crucial for any specific IM dynamics.

Mutation operates on a single individual, regardless of the population context, and so there is no difference whether it operates in one population, or multiple ones.

We may say that mutation is unaware of the context in which it is operating. As such, mutation gets less attention in the dissertation.

For experiments in this dissertation, I implemented the model described below. While I make a modest use of any formalism in later chapters, this very simple model may eliminate potential misunderstandings. For a more general model, refer to, e.g., Sprave (Sprave, 1999).

I have a set  $P = P_1, \dots, P_N$  of islands (subpopulations). Each island  $P_i$  is a multiset of size  $M$ , including genomes  $a_1, \dots, a_M$ . Genomes can, in general, be of various form, but I use common, fixed-length string representations of length  $L$ , so  $a_i = a_i^1 \dots a_i^L$ . The set of all possible strings of length  $L$  is denoted by  $G_L$ . In case of the binary encoding,  $G_L = \{0, 1\}^L$ , in the case of the real-valued encoding  $G_L = \mathbb{R}^L$ , and, in general, if alleles take values from sets  $V_1 \dots V_L$ , then  $G_L = \prod_{i=1}^L V_L$ . I have two operators, mutation:  $G_L \rightarrow G_L$  and crossover:  $G_L \times G_L \rightarrow G_L$ . I assume that the mutation operator is reversible, i.e., if genome  $a$  can be mutated into genome  $b$ , then also genome  $b$  can be mutated into genome  $a$ . Therefore, the mutation operator introduces a neighborhood relation on the set  $G_L$ , which we can thus treat as a graph  $G'_L = (G_L, E_L)$ , where  $E_L$  are edges connecting pairs of genomes that can be directly mutated into each other. I will omit the prime notation, using simply  $G_L$ .<sup>1</sup> Genomes represent individuals from a set  $\Omega$ . Therefore we always have some encoding function:  $\Omega \rightarrow G_L$ .

Islands evolve independently, but in a synchronous way. This means that the number of generations on each island is always the same. Every migration interval migrations occur. Migrations consist of emigration and immigration. Emigration is a

---

<sup>1</sup>See (Rowe *et al.*, 2004), where the graph of all strings of length  $n$  is denoted by  $G_n$ .

function:  $G_L^M \rightarrow G_L^s$ , choosing and copying  $s$  migrants from some island. Immigration is a function  $G_L^M \times G_L^s \rightarrow G_L^M$  replacing some locals with migrants. In my model the size of an island does not change.

## 3.2 Names, notation and conventions

Studying migrations requires analysis at two levels. One is a *global*, or *inter-island* one, when all islands are taken into consideration. I sometimes call it a *system* level. Another one is a *local*, or *intra-island* one, at the level of single islands.

I use the term *islands*, unless directly referring to a study using a different name (e.g. *subpopulations*). I sporadically use the term *populations* when I want to draw analogies with a single population. This may happen when describing something that is general enough to also apply to single populations, or when islands are separate and could be treated as single populations. Additionally, population can also refer to the contents (i.e. the set of individuals) of a given island. The term *demes*, also used in the literature, is not used in this dissertation. I use the word *system* to refer to all individuals in all islands, and in a few places, the phrase *whole population*, when I treat all individuals as a big population.

When describing a given migration from *source* island into a *target* island, I will call the migrating individuals *migrants*, and the individuals from the target island *locals*.

Because each migration needs to define both emigration and immigration policy, I will use a two-word description denoting what type of individuals are selected or replaced appropriately. For example, I will use random–random, tournament–random or best–random combinations. However, since in my experiments I rarely

use immigration policies other than random, sometimes I describe migrations only with their emigration policy. In particular, I say a *random migration* to denote the random–random policy.

In all figures, evolution of islands in generation  $t$  is drawn as a line connecting points  $(t-1, f(t-1))$  and  $(t, f(t))$  for a measurement  $f$ . Migrations occurring between generation  $t$  and  $t + 1$  are drawn as lines connecting points  $(t, f(t))$  and  $(t, f'(t))$ , where  $f$  and  $f'$  are the measurement before and after migrations. Therefore, all vertical lines on charts correspond to migrations, and all slanted lines correspond to evolution (selection and reproduction operators).

### 3.3 Experimental setup

In this section I describe details concerning implementation and running of experiments.

#### 3.3.1 Statistical significance

Unless written explicitly otherwise, all experiments are done for 60 iterations of a given setup and averaged. This number of iterations gives a small enough variance that, in most cases, I do not plot confidence intervals on the charts, to make them more readable. If confidence intervals are plot, then they are at approx. 95% confidence level (plus or minus 1.96 times the standard error from the mean).

One should remember that “best fitness” will in fact denote “an average of 60 best fitness values,” “locus diversity” is “average locus diversity,” “average fitness” is “average of 60 average fitness values” etc.

### 3.3.2 EA-1 and EA-2

I will mainly use two setups for EAs in islands, EA-1 and EA-2 (De Jong, 2006), which are different mostly with regard to the selection pressure. EA-1 has a weaker selection and EA-2 has a stronger selection. More formally they are characterized by the following parameters:

**EA-1** binary tournament parent selection, no survival selection, non-overlapping model (similar to GA setups)

**EA-2** uniform stochastic parent selection, truncation survival selection, overlapping model with brood ratio of 1.0 (similar to ES setups, in particular  $(\mu + \lambda)$ , where  $\mu = \lambda$ )

Those setups refer only to the selection pressure. Both recombination and mutation at various rates will be used in either of them (unlike traditional ES setups with no recombination). Mutation will often be set at  $1/L$  ratio. In case of binary representation I use a bit-flip mutation, and in the case of real representation, a non-adaptive Gaussian mutation with  $\sigma$  set to 0.01 domain range (so it's quite small). Note, that migrations may be another source of diversity. Migration parameters will be set independently from the EA-1/EA-2 choice.

EA-2 naturally preserves the best individual in the system. EA-1 does not have this property, so sometimes I also use *elitism* (and mention it), understood as replacing a random individual with the best-so-far individual, when the island's best fitness is lower than in the previous generation.

### **3.3.3 Migration mode**

There are some issues with the implementation of migration, as discussed below.

#### **Number of migrants**

There are two possibilities to choose the number of migrants between two islands. One can either first choose the number of migrants in a migration (as this is a standard approach), or one can assign a probability of migration to each individual. In the standard approach we are constrained by the granularity resulting from the island size (since we can only migrate an integer number of migrants). I assign a probability of migration to each individual, because it lets me experiment with arbitrary levels of migration, even for small islands.

In fact, for the random emigration policy choosing migrants according to individual probabilities is equivalent to first choosing the number of individuals from a binary distribution, and then randomly (but without repetition) choosing the actual migrants. One can approximate stronger policies by first choosing the number of migrants, and then using the policy (e.g. tournament-selection) for choosing the actual migrants. A more ambitious approach would be to compute actual probability for each individual and then make a single pass over all individuals to choose migrants, maybe even with a SUS technique (Baker, 1987), which I, however, find unnecessary.

#### **Target island**

In the standard approach a given number of migrants is sent between each two connected islands (depending on the connection topology). This approach is not very useful for studying IMs with various topologies, because when I change the average



number of neighbors, the number of migrants in the system changes too, so these two parameters are coupled. For large  $N$ , and dense topologies, the number of migrants could in fact grow very high, possibly above the total number of individuals in the system, resulting in migrants overwriting each other. For a fully connected topology the number of connections between islands is given by  $N(N-1)/2$ , which is quadratic with regard to  $N$ .

There seem to be two relatively easy ways to keep the migration level constant system-wide, when increasing topology density. One is to keep sending individuals to all neighboring islands, but to send accordingly smaller amount of migrants to each island as the number of neighbors increases (and one can actually find a target for each migrant *individually*). For example, for a fully connected topology, if I wanted an  $\alpha$  fraction of individuals to be migrated between each pair of islands, I would need to set the migration probability of each individual to  $(N-1)\alpha$  (or, in other words, a probability  $p$  of migration results in an average migration of  $p/(N-1)$  fraction of individuals between each two islands). A big number of islands means that interaction between any two of them is relatively small. With big  $N$ , a single island is unlikely to severely affect the content of another island with a single migration (unless the effect is considerably enhanced by evolution). An advantage of this approach would be that migrants are more evenly distributed, which should result in a smaller variance of results between different runs.

A second possibility is to keep the number of individuals migrating between islands constant, but choose only one target island for all migrants, out of a subset of connected islands. A similar solution, called *dynamic* topology was used e.g. by Fernandez (Fernández de Vega *et al.*, 2000). This solution has a few advantages. As

we will learn in later chapters, the size of migration can have a significant impact on the interaction between islands, so it is better to keep it constant when comparing different setups. This method provides for the possibility of varying the topology, since one can treat the original topology graph as a graph of *potential* migrations and choose just one of them. This way I can further limit the speed of information spread in the system (e.g. when I use a ring topology). The real full topology is not achievable, but if I use a full topology graph, for any two islands there is a chance that a migration may occur between them, and all islands are equally distant from each other. Also, even with large migrations, no island is “over-flooded” with migrants. For this migration mode the probability of migration  $p$  and the fractions of individuals sent from one island to another  $\alpha$  are equal.

To keep the same average level of migration with different topologies, one could try to both keep the number of migrants and send them to all the target islands, and perform migrations in accordingly bigger intervals (the average total number of migrants per generation would remain constant). Not only, however, would this result in considerable delays, but as we will see, migration intervals have an even stronger influence on the behavior of IMs, than migration size, so I do not want to change the interval when comparing different setups.

When many islands are used in an experiment, regardless of the migration mode, immigrations from multiple islands sum. This results in a similar average number of immigrants per island, although they have different origins. In the next chapter, where I study migration interaction with other operators in absence of any fitness function, I choose the target island independently for each migrant. In later chapters, where fitness and selection between migrants and locals play significant role, I migrate

groups of migrants to one target island.

### **Unidirectional versus bidirectional**

In a standard migration individuals are copied from one island and then replace some individuals in another island. Such migration proceeds in one direction (it is unidirectional). This obviously causes drift and a loss of genetic material in the whole system, because we obtain two copies of the migrating individual and we lose the individual being replaced. To avoid this we can exchange individuals between islands using a bidirectional migration. In this type of migration, two individuals in different islands switch places and the total genetic diversity is maintained. Because such migration moves migrants in both directions, the net effect of using it is migrating twice as many individuals in total.

One question that remains is what policies other than random–random could be used with bidirectional migrations. If any asymmetrical policy was used (such as best-worst), it would most likely result in favoring certain islands (which the researcher could intend). A possible choice to keep islands equal would be a symmetrical, but not random policy, e.g. the best-best policy. I will use the random–random mode with bidirectional migration.

## **3.4 Tools for island model analysis**

The approach used in this dissertation is, to a greater degree, experimental. Therefore I needed various tools to compare results of experiments and analyze the influence of certain parameters on the behavior of the system. Two sets of tools for studying IMs were developed in the process of writing this dissertation. The first one is a catalog

of appropriate measures, and the second is a set test functions on which I test IMs. Below I give a short summary of them, and all of them are listed in appendices.

### 3.4.1 Measures

The measures used in this dissertation fall into several categories outlined below. Some of the measures were adapted from literature, and others were created especially for my experiments. For a more comprehensive list, refer to Appendix B.

The simplest measures are based solely on fitness values of individuals. *Best-in-run*, *best-so-far*, *system-best*, *island-best* as well as *average* fitness measures all fall into this category. A related measure is to count the number of times the optimum was reached (if such optimum is known). *Selection intensity* measures the relative increase in fitness (with regard to how converged the population is).

Other measures depend on individual genomes. One can count the *number of individuals* defined by distinct genomes in the population, and measure *diversity* as an indicator of their spread. Diversity can be measured *locally*, *globally*, or between islands (*inter-island*).

A spread of individuals from island to island may be measured by a few variants of tracking their genes. Measures in this category indicate the level of so-called represented populations/individuals in other populations/individuals (the four combinations give the *RPP*, *RPI*, *RII* and *RIP* measures). The spread of individuals also can be measured in several ways as a *distance from* a “perfect mixing” (i.e. from *equilibrium*).

Focusing even more on single genes, their *average value*, the number of *fixed genes*, and *survivability* of particular alleles can all be measured. Based on certain genomic

criteria, one can categorize individuals as belonging to a few *types*, and also measure an average impact of such types on performance (i.e., fitness).

### 3.4.2 Test functions

I describe test functions used in this dissertation in appropriate sections and gather them in Appendix C. The test functions can be also gathered into a few groups.

Surprisingly, many experiments require no concrete function at all. In these cases, it is enough either to make only a few assumptions about the fitness, or to assign fitness arbitrarily to given individuals.

For certain experiments, simple functions were created in such a way that they possess required features. The IM1, IM1', IM1'', IM-Trap, IM2, IM2', IM3, IM6, and quadratic functions all fall into this category, although some of them are more complicated than others. A very important concept in this dissertation is composition of partial solutions, and the above functions were created to study this issue. They also show an increasing level of modularity. Two functions were created specifically to test heterogeneous IMs. These are F and F2 functions.

A few other functions are taken from the literature. Watson's H-IFF function is a hierarchical, modular function. Others are Goldberg's Deceptive function and Rosenbrock, Schwefel, Rastrigin and Griewank functions. While not created especially for IMs, they serve as a useful enhancement to the test suite.

Finally, this dissertation analyzes two complex domains. One is related to cellular automata, and the other one to traffic networks. These domains are described in Chapter 8.

## Chapter 4: Two-level evolution

In this chapter I introduce the two-level view of IMs. I argue that IMs have very different dynamics, which difference should be used rather than avoided. To begin, I show an interesting feature of IMs — its ability to reduce random gene fixation, or hitchhiking. These observations lead to a more profound observation, namely that IMs support so-called compositional evolution. Throughout this dissertation, I explore IM behavior in the context of compositional evolution and confront it with incremental evolution. This has an aim of ultimately understanding the internal dynamics of IMs, what problems they are suitable for and how to set up IMs for those problems.

To set the groundwork for more in-depth experiments in later chapters, in this one I present the results of experiments with IM parameters. The parameters were divided into two groups: parameters related to islands, and parameters related to migration between islands. In the first group, I analyze the number and the size of islands. In the second group, I analyze migration size, interval, policy and topology of migrations.

This chapter may serve as an initial guideline for setting up IMs. Recommendations given in this chapter result from analyzing IMs using the two-level viewpoint and suggest benefits from using settings very different from standard EAs. Utilizing these methods, a deeper explanation of the observed phenomena is discussed in the succeeding chapters.

## 4.1 Inter- and intra-island evolution

A common understanding of IMs is that they are a collection of independent EAs, which exchange solutions after they evolved them. In this framework islands “compete” to evolve the best possible solution, and the best island “wins.” Such an understanding suggests a few big islands, so that diversity and evolvability is maintained independently inside each of them. A migration either serves to bring the new best individual (and so one should always choose the best emigrants), or sometimes as a big mutation to maintain a high level of diversity.

The above understanding is one option, but taking a higher view of the islands allows us to see that they may be treated as individuals in a higher-level population. If we prove that there is enough interaction between the islands to support the claim of the higher-level evolution (which is done in this dissertation), the whole IM can be seen as a much more compact entity, even if migrations seem to loosely connect the islands.

Therefore, two levels of evolution can be identified. One is local, *intra-island evolution*, the one that has been studied so far on islands. The other is global, *inter-island evolution*, occurring between islands. It comes as no surprise that those two levels of evolution interact and mutually impact their dynamics. The global evolution uses the local evolution to carry out actual computations and significantly changes its dynamics. The local evolution creates the global evolution. What the exact reasons are for various behavior, and what the implications of such behavior are, we will see in the rest of this dissertation.

## 4.2 Oscillating convergence

The two levels of evolution are two sides of the same bigger system and interact with each other. While we may have more intuition about intra-island evolution, the inter-island level of evolution remains less understood.

It is a well known fact both in the biology and EC community (De Jong, 1975) that evolutionary processes, due to repeated sampling, result in an allele loss due to stochastic sampling. In IMs, islands are relatively smaller, so genes are getting fixed faster and drift (random effects due to stochasticity) becomes more visible. On the other hand, multiple islands can restore diversity with the help of migrations. In the following sections, I analyze the fixation of genes in IMs and try to show how the global level becomes more important, when the local level “stops working” due to gene fixation.

If a mutation is present (at a reasonable rate), then even though new alleles are constantly inserted into an island, few of them survive, and as we know, the island will ultimately converge to a small region. In the absence of a mutation, this process leads to a fixation of genes to single alleles. For the following study, I disregard mutation.

In Fig. 4.1, I show how the average number of fixated genes grows much slower in IMs than in standard EAs (and how migrations counteract fixation) on a OneMax function.<sup>1</sup> In Fig. 4.2, I compare the diversity of alleles for each of 20 genes in a run of a single-population EA and an IM.

---

<sup>1</sup> $\alpha = 0.1$ ,  $i = 10$ , policy = random, topology = full, EA-1, recombination = uniform crossover,  $L = 20$ , no mutation.



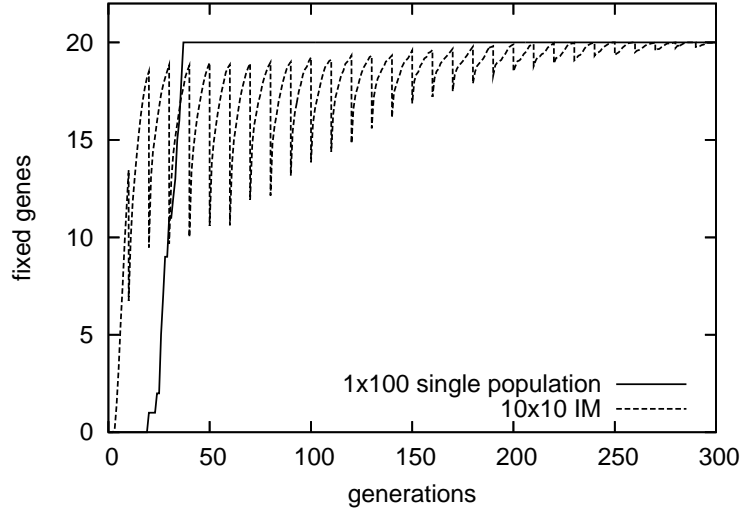
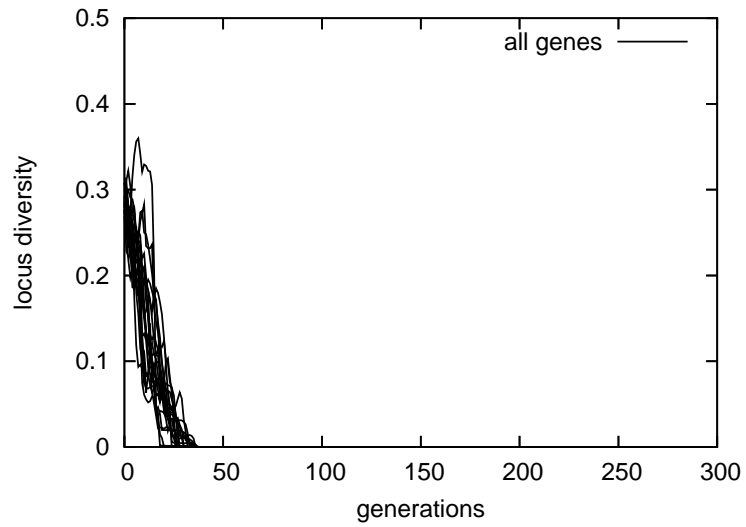


Figure 4.1: Slower gradual fixation in IMs compared to fast simultaneous fixation in a single-population EA.

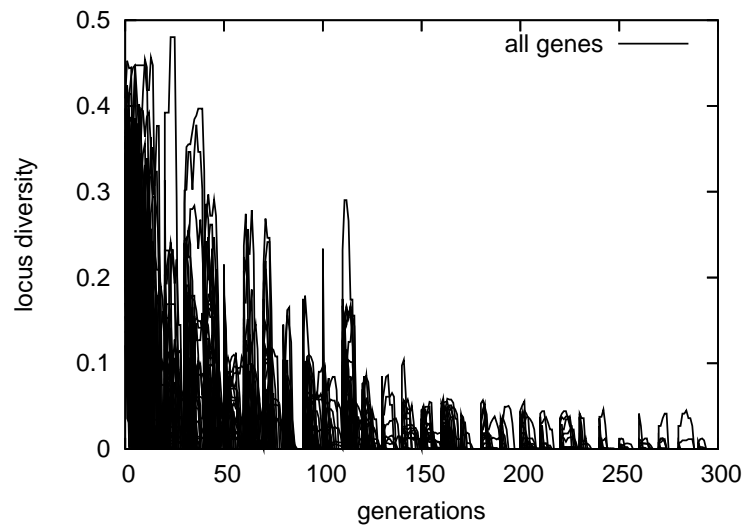
#### 4.2.1 Gene independence in island models

If some genes influence fitness more than others, these genes will be optimized by an EA earlier. This causes a situation in which better alleles surviving for one gene and due to the linkage, alleles of another gene that just happened to be in a good individual, are selected as well. This is called “hitch-hiking.” In some cases, such a situation may cause a fixation of the latter genes to a nearly randomly chosen value.

In IMs, it is unlikely that all genes will get fixed at the same time. This creates a situation in which some genes are already fixed and some still have multiple alleles across islands. This is visible in the example above — for a standard EA, most of the genes converge in more less the same time; for an IM some converge much earlier than others, due to migrations that partially restore diversity on islands. Only those genes that are non-fixed are still being evolved, and the others are not changed by recombination since the alleles are the same for all individuals in an island. Such a



(a) single population (1x100 ind.)



(b) IM (10 islands x 10 ind.)

Figure 4.2: Diversity of each locus in an exemplary single run.

situation in general allows weaker genes to evolve after the more influential ones got optimized. In each island, the less important genes can get fixed to a different value, maintaining their diversity at the inter-island level.

Let us take a function  $IM1'$  defined over  $[0, 1]^2$ , given by (see also Appendix C):

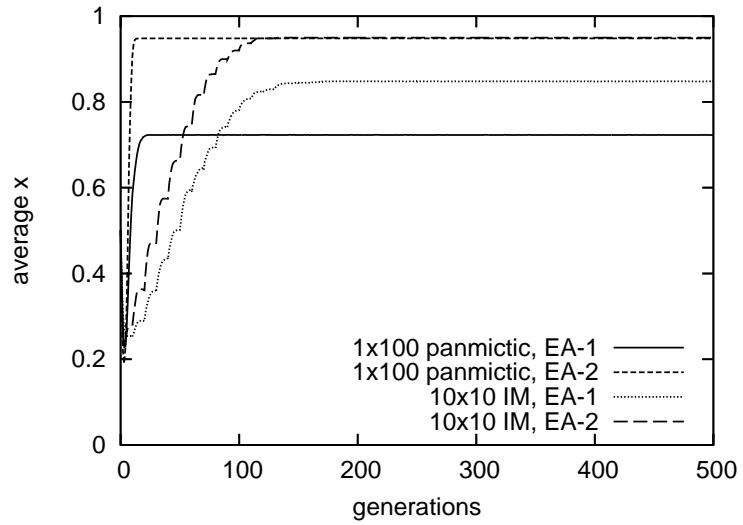
$$IM1'(x, y) = \max(1 - x, 10x - 8.9) + ay.$$

The  $x$  and  $y$  genes are independent. Two peaks exist for the first gene and one peak for the second gene. The parameter  $a$  regulates if the second gene is stronger or weaker than the first one. If  $a$  is big,  $y$  will dominate when calculating the fitness, and  $x$  will get fixed to quite random values. After  $y$  converges, evolution optimizes  $x$ , but since at that time its diversity is low, the process is similar to hill-climbing, often converging to the lower peak, which has a bigger basin of attraction.

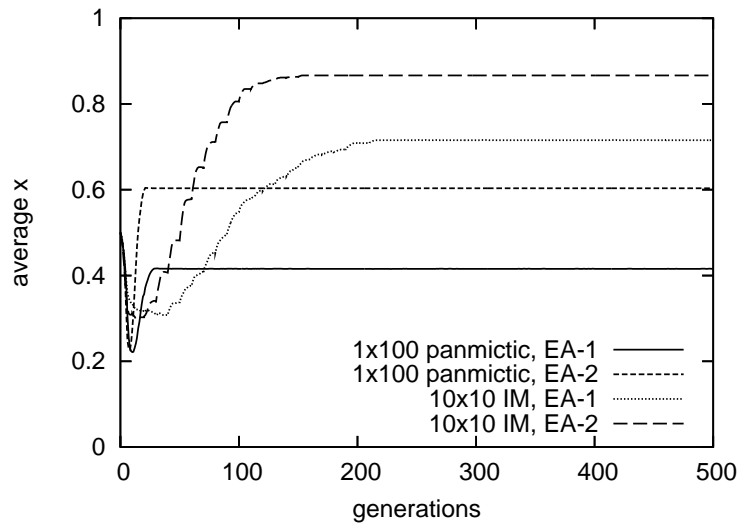
Since the two possible peaks for  $x$  are 0.0 and 1.0, I can measure the average  $x$  value from multiple runs and obtain a percentage with which the model converges to the higher peak. The results of experiments confirm that the use of island models increases the average  $x$  value, comparing it to the panmictic case. In Fig. 4.3, I show a comparison of the average  $x$  with two different values of  $a$ .<sup>2</sup> With weak selection (EA-1), an IM performs better than a single-populational EA, even with  $a = 0$ . With strong selection (EA-2), single-population EA does not “lose” individuals from the good peak, when  $a = 0$ , so it is performing better than an IM (although the IM comes close). However, when  $a = 10$ , the results for a single population are much worse, and for the IM, I observe only a small decrease compared to the previous case.

---

<sup>2</sup> $\alpha = 0.1$ ,  $i = 10$ , policy = random, topology = full, recombination = uniform crossover,  $L = 2$ , mutation rate =  $1/L = 0.5$ .



(a)  $a = 0$



(b)  $a = 10$

Figure 4.3: Average  $x$  (first gene) value, the IM1' function.

We learned that genes are analyzed more independently in IM than they are in traditional EAs. Let us test this hypothesis on one more example with more genes. Imagine a function shown below, with  $L = 20$  and defined over  $[0, 1]^2$  (see also Appendix C):

$$\text{IM1}''(x) = \sum_{i=1}^L (2x_i)^i.$$

The highest-index ( $i = L$ ) gene influences fitness most, but only if the value of the corresponding gene ( $x_L$ ) is above 0.5. The shape of function  $\text{IM1}''$  causes a very fast convergence of the first gene. This behavior in a single-population EA causes the other genes to have less of a chance of being optimized properly.

I have performed experiments, comparing a 100-individuals standard EA with a  $10 \times 10$  IM. EA-2 was used, but mutation was turned off, to stress the effect of losing the alleles. With mutation, both algorithms would clearly have results very close to the optimum, since  $\text{IM1}''$  is a unimodal function. Fig. 4.4 shows that the IM performs better than a single-population EA, both for EA-1 and EA-2.<sup>3</sup>

To understand why an IM achieves better results, I will compare the average change of values of different genes. In Fig. 4.5a, I plot the value of the first gene (least important) against the value of the last, 20-th gene (most important). For a single population the first gene is optimized first (because of its bigger impact on fitness for low gene values), and only then the latter one changes its average value. It is, however, too late to optimize it well, because many alleles have been lost. This asymmetry occurs between any two genes in the function, but the more distant they are, the

---

<sup>3</sup> $\alpha = 0.1$ ,  $i = 10$ , policy = random, topology = full, recombination = uniform crossover,  $L = 20$ , no mutation.

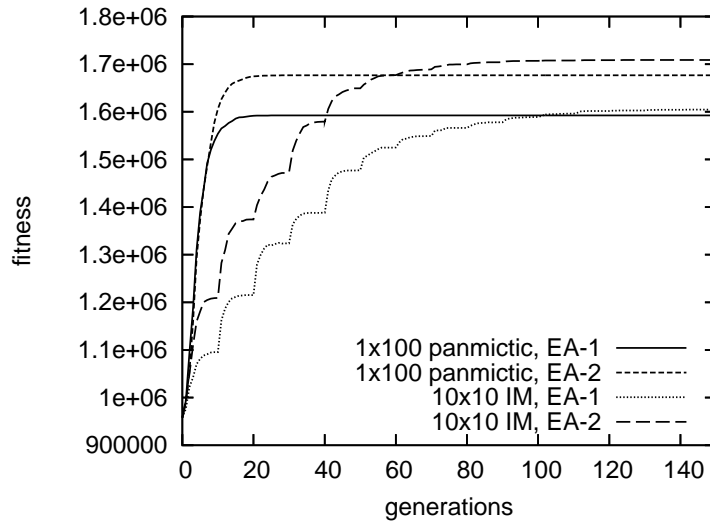


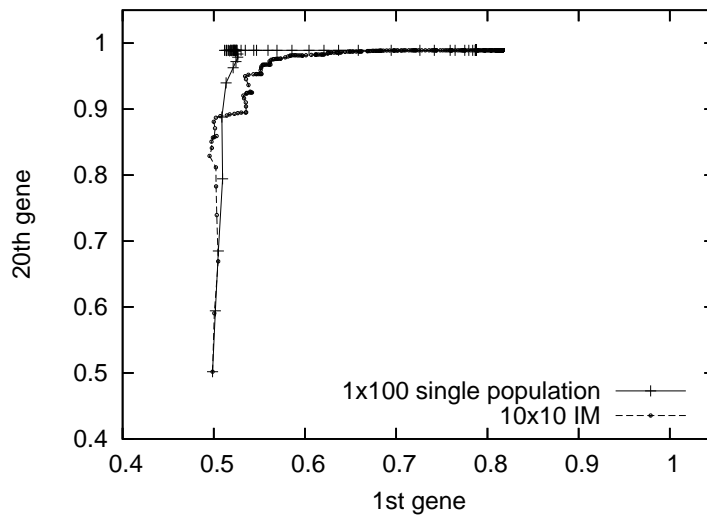
Figure 4.4: Fitness, the IM1'' function.

difference between them is more visible. On the other hand, in the IM we can clearly see, that the convergence of the first gene slows down, allowing stepwise optimization of the latter between migrations. The fact that evolution stagnates (points become denser) is clearly seen toward the ends of the intervals between migrations, and after these migrations additional optimization is possible for both genes.

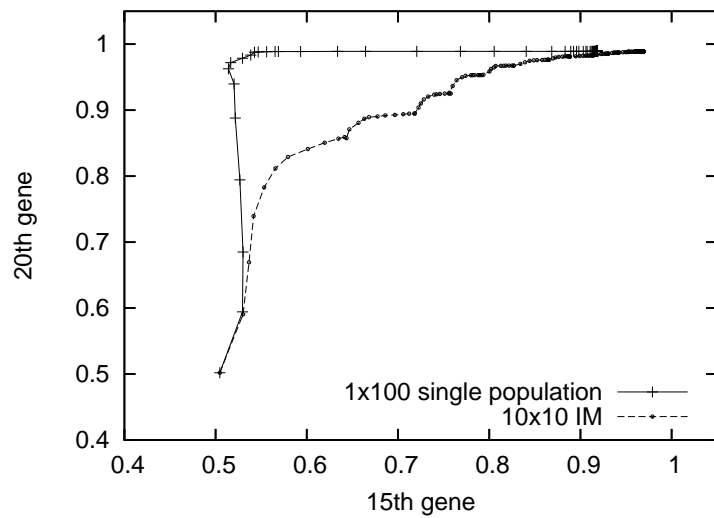
Migration intervals enforce some balance between genes. Stronger ones have to “wait” for the weaker ones. In Fig. 4.5b, we see an analogical plot for the 15-th and 20-th genes, where this effect is visible better.

### 4.2.2 Fixation forming building blocks

If islands converge totally before migrations, migrant groups will be homogeneous and in practice each island can be treated as a single individual in an inter-island evolution. If migrations occur earlier, we can treat the fixed genes as linked, forming



(a) 1st vs. 20th gene



(b) 15th vs. 20th gene

Figure 4.5: Average trajectories of gene values, the IM1'' function, EA-2.

building blocks (BBs). Migrants represent the fixed genes, and migration should correspond to an exchange of such building blocks. This behavior is similar to the SEAM algorithm (Watson, 2006).

In a single population, once genes get fixed, they remain so, and we have no guarantee that the newly formed building block is indeed a good one. In fact, unless selection is very weak and recombination characterized by a high mixing level, some genes will get fixed to suboptimal alleles. On the other hand, in IMs new individuals are inserted into islands with each migration. If the previously fixed building block was on average producing mediocre solutions, there is a chance that genes of better migrants will fill the island, and a new fixation will replace the previous fixation. The difference comparing this to standard EAs is that in IMs an already pre-evolved block of genes may replace other pre-evolved blocks of genes, which may be more probable than a new good building block emerging in one generation with the help of mutation and recombination. Different fixations are formed inside different islands, and they do not compete with each other until migrants are sent between islands. If the fixation in an island was good, migrants have a smaller chance to change the dominating individuals. As a result, with time, better fixations should survive.

Even if migrations introduce new alleles into the target island, a fixation occurs systemwide too. After some time, more and more of migrants' genes will be identical to the genes of local individuals. This will also cause an average number of fixed genes in each island to increase. The new larger "building blocks" should be composed of good smaller building blocks fixated earlier.

For a small number of islands  $N$  and the big island size  $M$ , BBs should be mostly created in a standard way by EAs inside the islands. When  $N$  is larger at the cost



of a smaller  $M$ , then the inter-island BBs competition might play a bigger role. Still, with a reasonable size of  $M$ , BBs created by fixation will be optimized by the EA (local optimization) before migration. In a special case when  $M = 1$ , fixations inside islands are created immediately.

### 4.3 Compositional evolution

In this section I show that IMs support compositional evolution. The existence of two levels of evolution, and of the oscillating convergence, suggest that local evolution might be able to construct building blocks, and the global level evolution may be efficient at mixing them. Generally, functions, which have solutions built from multiple sub-solutions, should benefit from using IMs. The examples in this section are a specially created IM6 function and the H-IFF function taken from (Watson, 2006). Throughout this dissertation I will explain deeper the processes underlying compositional evolution in IMs.

#### 4.3.1 The IM6 function

IM6 is defined on real-valued arguments over  $[0, 1]^n$  by the following formula:

$$\text{IM6}(x) = \prod_{i=1}^n \frac{(x_i^2 - 0.45)^2}{0.3025}.$$

The IM6 function has local optima in each corner of  $[0, 1]^n$ , and the global optimum in  $(1, \dots, 1)$ , equal to 1. See Appendix C for a plot in two dimensions.

In fact, we have already seen in the Chapter 1 (in Fig. 1.1) that an IM performs better than both a single-population EA, and isolated populations. These results

were obtained with uniform crossover, because for IM6 each gene can be treated as a separate module, and distance between them is not important in terms of fitness.

Function IM6 is used throughout most of this chapter to test the behavior of IMs of various settings.

### 4.3.2 The H-IFF function

Function H-IFF is the original function used by Watson to test compositional evolution. The domain is binary. The evaluation procedure first checks for two-bit modules having the same bits (either 00, or 11) and starting at an odd locus number. Each such a block contributes to fitness. Then, blocks of length four are tested, in positions  $4i \dots 4i + 3$ , for  $i = 1 \dots L/4$ . They exist only if matching submodules were earlier found in the left and right half of the four bits. This checking is performed with longer and longer modules, until the whole genome (presumably of length being a power of 2) is tested. Each next level has two times less potentially available blocks, but their weight is doubled, so they have the same contribution to the general fitness. This is done on purpose to balance the importance of intra-module and inter-module relations. As shown by Watson, the H-IFF can only be effectively solved by EAs using composition (recombination).

In Fig. 4.6, I show results for a configuration in which IMs perform better than both single-population and isolated setups. I used the same configuration as for IM6, except that genome length was 64, mutation rate appropriately lower ( $0.015 \approx 1/L$ ) and I used a two-point recombination, because for the H-IFF function close bits may belong to the same module.

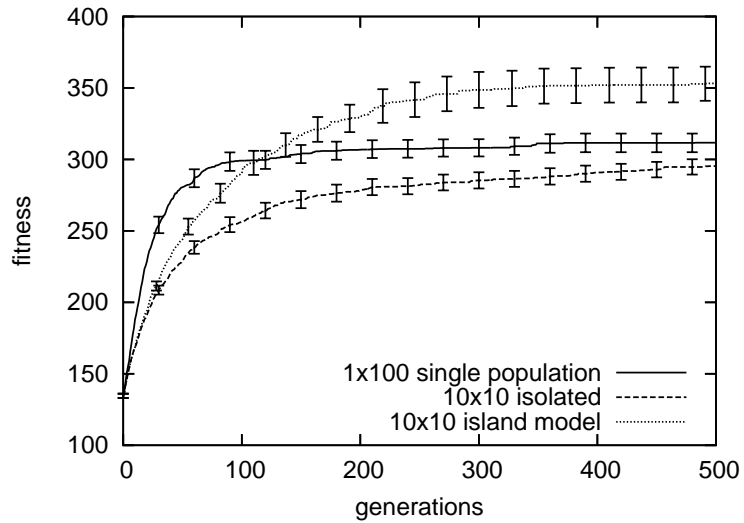


Figure 4.6: Island model obtains much better result than both single-population and isolated multi-population models, on the H-IFF function.

The average fitness obtained for an IM is significantly higher than the fitness values obtained for both panmictic and isolated models. Because of a special construction of the H-IFF function, these results confirm that IMs support compositional evolution.

## 4.4 Parameter analysis

We have seen that IMs are yielding good results on compositional problems, if only they are able to construct different sub-solutions, and compose them to create better solutions. Therefore, I use the function IM6 to test the influence of different parameters on IM behavior. By using this function, I am able to analyze the IM behavior in situations when they perform well, and thus, study aspects characteristic for them.

The experiments will use a basic setup, with moderate values of each parameter. To study certain parameter influence I will change its value usually in both directions.

Unless specified otherwise, the following setup is used:

- $\alpha = 0.1$  migration size.
- $i = 10$  migration iteration.
- random–random migration policy.
- (dynamic) full topology.
- binary tournament (EA-1) or stochastic (EA-2) parent selection.
- uniform one-child crossover at rate 1.0 (used always).
- $N = 10$  islands.
- $M = 10$  individuals per island.
- $L = 20$  real genes.
- $1/L$  mutation rate.
- no explicit elitism (in EA-2 it is implicit).
- deterministic (EA-1) or truncation (EA-2) survival selection.
- brood ratio of 1.0 (each parent on average produces one child).
- overlapping (EA-1) or non-overlapping (EA-2) generations.
- IM6 function.

### 4.4.1 Number of islands

As expected, the number of islands determine about how many areas the system converges to, because each island can converge to a different individual (see experiments with selection alone in Chapter 5).

All islands are usually initialized using the same procedure, but due to their finite sizes and stochasticity (in the form of sampling) they follow different evolutionary paths. Assuming the same EA on each island, stochasticity is the initial source of diversity between islands. Later in the run this diversity may be either enhanced or diminished, depending on the settings of an IM.

There exist some implementation reasons that influence the number of islands. It is usually a good idea to keep a whole island on one machine, if possible, which suggests that we will have at least as many islands as machines. This, however, makes sense only if the evaluation times are much faster than sending individuals over the network. Otherwise, if time for spreading individuals among machines is insignificant compared to the evaluation time, we can treat the whole cluster (or grid) as a uniform computing platform. An obvious upper limit for the number of islands is the amount of computational resources that is available.

However, there are still theoretical reasons to use multiple islands. One of the earliest explanations of IM mechanisms, given by evolutionary biologists, showed the benefit of a distributed search and spreading good solutions among islands. Restricted selection allows for different levels of optimization in different islands. Individuals that would get quickly eliminated in a single population, can now survive in their niche. Such individuals can occur useful later.

I will use the IM1 function to demonstrate this (see Appendix C). Function IM1

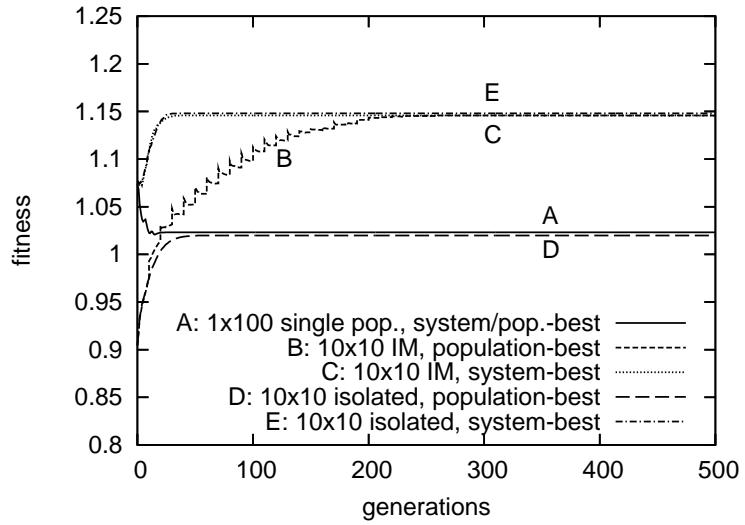
consists of one wide low peak and one narrow and high peak. A single big population running on this landscape most of the time would converge to the suboptimal peak. A weaker selection (meaning more exploration) would not help here, since it would be more difficult for the few individuals on the higher peak to dominate the population. However, if I used a set of smaller islands, a few of them would converge to the higher peak. In addition, if migration was used, this optimum would spread throughout the whole system.

In Fig. 4.7,<sup>4</sup> I show that the above multiple island mechanism occurs indeed. In each chart I plot both average island-best fitness, as well as system-best fitness. Using EA-1, I observe that isolated populations *on average* reach the same maximum fitness as the panmictic population. However, the maximum of their best fitness values is much higher, which means that usually there is at least one island that reaches the higher peak. As already mentioned, in the island model case this solution is then spread to other islands, raising the best individual in each island to the optimal value. For EA-2, the selection is strong enough to not “oversee” any individual that would lead to the better optimum. Therefore the panmictic population, having bigger chances of generating an individual in the higher peak’s basin of attraction, on average converges to a higher fitness than a single smaller island. However, again, a set of isolated islands reaches a similar high fitness in at least one island, which is then spread into other islands when migrations are used.

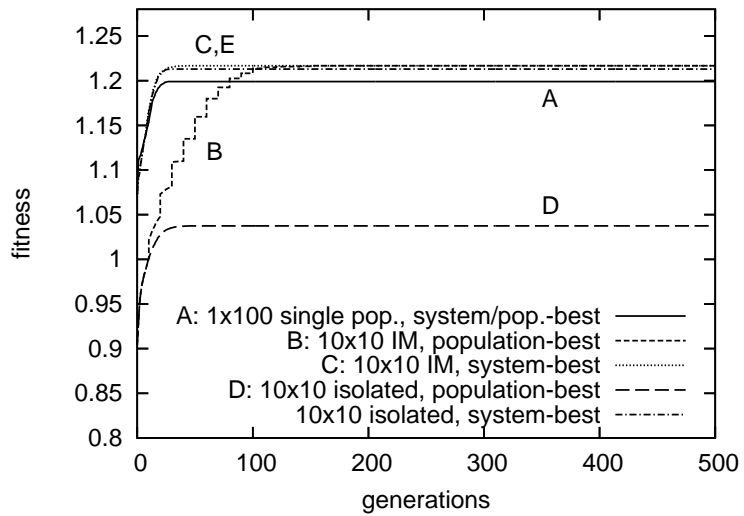
As we have seen, suboptimal solutions from different islands may be effectively recombined to create higher level solutions. Therefore, if the number of islands is too small, it may be difficult for an evolutionary algorithm to find solutions for modular

---

<sup>4</sup> $\alpha = 0.1$ ,  $i = 10$ , policy = random, topology = full, parent selection = binary tournament, recombination = uniform crossover,  $L = 2$ , mutation rate =  $1/L = 0.5$ , IM1.



(a) EA-1



(b) EA-2

Figure 4.7: Island models converge to the better peak due to restricted selection.

or hierarchical functions.

Obviously, the total number of islands limit how many islands can be represented in any single island (the *RPP* measure in Appendix B), and more importantly how many different sub-solutions can be mixed together, assuming that islands converged to some sub-solutions. In fact, the number of islands also affect the remaining measurements: *RPI*, *RIP* and *RII*.

In a standard IM, when two islands converge to the same region, it is very unlikely that they will later diverge. Therefore, the number of regions represented by individuals will only keep decreasing. We can, however, imagine a situation when a recombination of individuals from two islands causes one of them to converge to yet an unknown, new region. Furthermore, it is theoretically possible that two (or more) islands would repeatedly help each other to reach new better regions of solutions (such behavior would let a smaller number of islands to visit a bigger number of local sub-solutions). Therefore, the number of different regions represented in the system need not necessarily drop when migrations occur. Since  $N$  is practically the upper limit on the number of areas that the system is analyzing in a given moment, some knowledge about the complexity of good solutions can give insights about the minimal  $N$ .

An always valid question is what would be the benefit of using twice as many machines as you did so far. Would it double your solving capability (and what does it mean exactly)? The answer is, to a greater degree, dependent on the problem you are solving. Experiments with the IM6 function and different values of  $N$  show that increasing the number of islands is useful until it reaches the level when the global optimum is usually found. Increasing  $N$  further may speed up to some degree the



convergence, but generally seems questionable. This is shown in Fig. 4.8–4.9.<sup>5</sup>

Let us compare how changing the number of islands can affect the diversity. In Fig. 4.10 I show the diversity for the experiments with IM6 function (using EA-1). We see that, for  $N$  equal to 5, 10 or 50 (the curves on the plot overlap), the local diversity is maintained longer than for the case of  $N = 2$ . There is also a noticeable increase in the inter-island and global diversity as  $N$  grows. It may seem that local diversity in each island should drop at a rate unrelated to the number of islands. As we see, however, this is not true, because with more islands, inter-island diversity grows higher and “help” islands maintain diversity. For a further analysis of these observations, see the analysis of migration in Chapter 5.

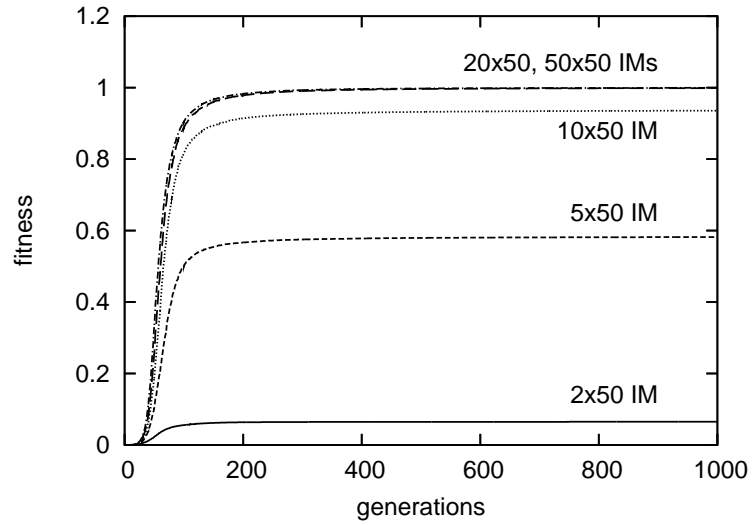
#### 4.4.2 Island size

Changing island size in IMs has similar implications as changing it in standard EAs. In general, however, bigger islands provide better exploration and therefore produce better results. With  $M \rightarrow \infty$  islands start to behave more similarly to each other, following the theoretical evolutionary path. With smaller  $M$ , stochastic effects play a much bigger role. The difference is that while in standard EAs this was in general an undesired behavior, in IMs drift may actually sometimes be useful. A small island size results in the island converging fast and fixating to a given sub-solution.

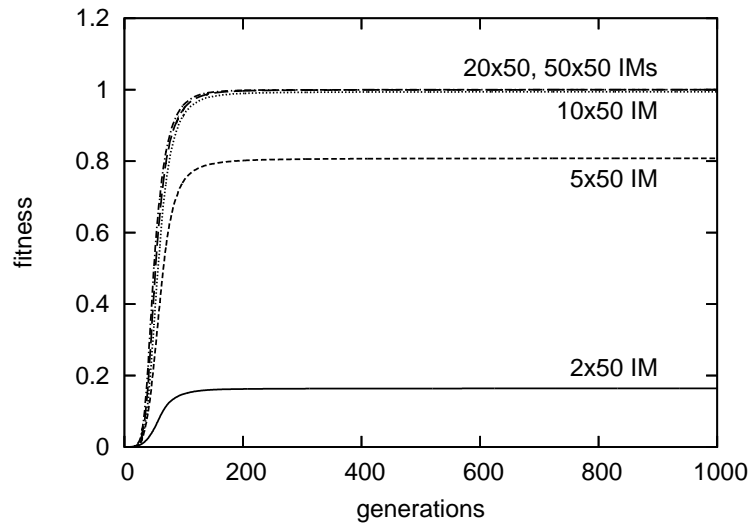
However, islands in IMs are responsible for a more local aspect of search, compared to inter-island behavior. Therefore, intuitively, if a lot of local “tweaking” is required, islands cannot be too small. Furthermore, when increasing the island size, there is

---

<sup>5</sup> $\alpha = 0.1$ ,  $i = 10$ , policy = random, topology = full, parent selection = binary tournament, recombination = uniform crossover,  $M = 50$ ,  $L = 20$ , mutation rate =  $1/L = 0.05$ , non-overlapping, IM6.



(a) system-best, EA-1



(b) system-best, EA-2

Figure 4.8: System-best fitness curves for different number of islands ( $N$ ), the IM6 function.

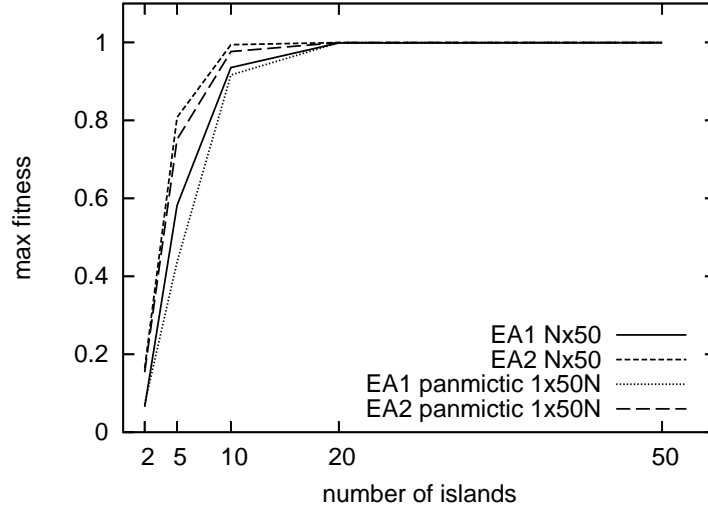
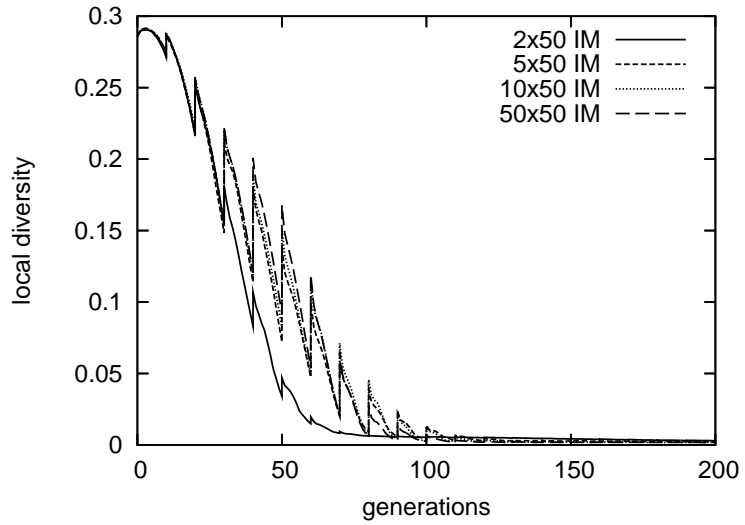


Figure 4.9: Best-in-run fitness values for different number of islands ( $N$ ), the IM6 function.

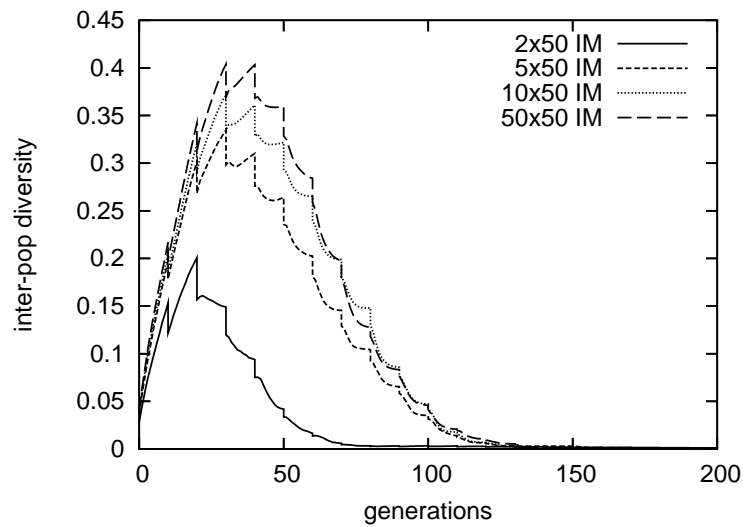
more space for storing genes coming from other islands and individuals. Therefore, both  $RPP$  and  $RIP$  can reach higher values. If migrant genes survive, islands will be better mixed with each other (and the  $d$  measurement will come closer to zero, as the system comes closer to the equilibrium). Such a situation is obviously favorable for modular/hierarchical functions where mixing sub-solutions is important.

I have performed experiments changing the island size, with the IM6 function. As expected, in Fig. 4.11 and 4.12, we see that increasing the island size has a positive effect on the performance.

Diversity in the experiments with the IM6 function (using EA-1) is shown in Fig. 4.13. Whereas smaller islands converge quite fast, bigger ones lose diversity slightly slower (e.g, for  $M = 20$  local diversity does not drop to zero until well beyond 100 generations, for  $M = 50$  the period seems shorter, but is also close to a 100 generations). After a few migration intervals though, in all cases local diversity is

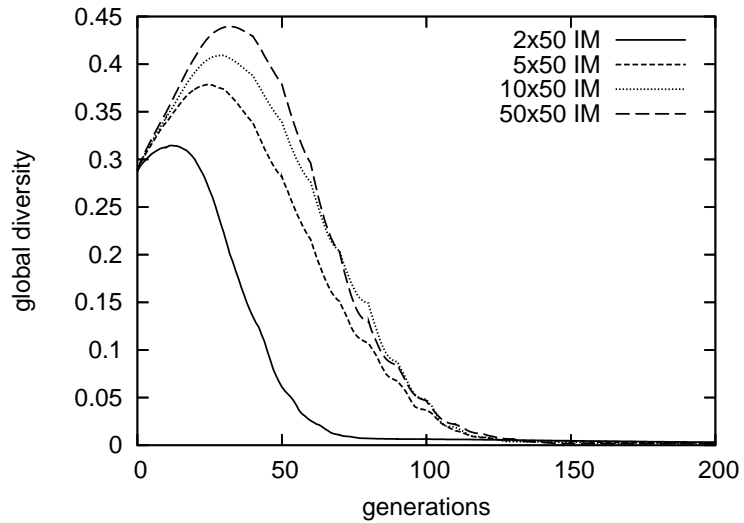


(a) Local diversity



(b) Inter-island diversity

Figure 4.10: Diversity for different number of islands ( $N$ ), the IM6 function, EA-1.



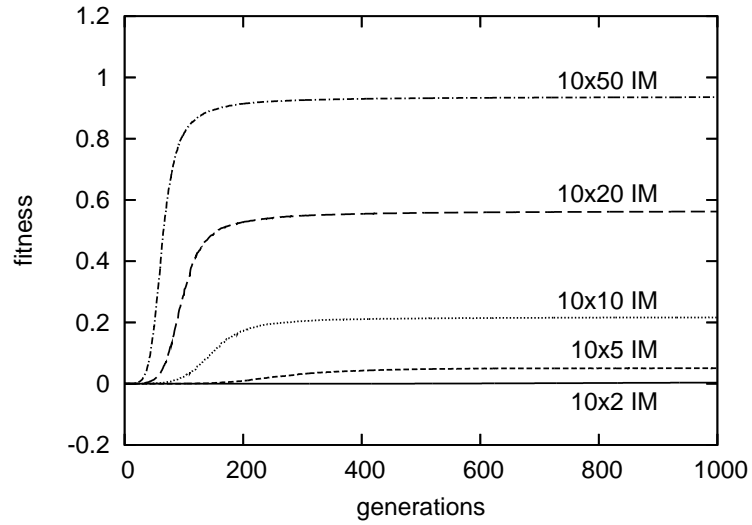
(c) Global diversity

Figure 4.10: Diversity for different number of islands ( $N$ ), the IM6 function, EA-1.

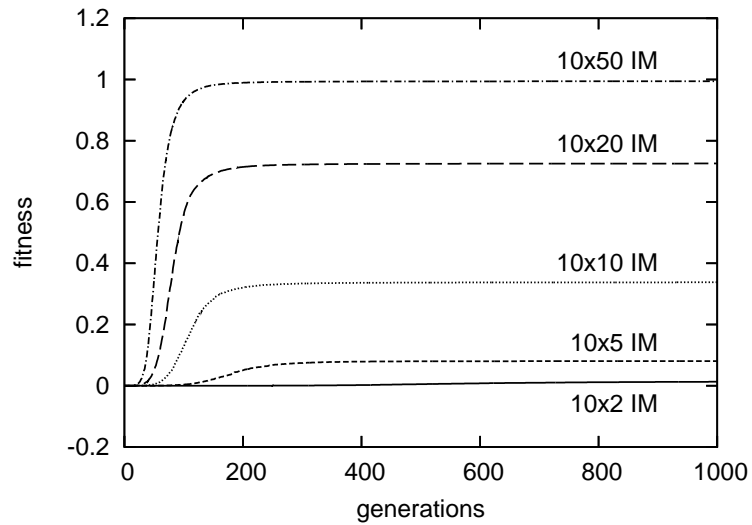
very low. What is interesting is that in the case of small islands, the global and inter-island diversity remain high for a longer time. When we look closer at the inter-island curve, we see that in all cases migrations cause a temporary decrease in this diversity. However, for bigger islands we observe a further decrease in periods between migrations, and for smaller islands we observe an increase. This suggests that in one case the effect of migration is enhanced, and in the other, this effect is disregarded. I will come back to this issue analyzing migrant gene survivability in Chapter 6. For now, I will just say that its easier to mix individuals in a bigger island.

### 4.4.3 Large number of small islands

The number of islands and their size correspond to two levels of granularity in IMs and are strongly related to each other. In this section I study a balance between them and get some insights about the two-level evolution of IMs.



(a) system-best, EA-1



(b) system-best, EA-2

Figure 4.11: System-best fitness curves for different island sizes ( $M$ ), the IM6 function.

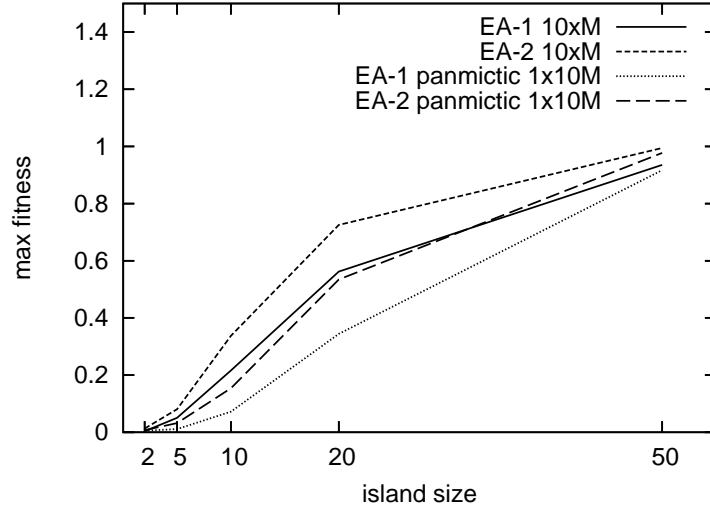
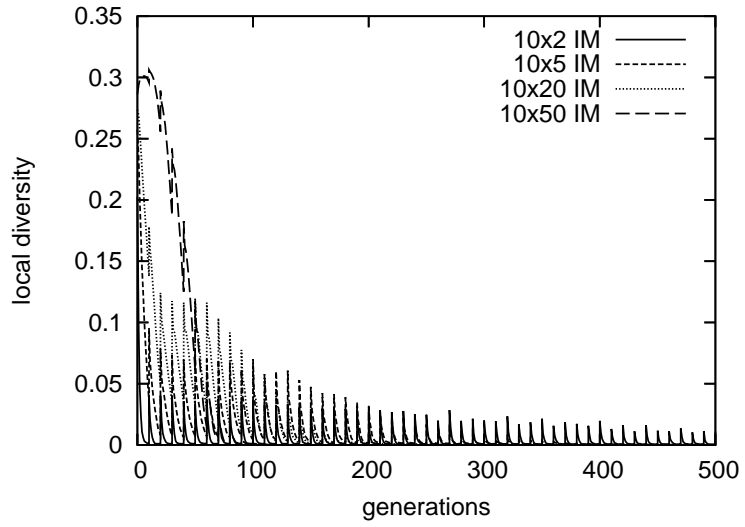


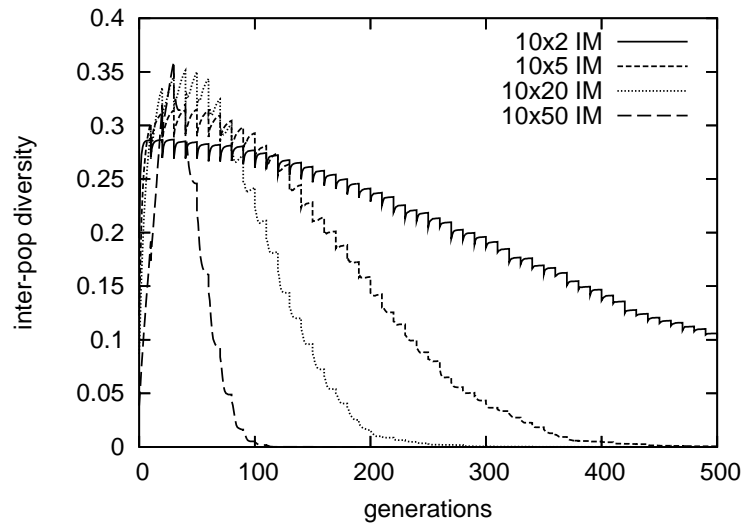
Figure 4.12: Best-in-run fitness values for different island sizes ( $M$ ), the IM6 function.

If the total number of individuals  $W = N \cdot M$  is limited, island size is coupled with the number of islands. The bigger the number of islands, the smaller the island size. An obvious question then is how to split individuals? I choose to measure how the solving ability change, focusing on the maximal fitness to which the EA will converge. In the case of a hierarchical function, where a combination of sub-solutions yields a much better result, a proper mixing of various sub-solutions must be enabled.

Islands cannot be too small, but also there cannot be too few of them. So, there is some middle ground, where the optimum is reached. Where exactly it is, it depends both on a particular problem (the difficulty at finding and combining sub-solutions) as well as the EA used on the island. In Fig. 4.14, I show some results for the IM6 function. We see that the weaker (in terms of selection) EA-1 algorithm observes a drop in the maximal fitness, when islands become smaller than 5. The more exploitative EA-2 is able to effectively evolve solutions even with very small islands



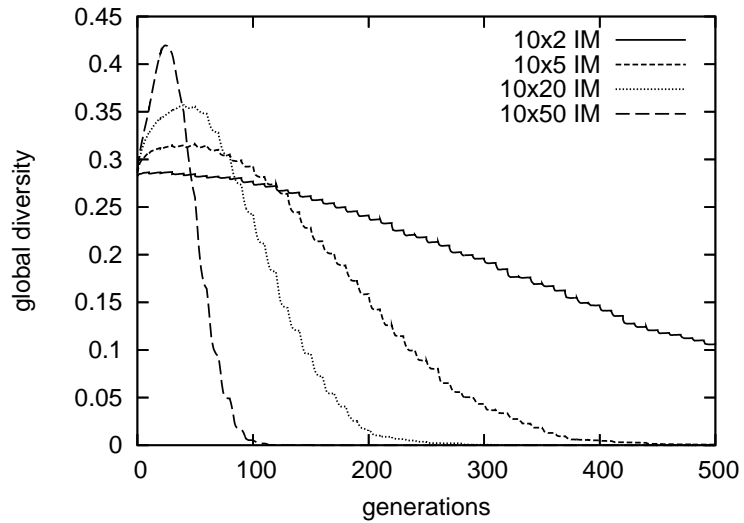
(a) Local diversity



(b) Inter-island diversity

Figure 4.13: Diversity for different island sizes ( $M$ ), the IM6 function, EA-1.



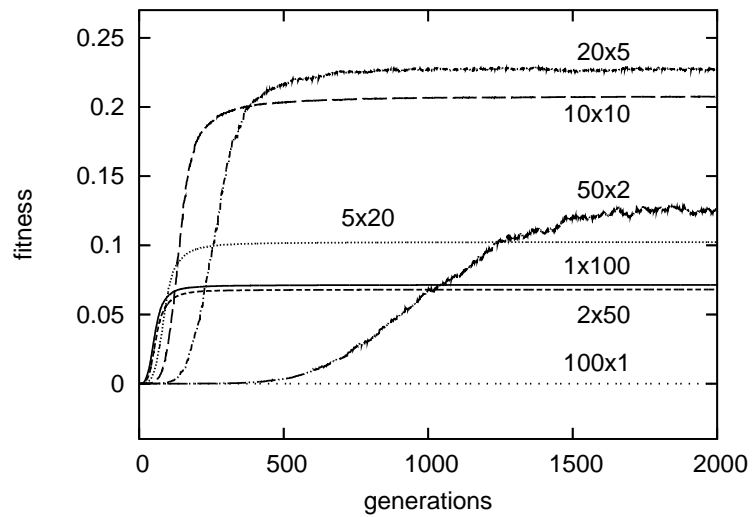


(c) Global diversity

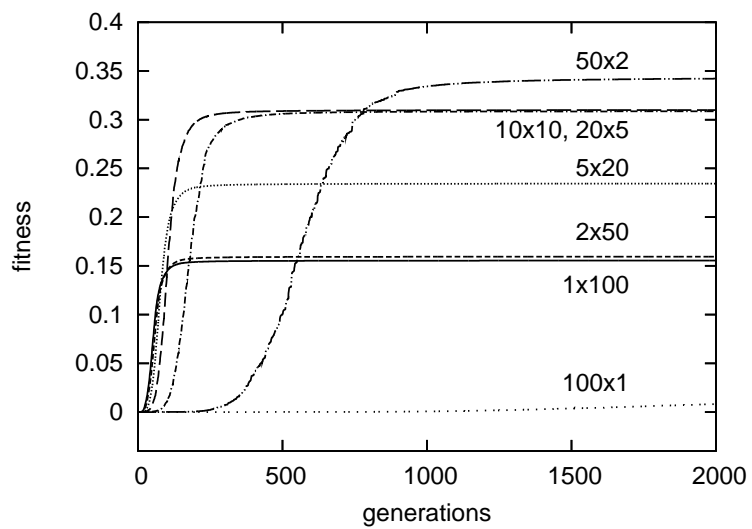
Figure 4.13: Diversity for different island sizes ( $M$ ), the IM6 function, EA-1.

of size 2 — and use the bigger number of islands to achieve better results. For other multi-modal and hierarchical functions the results were qualitatively comparable, and results are shown in Fig. 4.16.

The results look like the  $N = M$  balance between the number of islands and islands size is reasonable, and it is in agreement with the understanding of IMs as two evolutionary systems — one at the local level inside islands, and another at the global scale, between islands. As we see, for simpler functions even a  $N > M$  setup seems good, especially since the island local diversity is increased with every migration, but nothing helps the global evolution. However, I expect that for more complex domains, one may need a significant amount of resources for the “local” improvement of sub-solutions, and the global scale would have a (probably) less significant function of mixing the partial solutions, so generally islands cannot be too small. Note that for the extremely small islands with a single individual, EA-1 becomes random walk



(a) EA-1



(b) EA-2

Figure 4.14: System-best fitness curves when keeping  $N \cdot M$  constant, the IM6 function.

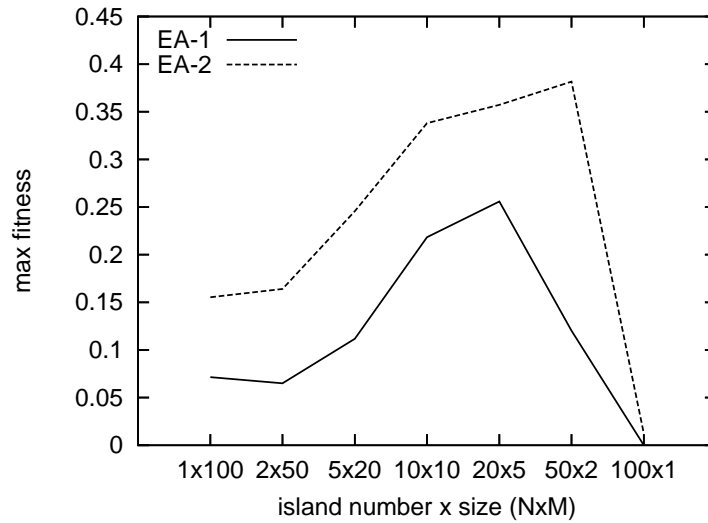


Figure 4.15: Best-in-run fitness values when keeping  $N \cdot M$  constant, the IM6 function.

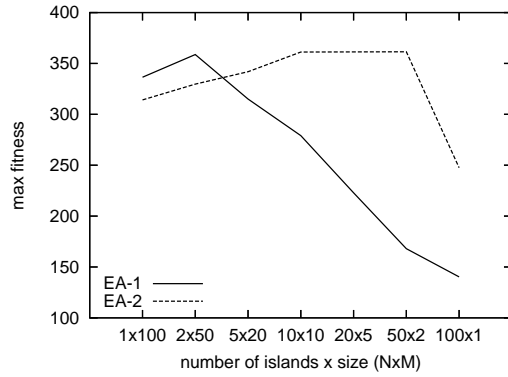
and EA-2 becomes a stochastic hill-climbing.

#### 4.4.4 Migration size

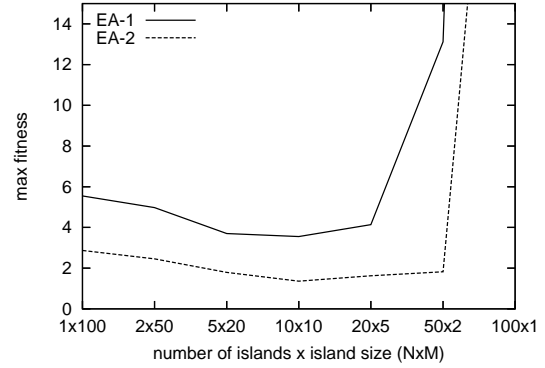
After gathering some insight into splitting individuals into islands, I will now study the parameters of migration. Migration is a central concept in IMs — understanding its impact should give us a good understanding of IMs.

Migration causes an increase of local diversity, because it inserts into an island individuals very different from the ones to which the island converged. Migrations, especially when many islands exist in the system, may keep the local diversity high for a long time, effectively counteracting the decrease caused by selection. On the other hand, a larger migration decreases system-wide diversity (unless migration simply exchanges individuals) and speeds up global convergence.

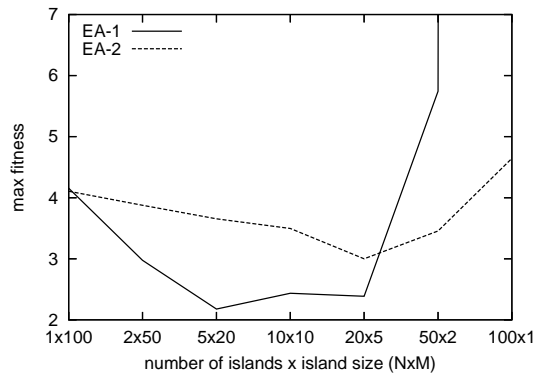
I studied the EA behavior on IM6 with a variable migration size. Fig. 4.17–4.18



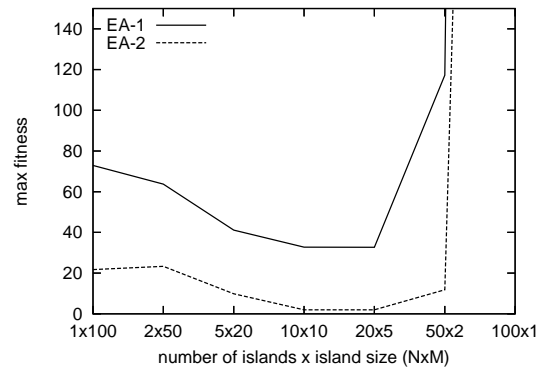
(a) H-IFF (maximization)



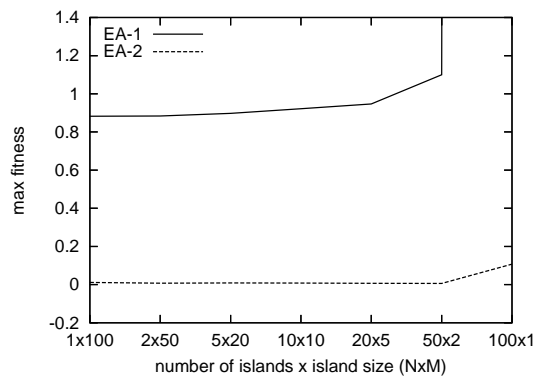
(b) Rastrigin function (minimization)



(c) Rosenbrock function (minimization)

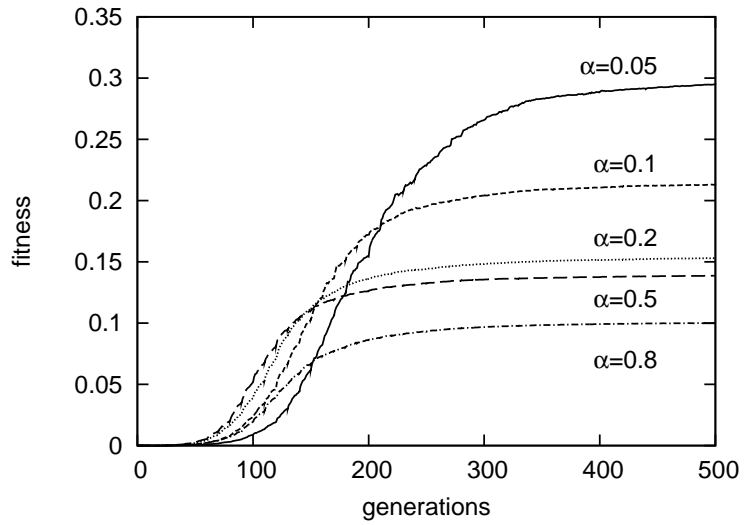


(d) Schwefel function (minimization)

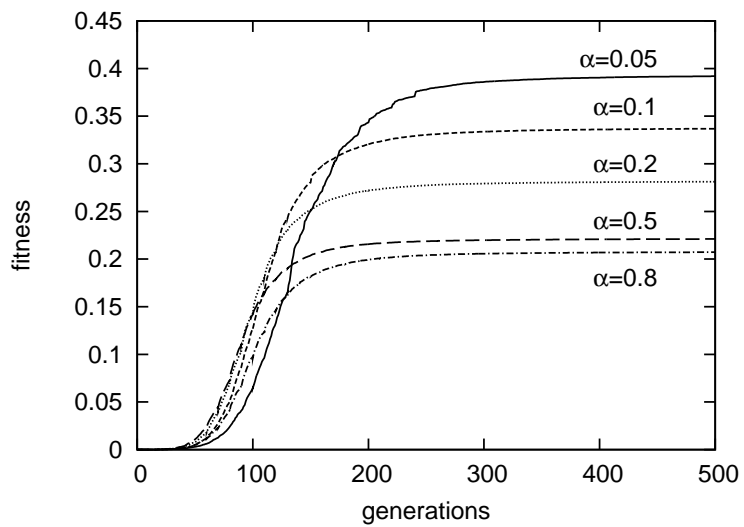


(e) Griewank function (minimization)

Figure 4.16: Best-in-run fitness values, when keeping  $N \cdot M$  constant, various test functions.



(a) EA-1



(b) EA-2

Figure 4.17: System-best fitness curves in runs with different migration sizes ( $\alpha$ ), interval = 10, the IM6 function.

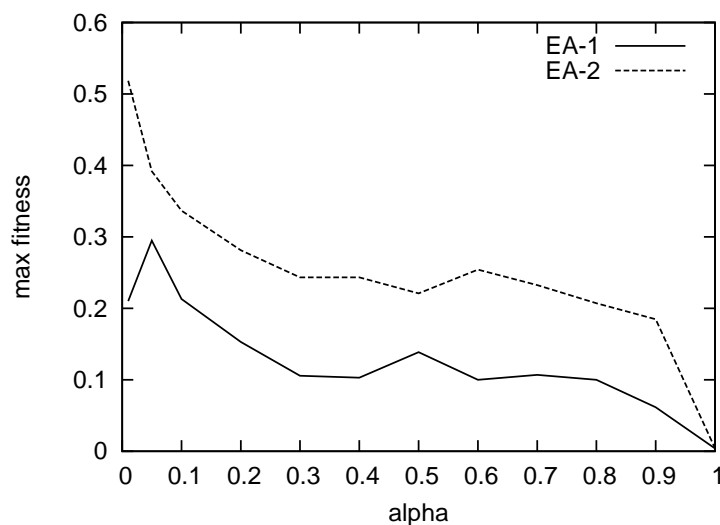
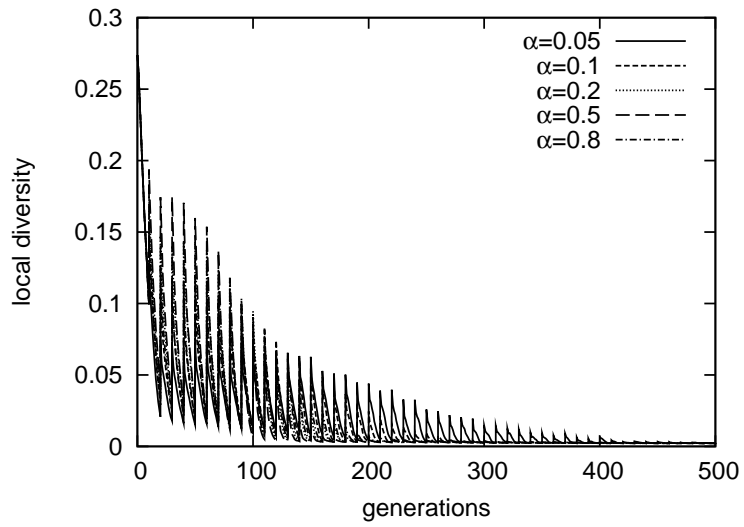


Figure 4.18: Best-in-run fitness values in runs with different migration sizes ( $\alpha$ ), interval = 10, the IM6 function.

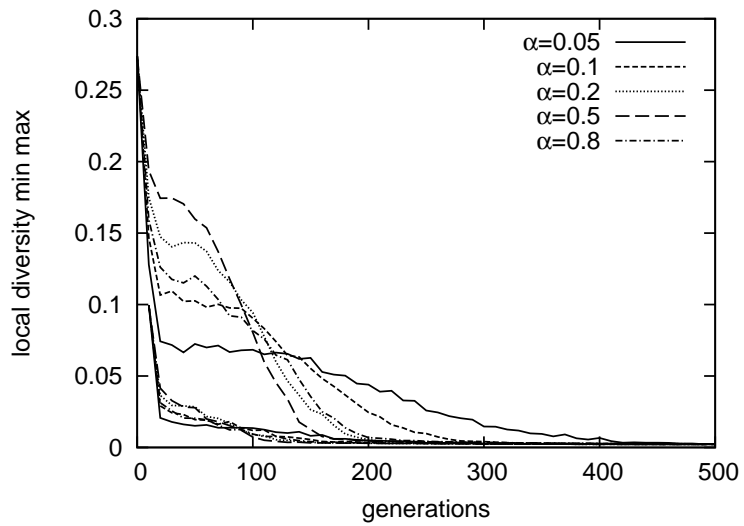
show fitness plots, and Fig. 4.19 shows diversity. For each configuration the local diversity observes rapid increases due to migrations and decreases due to selection. This makes the lines in Fig. 4.19a overlap and it is hard to tell them apart. Therefore, in Fig. 4.19b I plot the boundaries of each curve variations — i.e., for each of the 5 lines in the first chart, I plot two lines of the same type, marking the maxima and minima of each original line. I will use this technique also for other figures.

I observed that migrations of even single individuals often result in a bigger fitness increase compared to the case in which a large number of individuals migrate. Smaller migration levels have a smaller impact on local diversity initially, but clearly result in keeping all types of diversity longer. Generally, small migrations are better and increasing migration size  $\alpha$  worsens the behavior.

For very small migrations, I have sometimes observed an increase in diversity in generations following a migration, which suggests that a single individual is able to act

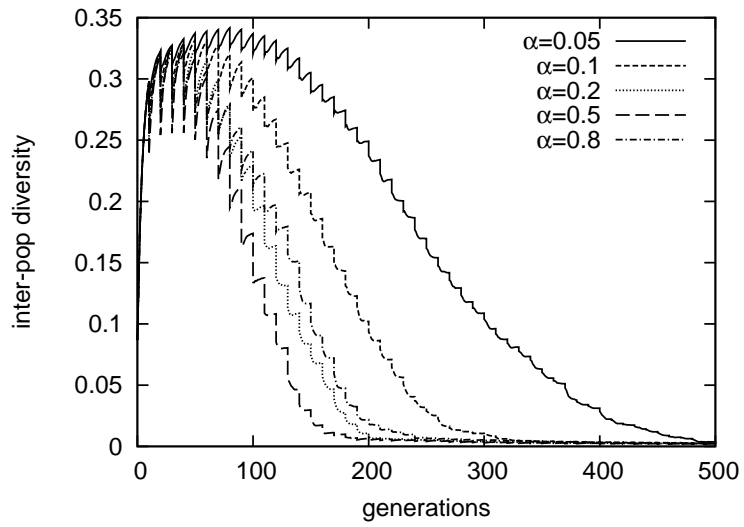


(a) Local diversity (see also the next subfigure)

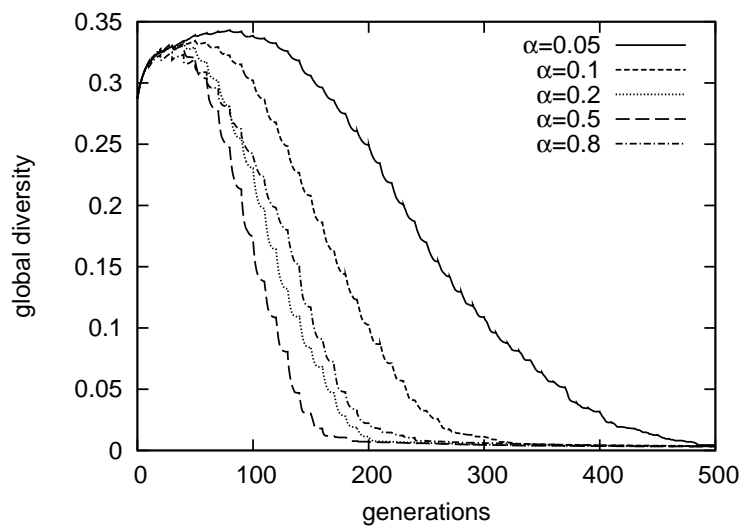


(b) Local diversity (a simplified plot showing boundaries)

Figure 4.19: Diversity in runs with different migration sizes ( $\alpha$ ), interval = 10, the IM6 function.



(c) Inter-island diversity



(d) Global diversity

Figure 4.19: Diversity in runs with different migration sizes ( $\alpha$ ), interval = 10, the IM6 function.



as a seed for changes in a stagnated island. Big migrations tend to have an immediate effect on diversity simply because of replacing a larger number of individuals. However in such cases, the diversity does not increase further in consecutive generations.

Migration size changes the interaction between islands. In Chapter 6 I will study this relation deeper and the impact of migration on after-migration dynamics.

#### 4.4.5 Migration interval

Recombination restricted to islands means that mixing solutions from different islands occurs only after periods of isolation (when migrants recombine with the target island individuals). It is tempting to think that when genes are independent, then early recombination is always a good choice. Indeed, in such a case the value of one gene does not influence the optima of the other, so why wait with recombination until the convergence of islands, meanwhile losing alleles? Are IMs of any use in such a situation? As we have seen, sometimes it is crucial to converge to several optima at the same time, and this is highly improbable with early recombination. It is desirable to maintain global diversity until recombination of different sub-optima is possible, otherwise we would not benefit much from this operator. Longer intervals make islands converge more independently (and separates the local and global evolution levels more), which causes rare migrations to have bigger impact on local diversity.

Function IM2 is a function with two wide peaks, and one narrow one, reachable by a crossover of solutions from the wide peaks (see Appendix C). It is unlikely that the global optimum is found without performing a crossover, and thus without islands interacting.

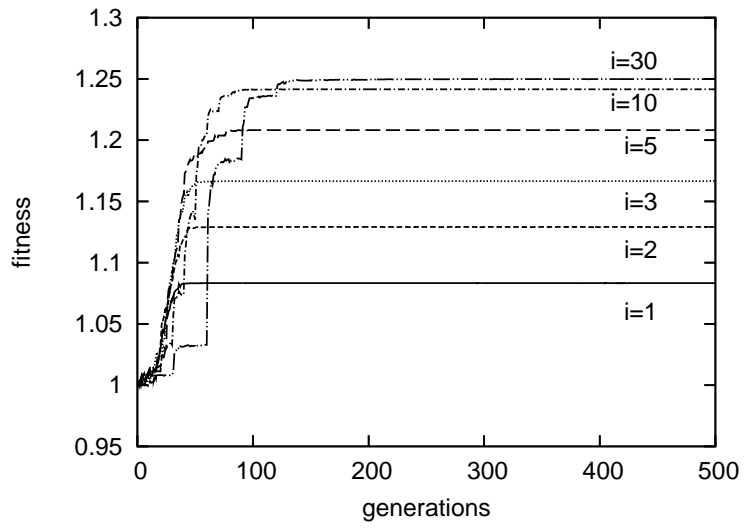
Experiments with the IM2 function show that frequent migration does not yield good results. Best fitness curves in runs for different migration intervals are shown in Fig. 4.20, and in Fig. 4.21, the maximal fitness achieved for various migration intervals is shown — the longer the interval, the better the result (of course assuming some migration *will* eventually occur).

A possible drawback of an interval that is too long is that it may let an island converge to a point with no local diversity. It may be hard for exploration operators (specifically recombination) to increase diversity again because of a smaller number of alleles available. If migrants, however, are significantly different from locals, mixing them can easily overcome the stagnation.

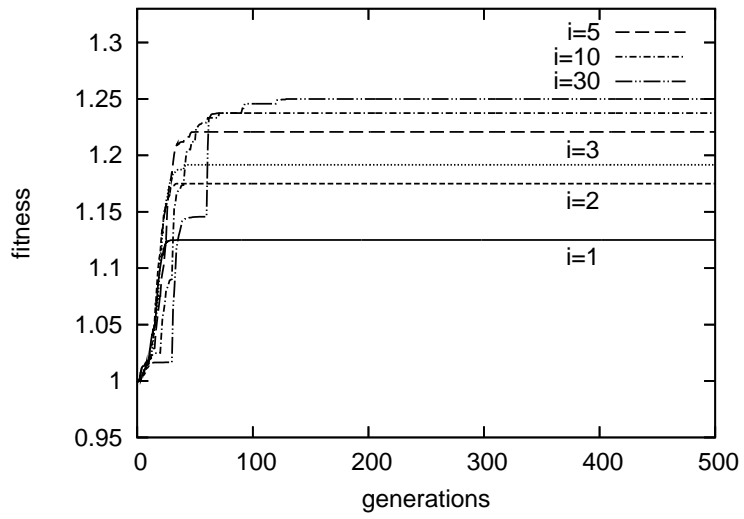
On the other hand, intervals that are too short may not allow individuals to evolve independently. Shorter intervals, even if the migration size is small, make migration act more selectively, because the impact of successive migrations can aggregate easier; this may lead to a convergence toward individuals represented by migrants.

I have tested various migration intervals using the IM6 function. I show the fitness plots in Fig. 4.22–4.23 and the diversity plots in Fig. 4.24. The graphs clearly show that frequent migrations lead to a fast drop in diversity. This result is caused by individuals from one island dominating individuals from another island. If the migration interval is long enough, the global diversity remains large, even though the individual's island diversities drop. This means that different islands converge to different peaks. The exchange of individuals at this point ultimately results in a better mean fitness.

Generally, longer intervals yield better results. One must remember however that with limited resources longer intervals may be too expensive computationally.



(a) EA-1



(b) EA-2

Figure 4.20: System-best fitness curves in runs with different migration intervals  $i$ , the IM2 function.

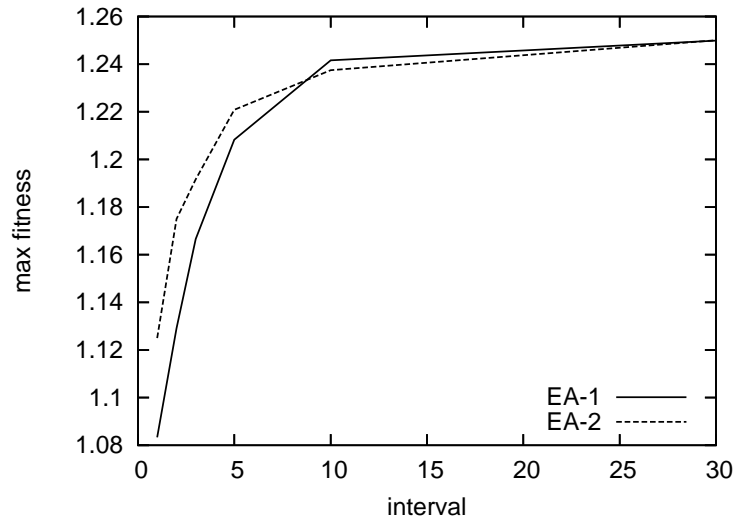


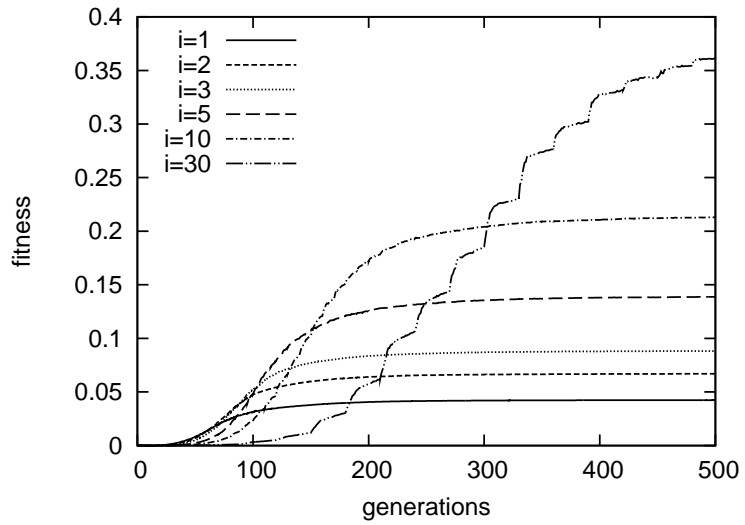
Figure 4.21: Best-in-run fitness values in runs with different migration intervals  $i$ . Larger migration interval helps islands to converge to different optima and recombine later to create the optimal solution, the IM2 function.

Stopping evolution prematurely results in a worse performance.

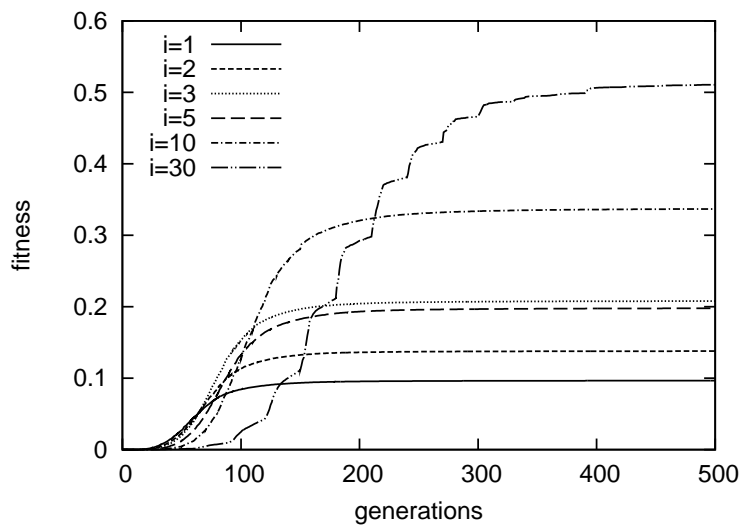
#### 4.4.6 Migration size and interval interaction

Migration size ( $\alpha$ ) and migration interval ( $i$ ) both describe the level of migration, and it is useful to know to what extent they are coupled. Can we approximate the IM's behavior as a function of just the total number of migrants? Or more generally, can we divide the  $\alpha$ - $i$  plane into regions with high, regular and low migration, affecting the performance of an IM? What are the underlying reasons?

One of the questions I wanted to answer was whether setups with different migration intervals and sizes, but with the same total number of migrants (on average per generation), will show the same behavior. This question is especially valid when network resources are limited, and the total number of migrants must not exceed a certain limit.



(a) EA-1



(b) EA-2

Figure 4.22: System-best fitness curves in runs with different migration intervals (i),  $\alpha = 0.1$ , the IM6 function.

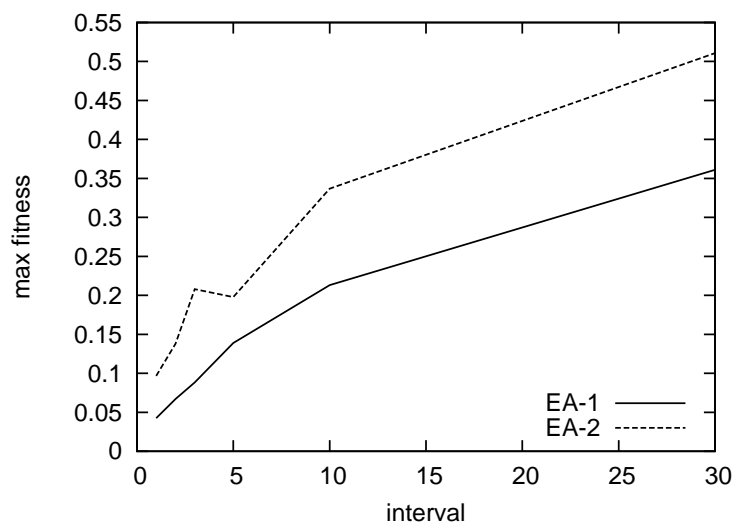
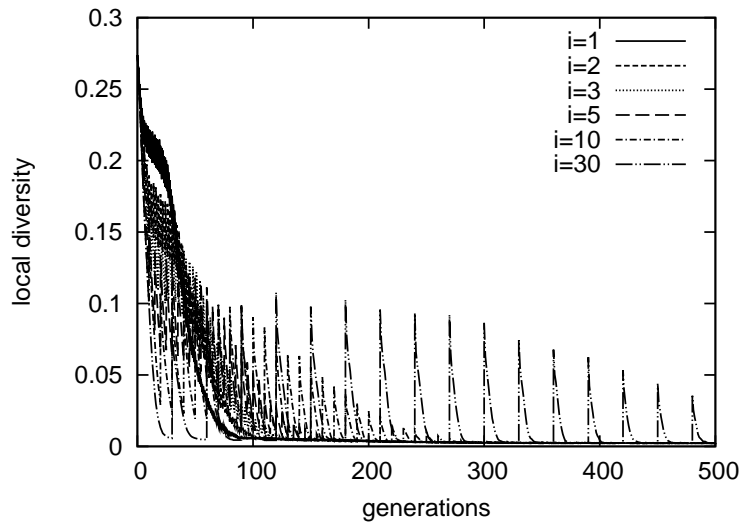


Figure 4.23: Best-in-run fitness values in runs with different migration intervals ( $i$ ),  $\alpha = 0.1$ , the IM6 function.

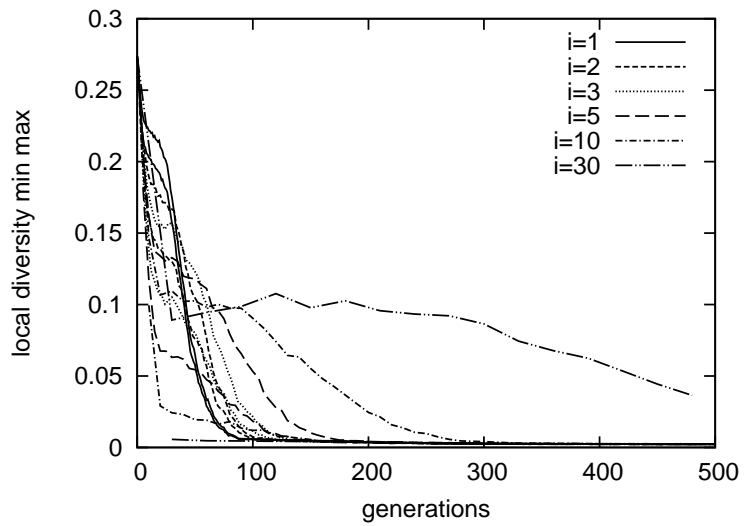
In an earlier paper (Skolicki and De Jong, 2005), we observed that most of these types of setups converged to relatively similar fitnesses, with the exception of the extreme setups. With the IM6 function, I also observed a worse performance for extreme setups, however, the maximal fitness changes for intermediate values as well. In fact, it seems that unless migration size is very large, the positive impact of longer intervals prevails. This is shown in Fig. 4.25–4.26. Studying diversity plots (Fig. 4.27) shows that setups with bigger intervals and sizes observe a faster initial drop in diversity, but need more time to converge completely and are similar to previous charts for various intervals.

#### 4.4.7 Policy

Generally, I have observed a very limited influence of policy on the result of IM behavior. Whereas this situation may be due to the particular settings and landscape

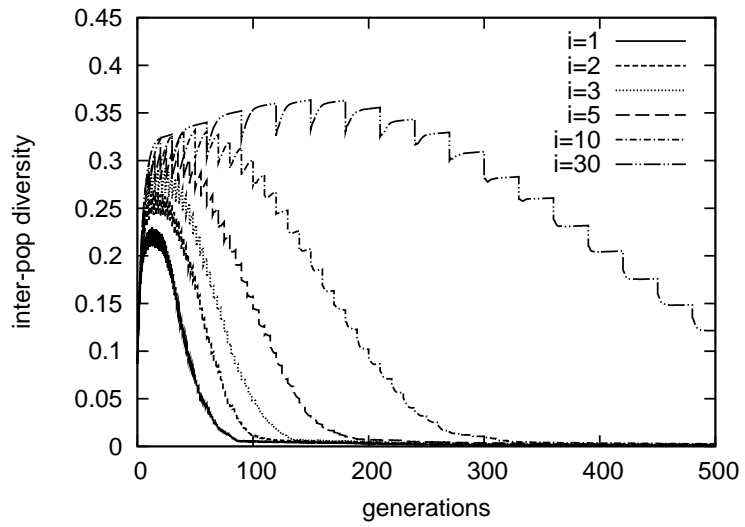


(a) Local diversity (see also the next subfigure)

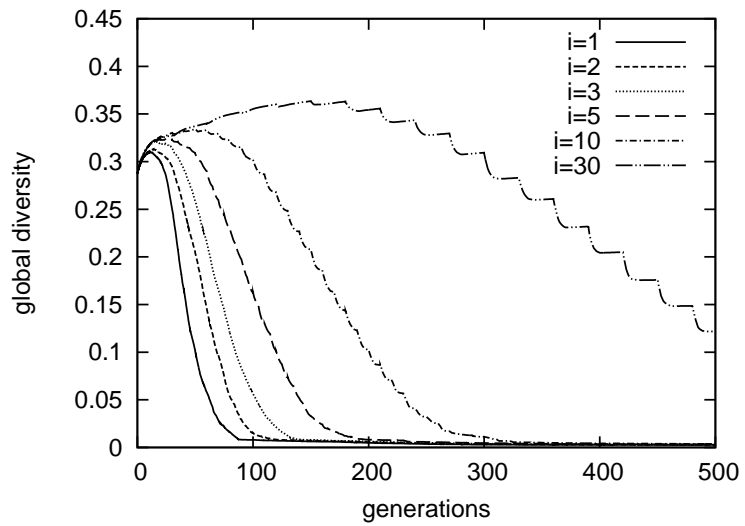


(b) Local diversity (a simplified plot showing boundaries)

Figure 4.24: Diversity in runs with different migration intervals (i),  $\alpha = 0.1$ , the IM6 function.



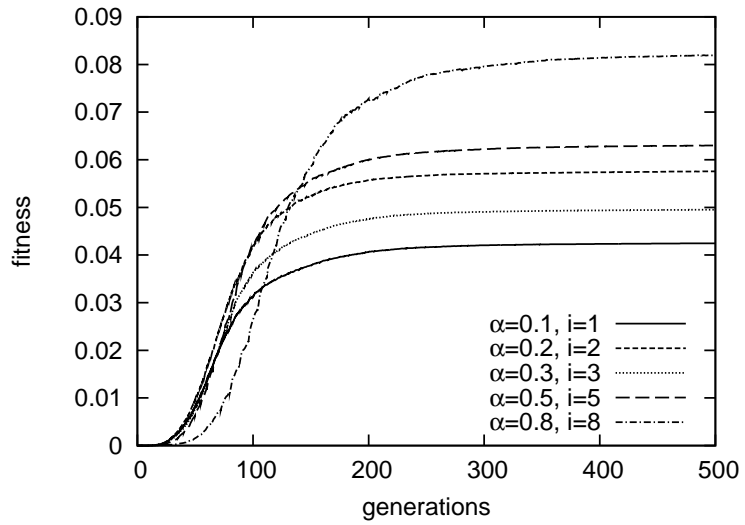
(c) Inter-island diversity



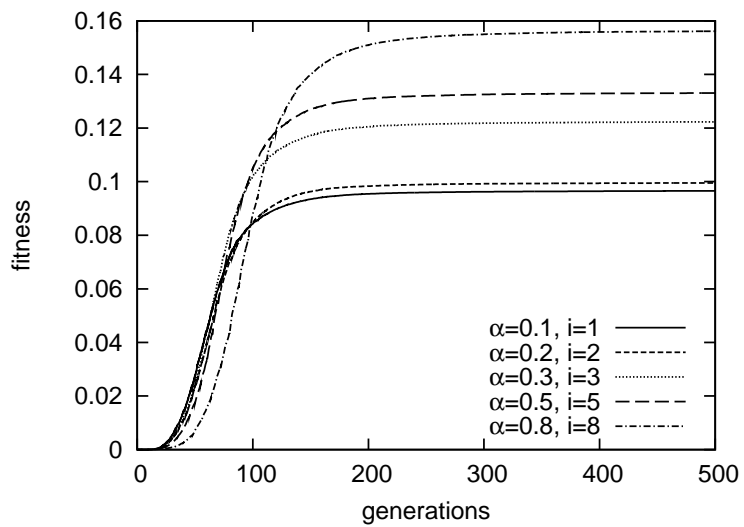
(d) Global diversity

Figure 4.24: Diversity in runs with different migration intervals ( $i$ ),  $\alpha = 0.1$ , the IM6 function.





(a) EA-1



(b) EA-2

Figure 4.25: System-best fitness curves in runs with the same average number of migrants per generation on the IM6 function.

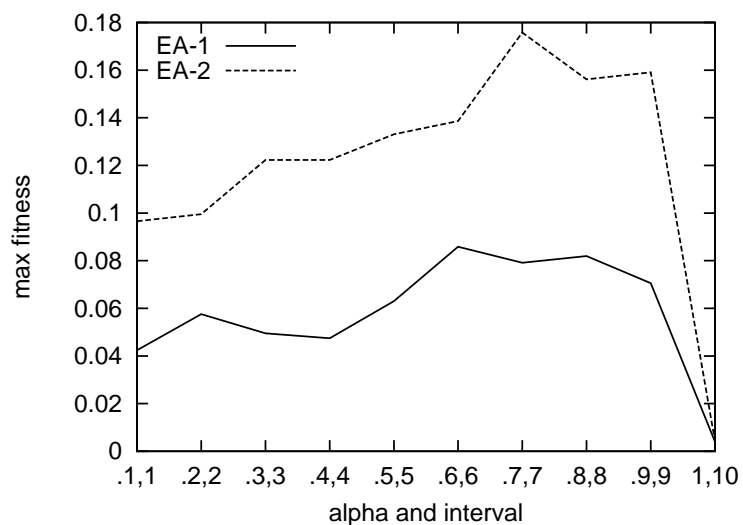


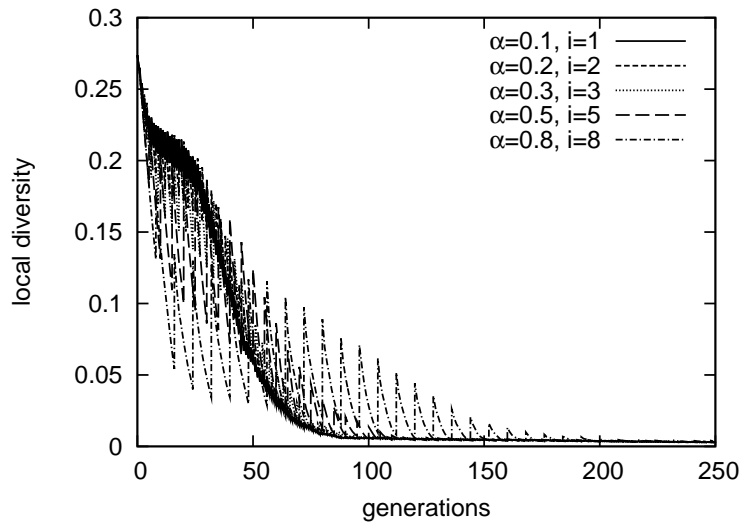
Figure 4.26: Best-in-run fitness values in runs with the same average number of migrants per generation on the IM6 function.

that I used, I think that there are good reasons why policy may simply have a minor influence on the behavior of IMs, especially for setups with many small islands.

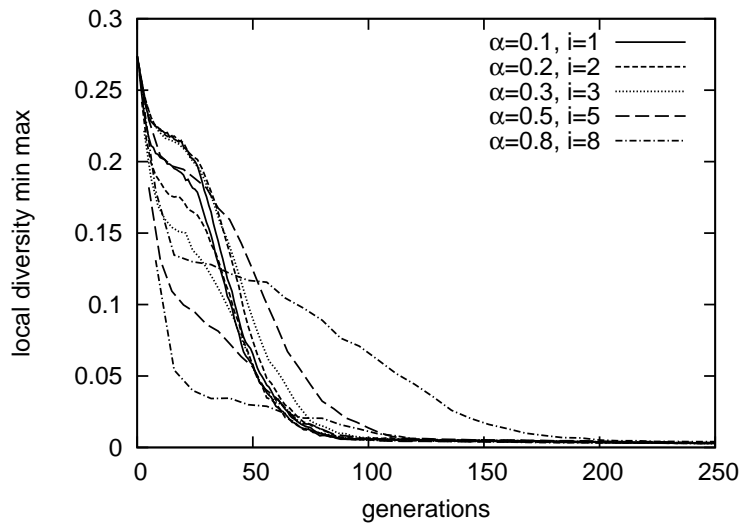
If we migrate individuals after more than a few generations, islands are already to much extent converged. This in turn, means that regardless of what emigration policy we choose, we always select very similar individuals.

In Fig. 4.28, I show exemplary experiments with the IM6 function using random, EA-1 and EA-2 selection schemes. I observe a very small difference in fitness when changing policy settings for medium values of migration size and interval (see Fig. 4.29). Note, that on different charts the range of values on the  $z$  axis changes. Also the diversity plots in Fig. 4.30, suggest that there is little difference in actual dynamics underneath (all the six lines practically overlap, especially for local diversity).

An exception to the little impact of migration policy is the case with very weak selection, frequent migrations and strong migration policy. Let us look at aggregated

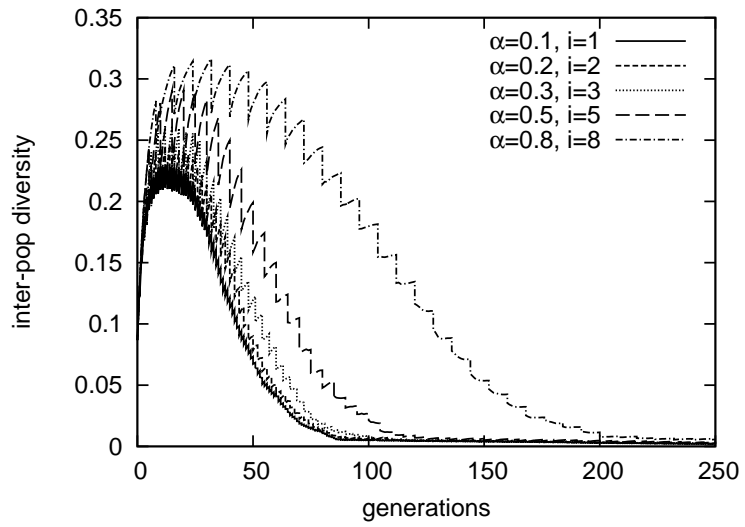


(a) Local diversity (see also the next subfigure)

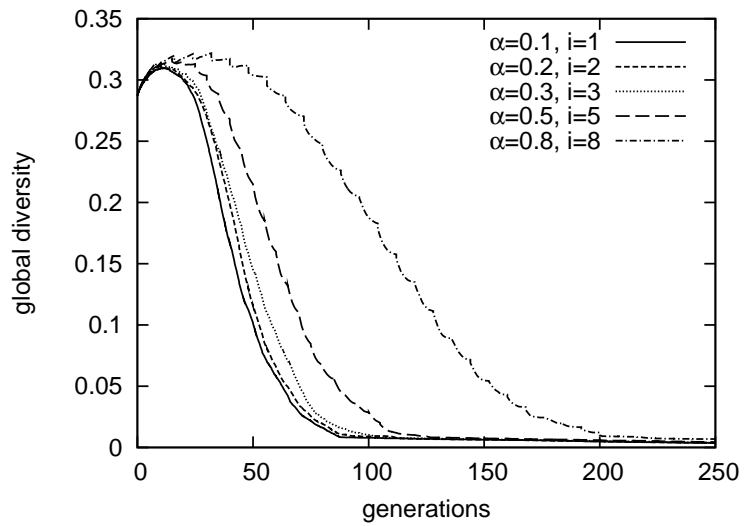


(b) Local diversity (a simplified plot showing boundaries)

Figure 4.27: Diversity in runs with the same average number of migrants per generation on the IM6 function.

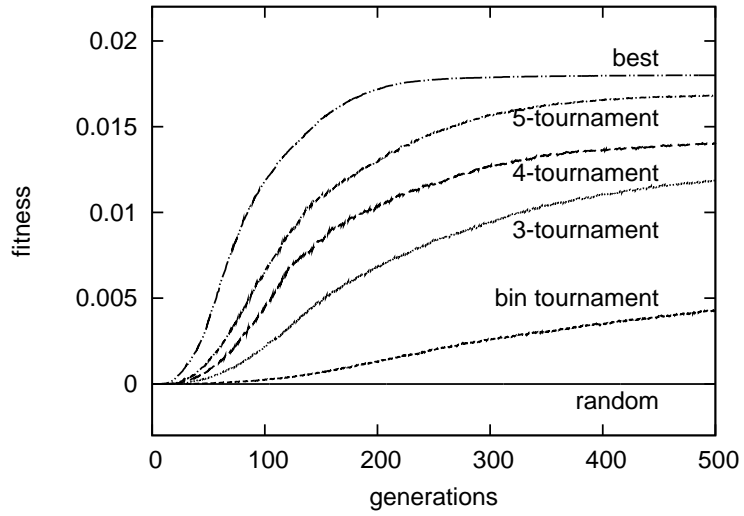


(c) Inter-island diversity

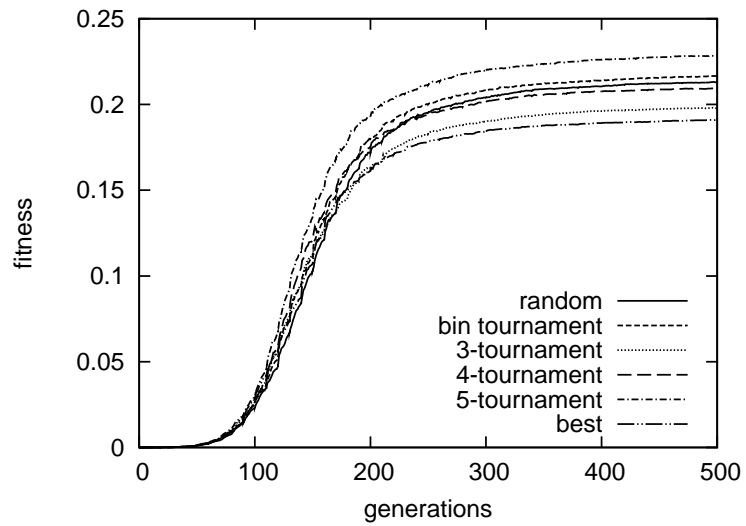


(d) Global diversity

Figure 4.27: Diversity in runs with the same average number of migrants per generation on the IM6 function.

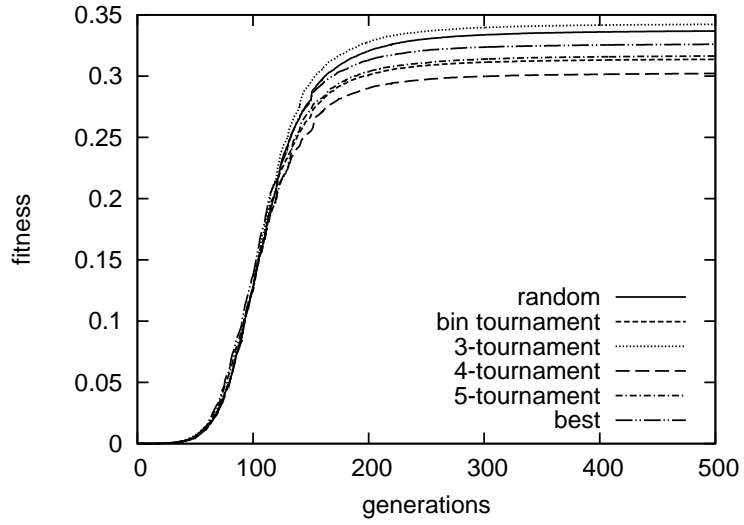


(a) uniform stochastic selection (big and often migrations)



(b) EA-1

Figure 4.28: System-best fitness curves in runs with different migration policy, the IM6 function.



(c) EA-2

Figure 4.28: System-best fitness curves in runs with different migration policy, the IM6 function.

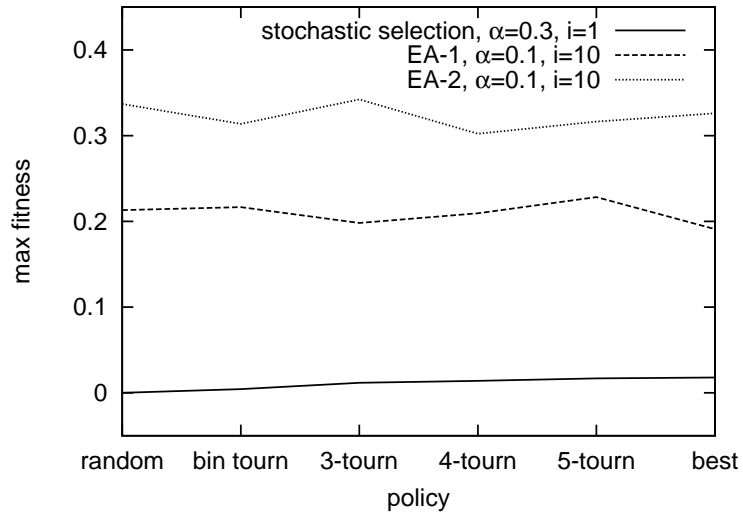
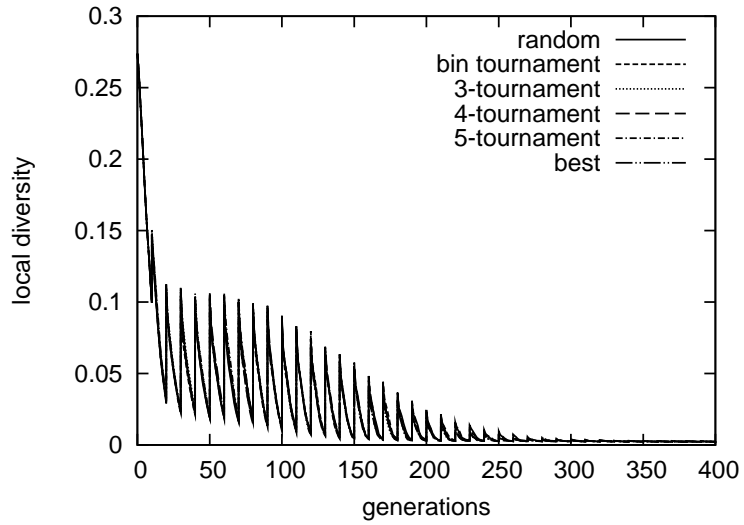
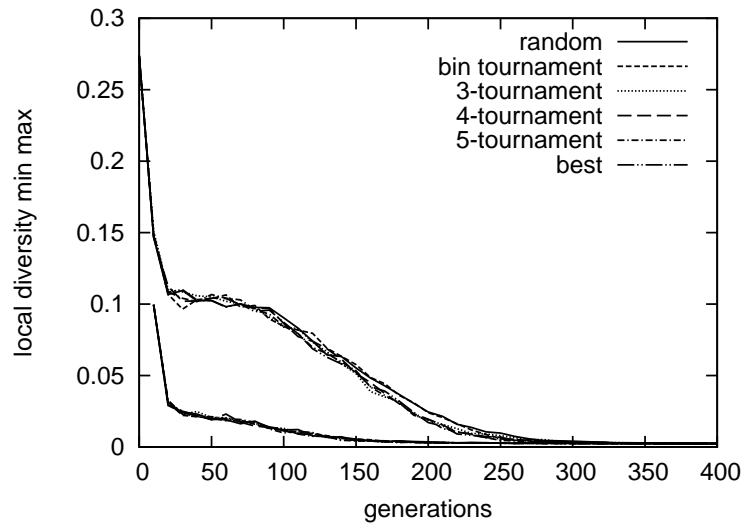


Figure 4.29: Best-in-run fitness values in runs with different migration policy, the IM6 function.

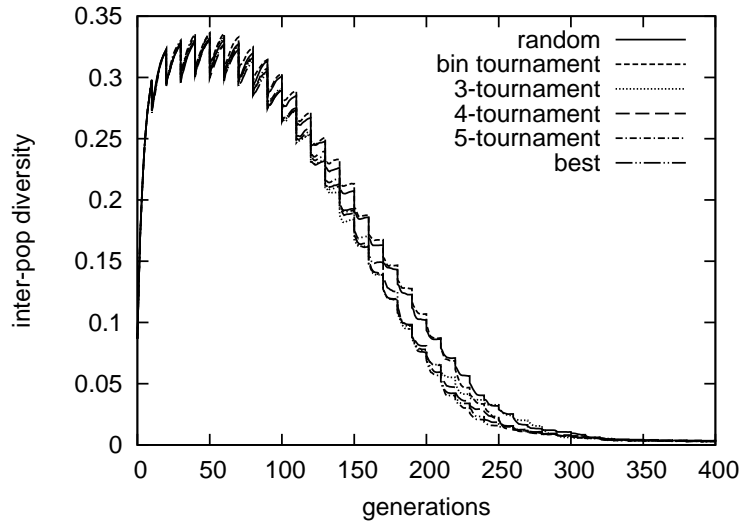


(a) Local diversity (see also the next subfigure)

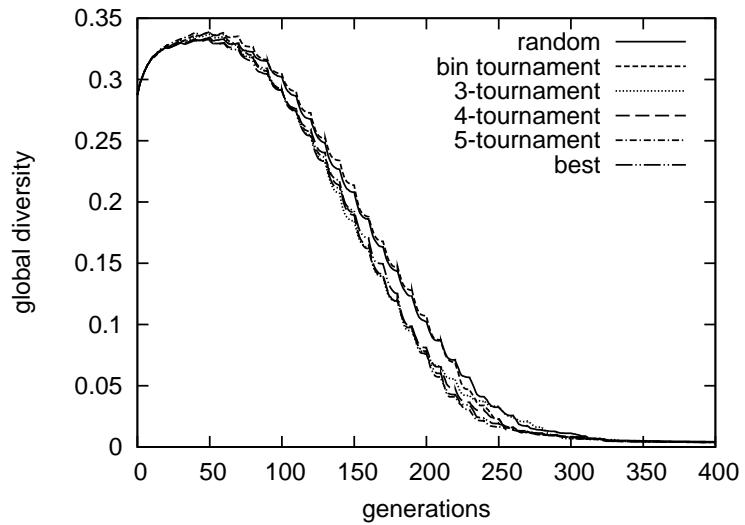


(b) Local diversity (a simplified plot showing boundaries)

Figure 4.30: Diversity in runs with different migration policy, EA-1, the IM6 function.



(c) Inter-island diversity



(d) Global diversity

Figure 4.30: Diversity in runs with different migration policy, EA-1, the IM6 function.

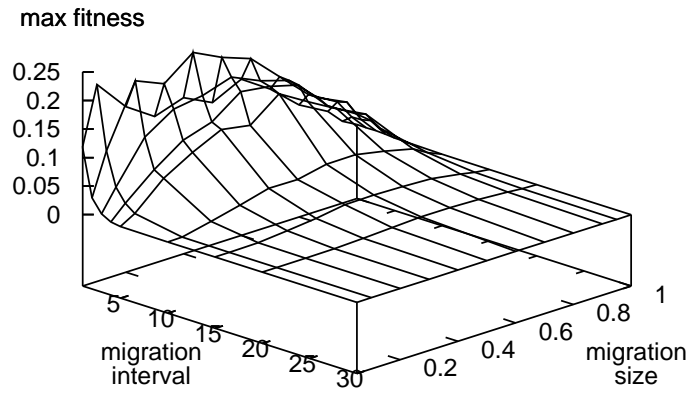


results for the random selection case (Fig. 4.31). We can see that with the increase in the frequency (and for a larger than normal amount of migration), the average fitness reached in corresponding runs also increases. This confirms that for such setups more interaction between islands result in a higher level of fitness. However, in general the worst results with higher selection pressure are comparable to the the best results with random selection pressure.

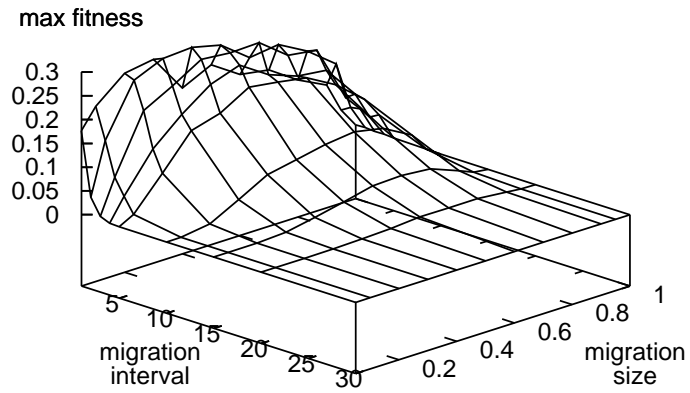
#### 4.4.8 Topology

Topology is not strictly a migration parameter because it does not affect a migration process itself, or it does not characterize a migration between two selected islands. In fact, I believe that the impact of topology is to some degree orthogonal to the impact that other properties of IMs play. Most of the IM dynamics described in this dissertation results solely from the separation of individuals into multiple islands, and from occasional migrations. Even with a fully connected topology, these two aspects are enough to create IM dynamics very different from a standard EA's dynamics. However, topology influences the speed with which individuals are spread in the system. The scarcer the connection is, the more time it takes for an individual to propagate, and the smaller the interaction between islands. Intuitively, with a large grid of islands, the effect of the topology to some degree should resemble the effects observed in a fine-grained distributed EA, in the context of inter-island evolution. While these aspects are outside the scope of my research, for completeness I have performed a limited set of experiments.

To describe the topology with a single parameter, I needed a way to scale it from one extreme (e.g. ring topology) to another (e.g. fully connected topology). This



(a) (dynamic) full topology



(b) (dynamic) ring topology

Figure 4.31: With very weak (a uniform stochastic) selection, strong (best–random) policy together with a high level of migration results in additional selection pressure.  $N = 20$ ,  $M = 20$ .

was achieved in the following way. In each setup all islands were initially connected into a ring. To increase the topology parameter, I decided to gradually increase the number of consecutive neighbors on each side of a given island. If the topology parameter had a value of one, nothing more was done (islands were connected only to the immediate neighbors). If the value was two, additional connections were made to islands two positions away to the left and to the right on the ring. If the value was three, connections were made to all islands up to 3 positions away, and so on (note that for  $N$  islands and topology value of  $N/2$  a fully connected topology is achieved). In this way as the topology parameter grew, the topology gradually changed from a ring to a fully connected topology, and at the same time the diameter of the topology graph quickly diminished. Additional connections created between islands opposite to each other might “short-circuit” the graph and decrease the diameter. However, by renumbering nodes many graphs can be transformed into topologically equivalent ones in which nodes sharing an edge lay much closer to each other, and therefore I did not take these topologies into consideration.

Note, that in this chapter I use a dynamical topology, by which I mean that a graph of connections denotes *potential* migrations, and which ones actually occur is decided randomly in appropriate generation. Each time when migrations are about to occur, for each island only one target island is randomly chosen. There are many good reasons for using dynamical topology as described earlier in this dissertation (see section 3.3.3), even if the method effectively results in somewhat sparser topologies.

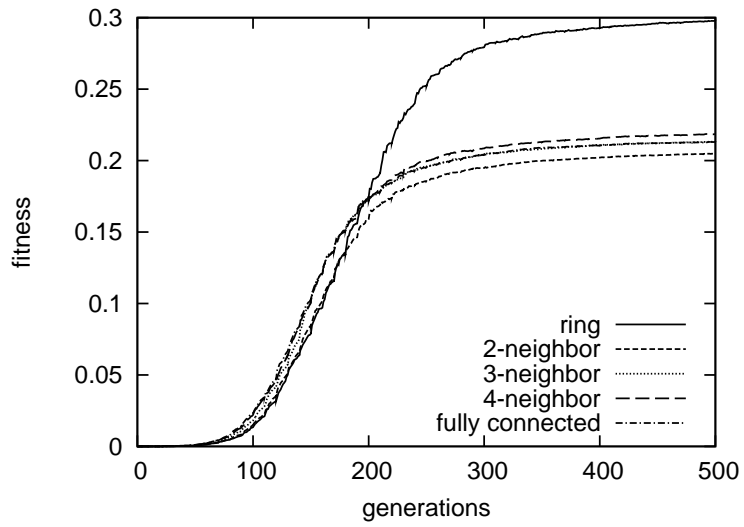
Similar to previous sections, I show analysis with the IM6 function (and observe similar results for other similar functions). The fitness comparison is shown in Fig. 4.32 and 4.33. In the experiments, topology resulted in a minimal impact on the

outcome and most of the time the result was within the statistical error range. With more experiments, the differences would probably become more distinctive, however the fact remains that its effect is less than of other parameters. The exception to this might be results for the very sparse (ring) topology. Possibly, with a larger number of islands, the differences would be more visible.

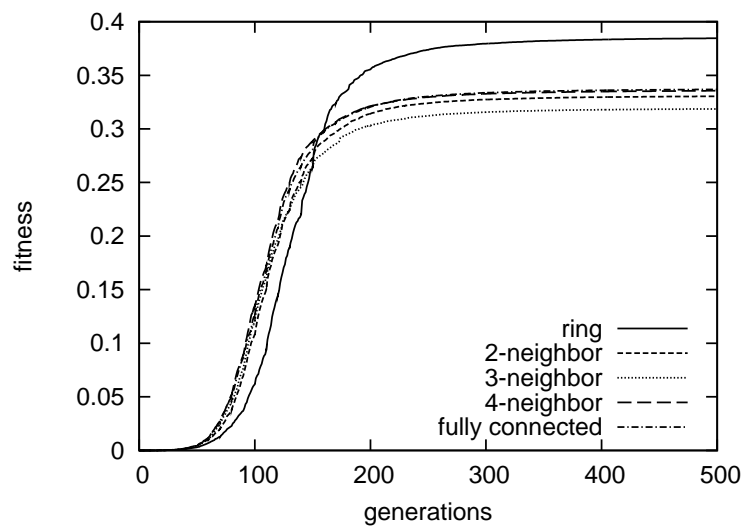
Similar to fitness, studying diversity in the plots with different topology, we again observe little difference. Lines for different configurations again practically overlap. We may notice one subtle thing about local diversity which is visible in Fig. 4.34b: topologies with sparser connections cause the local diversity to drop lower initially, but ultimately to remain higher for a longer time (see the line for ring topology). This can be explained by the fact that migrations between a smaller number of neighbors cause their diversities to decrease faster initially. However, over the longer run, separating islands from different parts of the topology maintains global diversity longer (as confirmed in Fig. 4.34c–4.34d), which in turn causes (re)introduction of unique alleles into islands.

#### **4.4.9 Joint migration parameters analysis**

After analyzing selected migration parameters, I show results for a spectrum of all of them. I will show the results for different selection setups separately. My experiments proved both policy and topology to be of lesser importance, and therefore, I will restrict further trials to two different setups of each, and draw results on single charts. Moreover, migration size and interval seemed to play an important role and therefore, I plot results as a function of these two parameters. The parameters under study are given in Table 4.1.



(a) EA-1



(b) EA-2

Figure 4.32: System-best fitness curves in runs with different migration topology, the IM6 function.

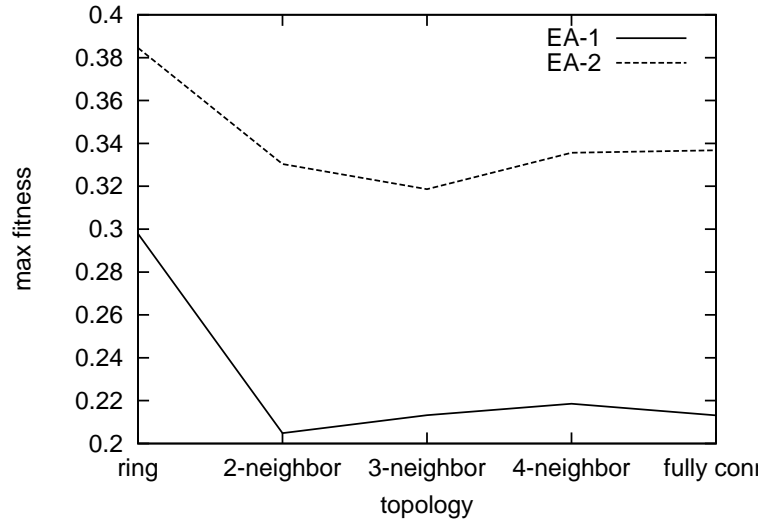
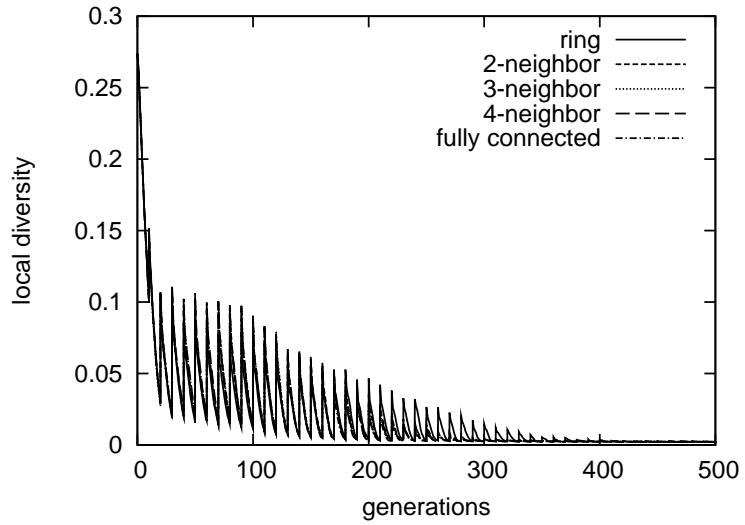


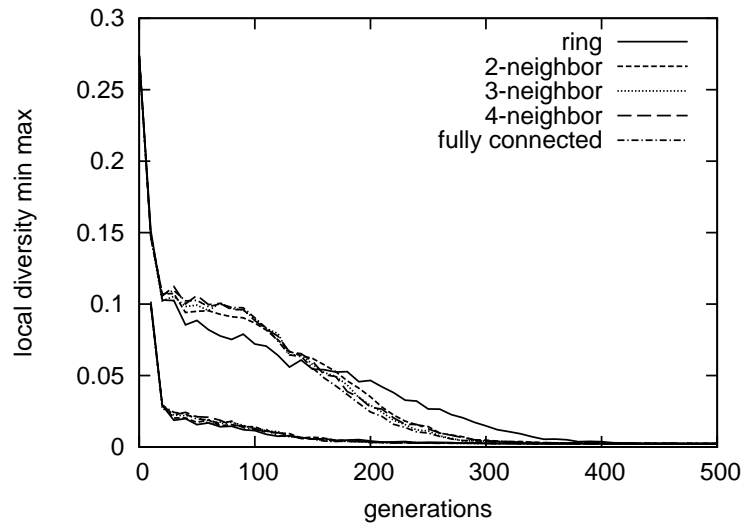
Figure 4.33: Best-in-run fitness values in runs with different migration topology, the IM6 function.

Table 4.1: Parameters and their values.

| <i>parameter</i>              | <i>possible values</i>  |
|-------------------------------|---|
| migration size, $\alpha$      | 0.05, 0.1, 0.2, 0.3, 0.4, 0.5,<br>0.6, 0.7, 0.8, 0.9, 0.95, 1.0 |
| migration interval, $i$       | 1, 2, 3, 5, 10, 15, 20, 30                                      |
| migration policy              | random, best  |
| migration topology            | (dynamic) ring, (dynamic) fully connected                       |
| island EAs selection pressure | EA-1, EA-1 with 5 tournament, EA-2                              |

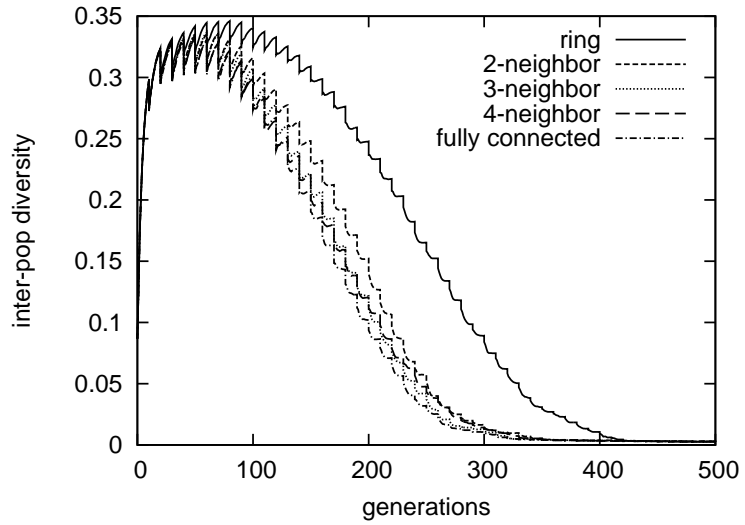


(a) Local diversity (see also the next subfigure)

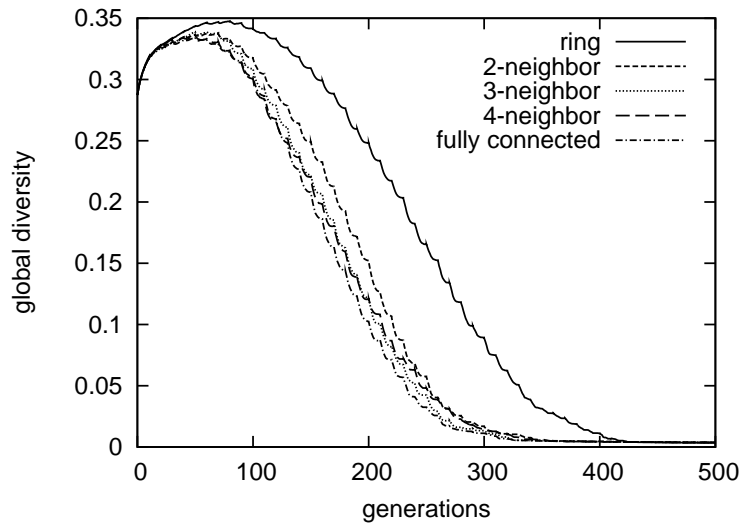


(b) Local diversity (a simplified plot showing boundaries)

Figure 4.34: Diversity in runs with different migration topology, the IM6 function.



(c) Inter-island diversity



(d) Global diversity

Figure 4.34: Diversity in runs with different migration topology, the IM6 function.



I have set the number of islands to 20, and the size of islands also to 20. The experiments used uniform crossover at a rate of 0.7, as a more common setup, and ran for 1000 generations. Other parameters were left as before, and I show results for the IM6 function. Experiments with other functions produced qualitatively similar results. They included the IM3, H-IFF and Deceptive functions, and also standard multimodal functions like Rosenbrock, Schwefel, Rastrigin or Griewank; see also (Skolicki and De Jong, 2005).

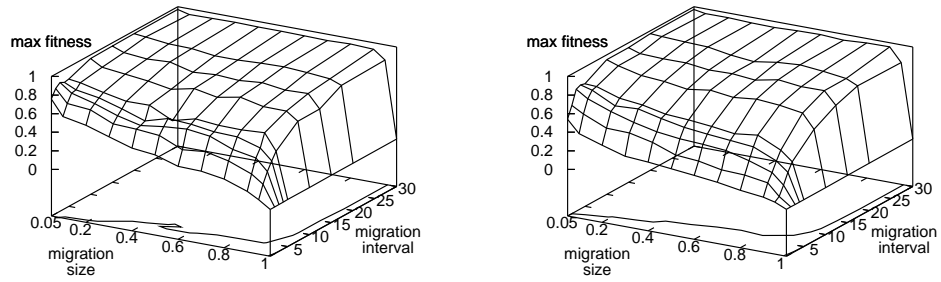
The results are shown in three figures, Fig. 4.35, Fig. 4.36 and Fig. 4.37 and correspond to three different selection schemes (EA-1, EA-1 with 5-tournament selection, EA-2).<sup>6</sup> In each figure, four charts are shown separately and overlaid to stress their similarity. They correspond to different topology and policy settings, namely: random–random policy and (dynamic) ring topology, best–random policy and (dynamic) ring topology, random–random policy and (dynamic) fully connected topology, and, best–random policy and (dynamic) fully connected topology. All the charts are qualitatively similar to each other but, as mentioned before, they differ qualitatively from the case with no (or very weak) selection pressure.

Again, we can observe that the migration interval was playing a much bigger role than the migration size, especially for short intervals, when we could regularly observe a strong decrease in the performance. As already shown, a longer migration interval resulted in a better performance (unless I had to stop the evolution prematurely).

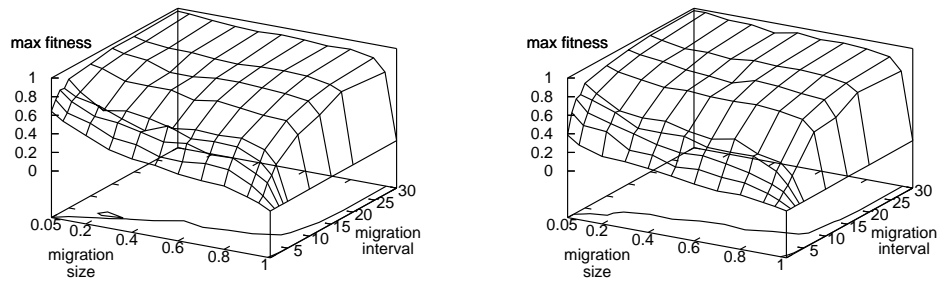
The migration sizes had an inferior impact, but generally smaller migrations resulted in better performance. Knowing that for no migration (isolated islands) the performance is much worse, I expect that for very small migration sizes the

---

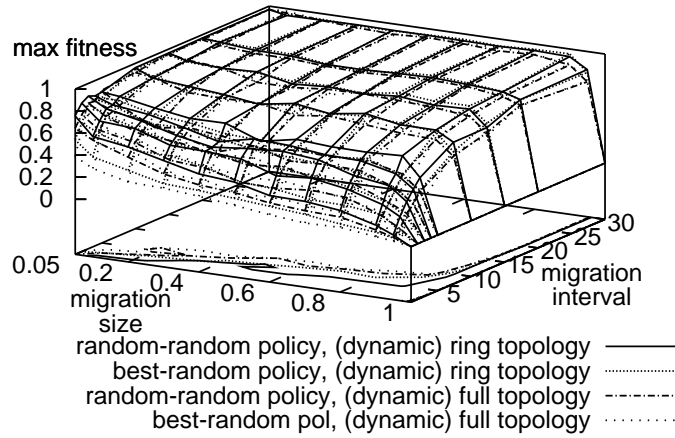
<sup>6</sup>Note that for better visibility migration size axis is not on the left, and migration interval axis on the right.



(a) random-random policy, (dynamic) ring topology (b) best-random policy, (dynamic) ring topology

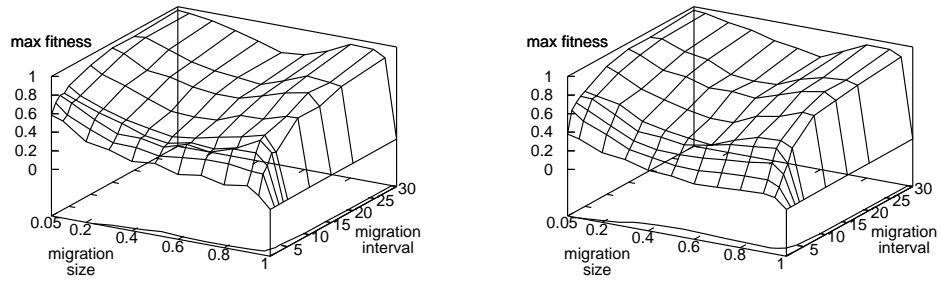


(c) random-random policy, (dynamic) full topology (d) best-random policy, (dynamic) full topology

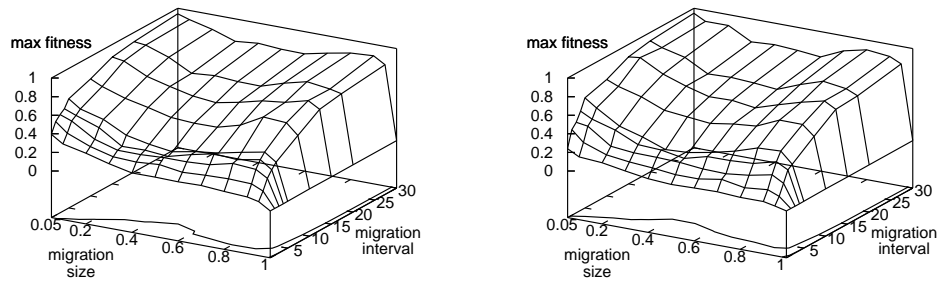


(e) a)-d) setups in one chart

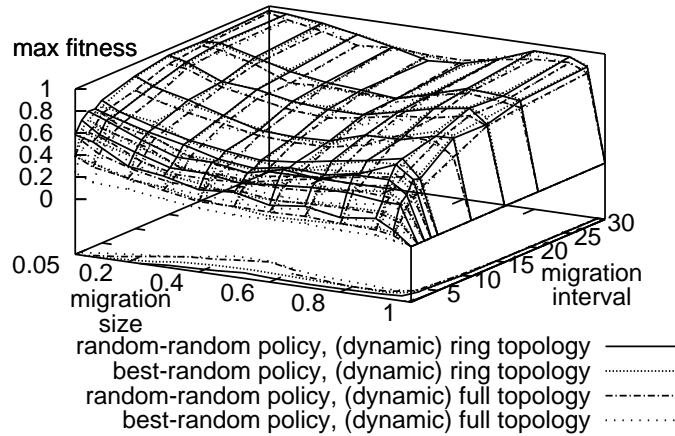
Figure 4.35: Maximal fitness for various parameters, EA-1 (with isolated islands  $\approx 0.019$ , with panmictic  $\approx 0.651$ ).



(a) random-random policy, (dynamic) ring topology      (b) best-random policy, (dynamic) ring topology

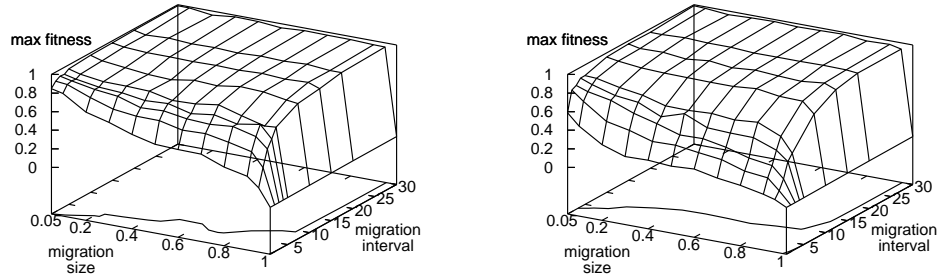


(c) random-random policy, (dynamic) full topology      (d) best-random policy, (dynamic) full topology

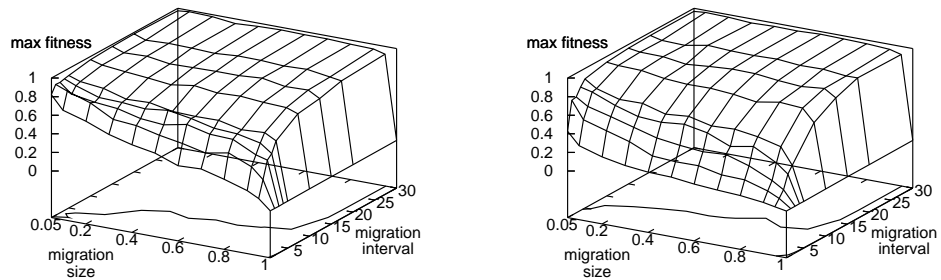


(e) a)-d) setups in one chart

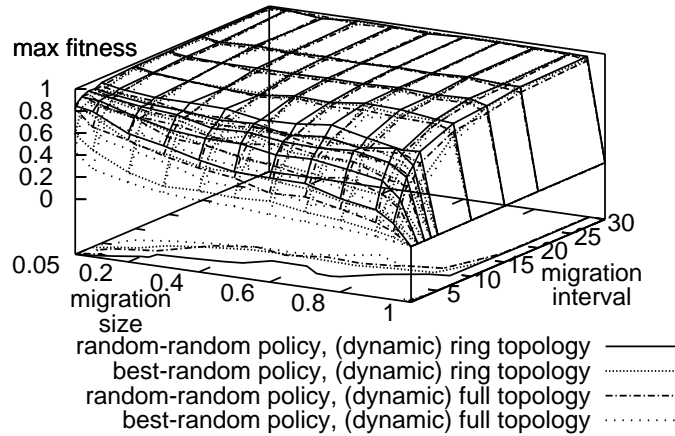
Figure 4.36: Maximal fitness for various parameters, EA-1 with 5-tournament parent selection (with isolated islands  $\approx 0.017$ , with panmictic  $\approx 0.355$ ).



(a) random-random policy, (dynamic) ring topology (b) best-random policy, (dynamic) ring topology



(c) random-random policy, (dynamic) full topology (d) best-random policy, (dynamic) full topology



(e) a)-d) setups in one chart

Figure 4.37: Maximal fitness for various parameters, EA-2 (with isolated islands  $\approx 0.025$ , with panmictic  $\approx 0.851$ ).

performance must be more sensitive to this parameter. Bigger migrations could cause migrants to quickly dominate and “overwrite” the target island, before a successful information exchange and recombination of solutions could occur. For  $\alpha = 1$ , I observe a significant drop in performance. This is not a surprise because target islands are then overwritten with source island populations and no mixing is possible.

I observed (also in previous experiments) that for rare migrations the convergence usually occurred faster with medium migration sizes, rather than with big or small ones. Migration sizes close to island sizes probably make it difficult to mix the genetic material, since they swap the contents of whole islands. On the other hand, with very small migrations it is more probable that the migrant will die without affecting the target island, and thus I observe a longer average time to converge. For 5-tournament selection, I also observe a worse performance with the medium migration size, which is probably related to a faster convergence. All this might be caused by a choice of the 0.7 recombination rate, which results in migrants or locals more easily eliminating others without recombining with each other.

Remembering that an IM is something in between a big, panmictic population, and a set of separated ones, one should compare an IM with those models. In brackets under the charts I show the best-in-run fitness for the panmictic and separated populations cases, and I plot the contour lines for the a panmictic setup on the bases of the charts. For IM6, a panmictic setup always gave better results than an isolated model and the isolated case resulted in values worse than the worst value using IMs.

#### 4.4.10 Increase of evolvability

In this section I discuss changes in island dynamics *following* migrations. We saw that IMs give better results if the two levels of evolution were separated (low migration size and interval). I will analyze a little deeper two special cases, one for large and often ( $\alpha = 0.2, i = 1$ ), and one for small and rare ( $\alpha = 0.02, i = 10$ ) migrations. I will focus on the dynamics of the second one and show that even though I migrate a lot fewer individuals in the second case, migrations have greater impact. One may think that increasing the number of migrants for the frequent migration might help, but this is not true. At the same time, even a very small number of migrants under rare migration significantly changes the dynamics of the system.

Certain settings allow to better see the impact of migrations on evolvability. I show experiments with a strong parent selection (EA-2 with 5-tournament). For binary parent selection (EA-1), this behavior, although existing, is difficult to notice due to a weak impact. For the EA-2 setup, the behavior is also difficult to observe, most likely due to overlapping generations. I also show results for binary tournament emigration policy, rather than random (although it had little influence). To even better visualize the dynamics inside islands after migrations, I use a setup with bigger islands, namely 20x50 IMs. Topology was set to a dynamic fully connected one. I will again analyze the IM6 function. In Chapter 8, I show corresponding results for the H-IFF and Deceptive functions.

Let us look at the fitness charts in Fig. 4.38. We can observe something interesting with infrequent migrations, namely *after* every migration there is an improvement in fitness, and even though the total amount of migrants is much smaller, the overall improvement turns out to be bigger. Even with very small migrations, the

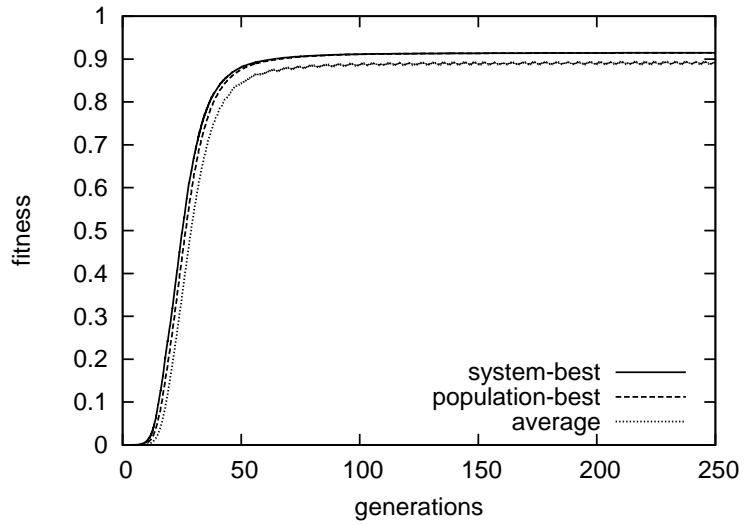
effect remains practically the same. Let us look at this phenomena using different measurements to better understand its dynamics.

A measurement related to fitness is selection intensity, which measures relative improvement of average fitness with relation to the fitness diversity. In a typical run of an EA selection intensity is positive in the beginning and then slowly drops to zero, as the population converges and no further improvement in fitness is observed.

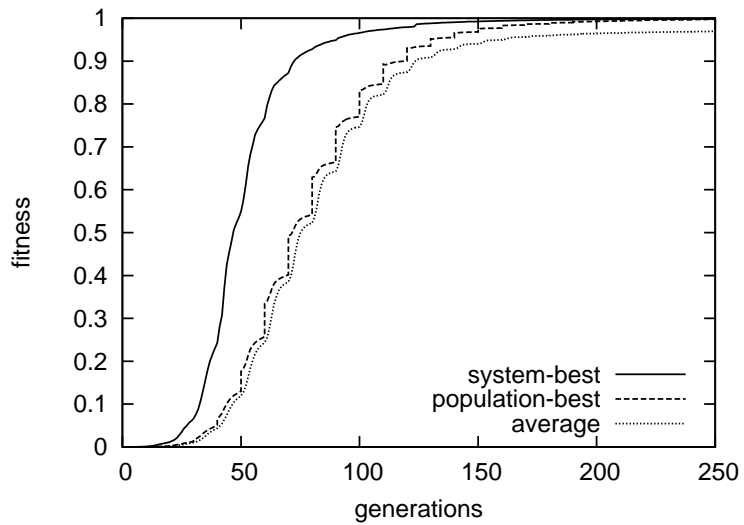
With island models I actually plot two lines (Fig. 4.39). One of them represents selection intensity of the migration itself, and so I connect points corresponding to the change in fitness before and after migrations. The other line corresponds to evolution between migrations, so I connect points corresponding to all fitness changes, except for the changes resulting from migration.

The charts for the two cases differ a lot. Let us first analyze the case of the big and frequent migrations. We see that after some time, they also stabilize, but not exactly at zero. Selection intensity from migration remains positive and selection intensity from evolution becomes negative. The explanation is probably the following. Because good individuals are migrated from other islands, they improve the average fitness of the target island. Hence, the migration line remains always positive. Evolution produces a negative selection intensity. Apparently, migrants either do not mix well with others and get eliminated immediately after migrations, or the average offspring fitness is low and thus the reported fitness change is negative. Before any improvement is possible, the next migration occurs. So, in the case of big and frequent migrations, migrations are responsible for improvement in the islands, carrying good solutions with them, and short periods of evolution between migrations has negative effect.

A notable difference to the scheme described above is for infrequent migrations



(a)  $\alpha=0.2, i=1$



(b)  $\alpha=0.02, i=10$

Figure 4.38: Fitness compared. Small migrations boost evolution.

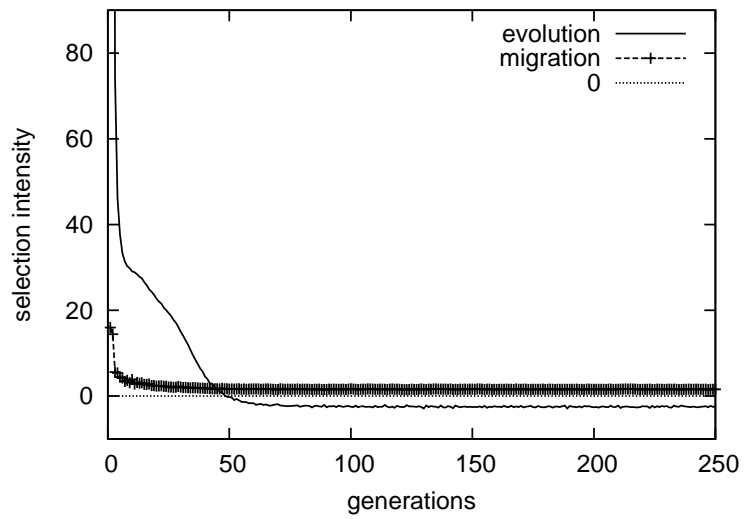


with strong selection on islands. After each migration, there is a boost in the evolution line. This probably means that migrants not only increase their number in the target island, but are successfully recombined with other individuals and for some period keep the island evolving. This seems to be the reason also for ultimately achieving a higher fitness. In the case of small and infrequent migrations, they are too small to cause a significant immediate change in the average fitness, and in fact the migration line remains very close to zero. All the improvement comes from evolution, which would, however, not happen without those migrations.

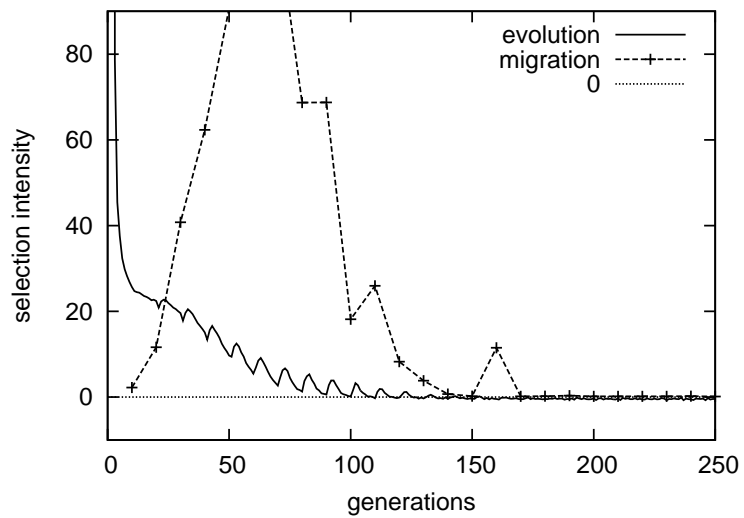
As expected, the increased ability to evolve is also seen in diversity plots (Fig. 4.40). The largest increase in diversity is observed following infrequent migrations. Since, in general, an increase in global diversity occurs when an island population moves from one peak to another, this movement is what may be happening underneath.

In Fig. 4.41, I show the results of tracking particular alleles after migrations. Note that for a clarity of the pictures, for migrations that occur every generation I plot every tenth migration. We see that in the case of big and frequent migrations, new alleles are quickly removed from the target islands, as the new immigrants come. The percentage of them never grows except for the initial injection of alleles from the migration itself. With less frequent migrations, and especially with the small infrequent migrations, the situation is very different. The initial percentage  $\alpha$  is nearly ten times smaller than the maximum! It is the generations following the migration when the percentage of appropriate genes increases to the levels comparable with the first case.

In earlier experiments with infrequent migrations we observed that the percentage

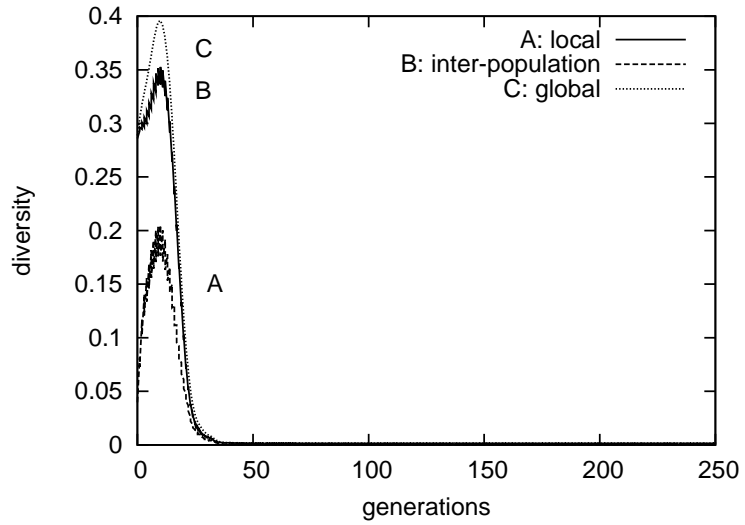


(a)  $\alpha=0.2, i=1$

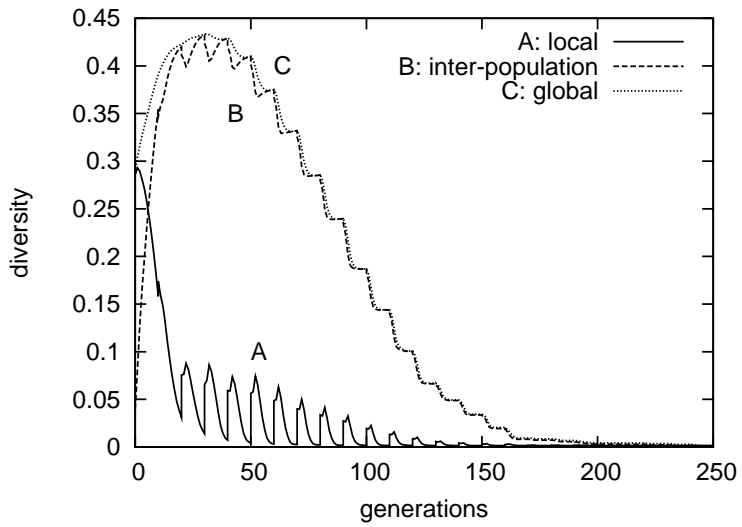


(b)  $\alpha=0.02, i=10$

Figure 4.39: Selection intensity compared. Small migrations boost evolution.



(a)  $\alpha=0.2, i=1$



(b)  $\alpha=0.02, i=10$

Figure 4.40: Diversity compared. Small migrations boost evolution.

of some migrations' alleles stabilized above zero, after some number of generations. The final solutions consisted of alleles coming from several islands. This confirms that it is a successful recombination of individuals, rather than domination of migrants from a certain island, that produces the highest fitness, at least for the functions studied in this dissertation.

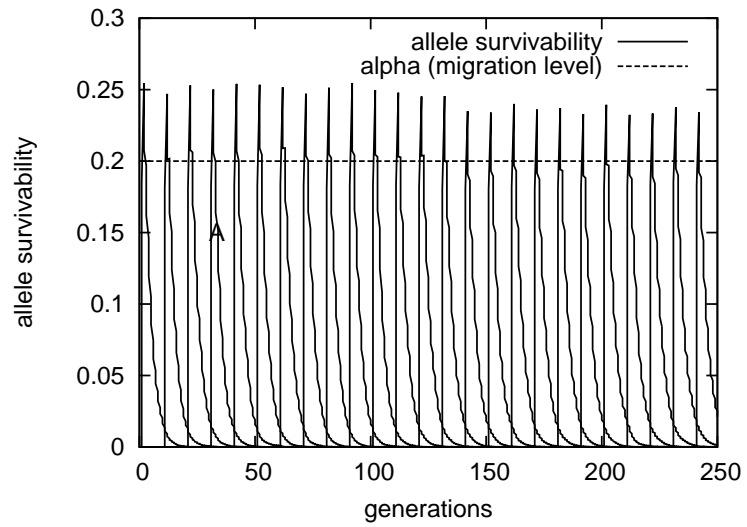
## 4.5 Pitfalls

In this section I show examples of problems that are difficult for IMs. I show how their failure can be understood in the context of the two-level evolution. The reasons for these failures seem to be deep, and increasing the total computational budget does not help. This is because islands may “block each other” either by mixing incompatible solutions into a deceptive one, or by converging to the same peak and losing the inter-island diversity.

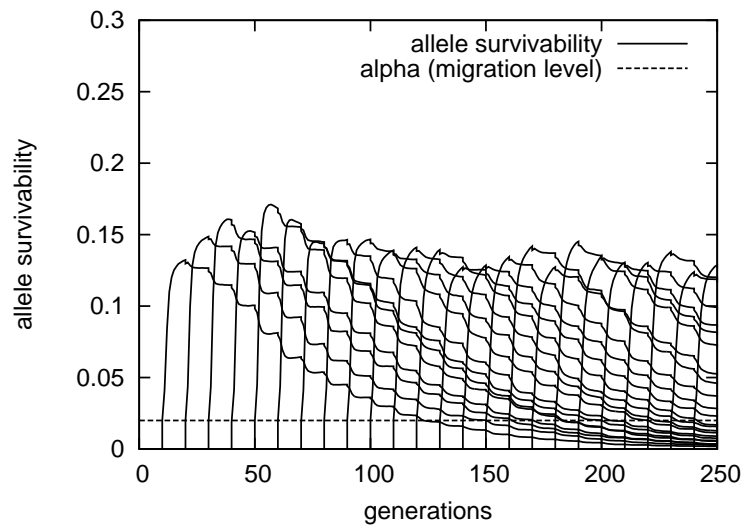
### 4.5.1 Deception at the inter-island level

One possibility for an IM failure is a function or problem that has various “paths” to reach the global solution, but mixing solutions from different paths would create a stable, sub-optimal solution (a “trap”). With bigger  $N$ , the probability that islands choose different paths is higher, and therefore migration occurring between the islands may cause all the islands to become trapped.

Let us construct the following IM-Trap function. The function operates on binary genomes and returns the number of consecutive zeros starting from the left or the number of consecutive ones starting from the right, whichever is greater. Additionally, if there is simultaneously a block of zeros from the left, and a block of ones from the



(a)  $\alpha=0.2, i=1$



(b)  $\alpha=0.02, i=10$

Figure 4.41: Allele survivability compared. Small migrations boost evolution.

right, then the function falls into a “trap” and returns a fixed, high but sub-optimal value. More formally for a genome  $a = (a_1 \dots a_L)$

$$\text{left}(a) = \max_{i=0\dots L} \{i : \forall 1 \leq j \leq i \quad a_j = 0\}$$

$$\text{right}(a) = \max_{i=0\dots L} \{i : \forall 0 \leq j \leq i - 1 \quad a_{L-i+j} = 0\}$$

$$\text{IM-Trap}(a) = \begin{cases} 0.9 \cdot L & \text{if } \text{left}(a) > \frac{L}{4} \text{ and } \text{right}(a) > \frac{L}{4} \\ \max\{\text{left}(a), \text{right}(a)\} & \text{otherwise.} \end{cases}$$

If the migration level is very low, then there is a chance that a single population will discover the global optimum before islands get trapped due to the exchange of individuals. If the migration level is very high, islands may exchange solutions before they discover the BBs of the trap — and also find the global optimum, all by following the same path. If the migration level is moderate, however, BBs for the trap will be found on different islands, and then mix together to create the trap.

In Fig. 4.42, I show the results of simulation for a different  $N$ , using EA-2 this time.<sup>7</sup> Simulations with EA-1 and the same other parameters do not get trapped because the islands do not converge to the trap as fast. To show the effect with EA-2, some tweaking in the experiment would be necessary. We clearly see that more islands results in a worse performance (even though I do not change the size of the islands, so more computational resources are used). With a larger  $N$ , the focus is shifted toward inter-island evolution which is the reason for the poor performance. The slight growth

---

<sup>7</sup> $\alpha = 0.1$ ,  $i = 50$ , policy = random, topology = full, parent selection = stochastic, recombination = one-point crossover,  $M = 50$ ,  $L = 100$ , mutation rate =  $1/L = 0.01$ , survival selection = truncation, overlapping, IM-Trap.

of the maximal fitness for a large  $N$  may result from the fact that I use a dynamic topology. Each island sends migrants to only one target island; with more islands it takes longer for a given solution to spread in the system, therefore, giving islands more time to reach the global optimum on their own. A better performance may, however, also result simply from the fact that for a bigger  $N$ , there are correspondingly more individuals that exist in the system.

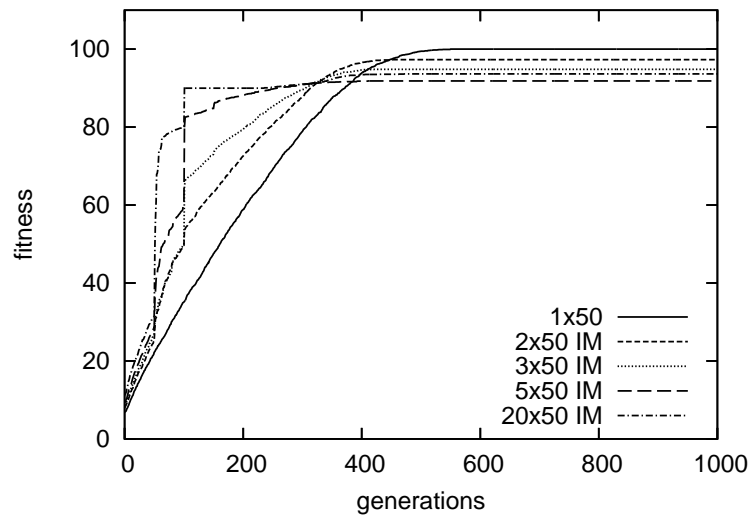
### 4.5.2 Losing inter-island diversity

Surprisingly, another potential pitfall is large islands, which may be disastrous to inter-island diversity. This may happen for cases where randomness and different evolution paths for islands play a major role (like in compositional evolution!).

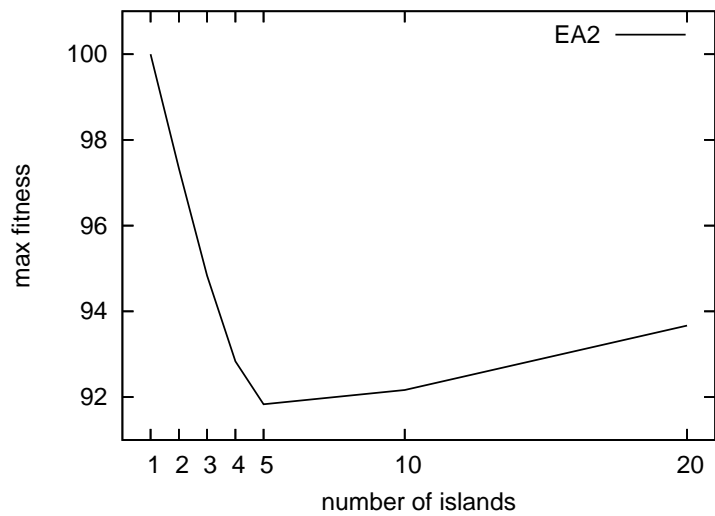
Let us take a function  $IM2'$ , similar to  $IM2$ , but in which one of the wide peaks is higher than another (see Appendix C). The  $IM2'$  function is defined over  $[0, 1]^2$  by the following formula:

$$IM2'(x, y) = \begin{cases} 3 & \text{if } x, y > 1 - \frac{1}{10^3}; \\ \max(x^2(y-1)^2, 2(x-1)^2y^2) & \text{otherwise.} \end{cases}$$

It is easy to predict that the global optimum will not likely be found by random sampling, and the higher of the wide peaks will be chosen more often by an EA. This tendency becomes stronger with the growing size of a population, which in turn, means that larger islands will more likely converge to such a peak. In particular a weak selection, together with islands big enough to oppose drift, may no longer converge to sub-optimal solutions that would be beneficial for combining later. As a result, if similar BBs are created, recombination of them after migration is ineffective.



(a) system-best, EA-2



(b) best-in-run, EA-2

Figure 4.42: Fitness for different number of islands ( $N$ ), the IM-Trap function.



To maintain diversity, islands must converge to different optima. If the best solution can be achieved only as a recombination of different sub-optima, then a system with big islands will fail.

Figure 4.43 shows this effect.<sup>8</sup> Note, that with a weaker selection (EA-1), every island size results in a suboptimal solution, because the algorithm is more explorative and finds the higher of the two wide peaks more often. Even if the best solution is found (as seen by peaks appearing just after migrations), it does not survive in the system due to the lack of elitism. The final fitness as a function of island size is shown in Fig. 4.44. It may be counter-intuitive to realize that this function is not monotonic.

## 4.6 Summary

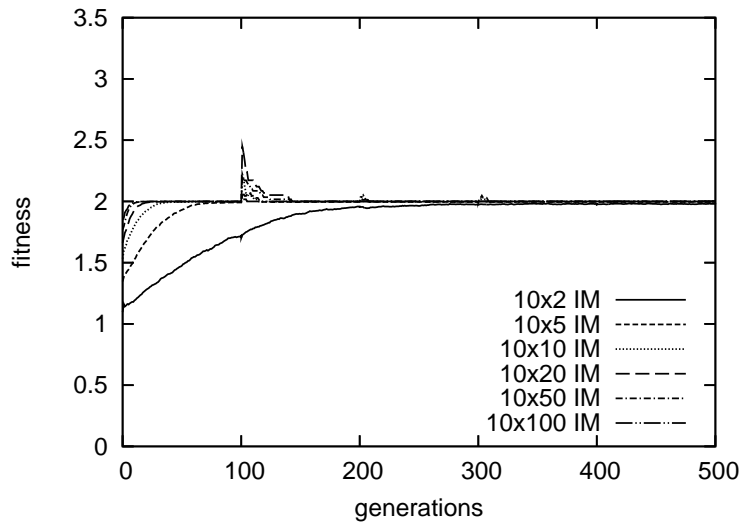
Observations from this chapter lead to a certain understanding of distributing individuals into islands, as well as the migration role in IMs.

- Island models to some extent behave like two-level evolutionary algorithms. Local evolution takes place inside islands, and global evolution takes place between islands (how this happens exactly I will try to explain in the next chapters). This two-step evolution has positive influence, since it enables genes to be optimized more evenly, and enables the creation of various partial solutions.
- More islands enable more exploration at the inter-island level of evolution.

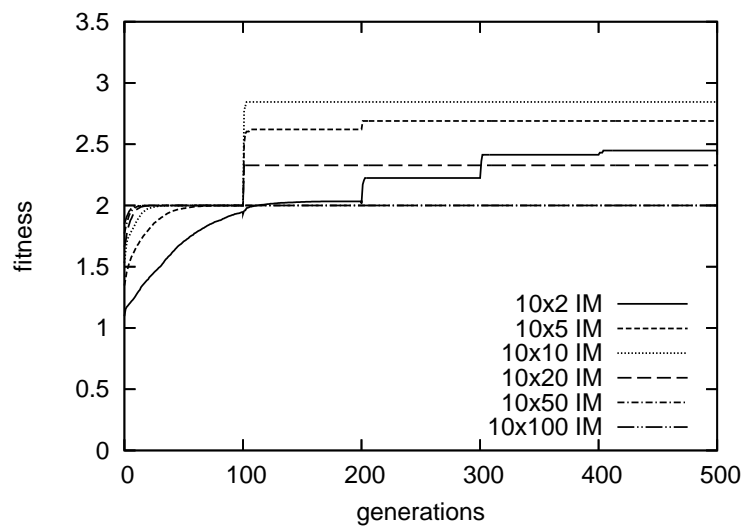
Moreover, with more islands, local diversities drop more slowly, which is

---

<sup>8</sup> $\alpha = 0.5$ ,  $i = 100$ , policy = random, topology = full, recombination = uniform crossover,  $N = 10$ ,  $L = 2$ , mutation rate =  $1/l = 0.5$ , IM2'.



(a) system-best, EA-1



(b) system-best, EA-2

Figure 4.43: System-best fitness curves in runs with different island sizes,  $M$ , the IM2' function.

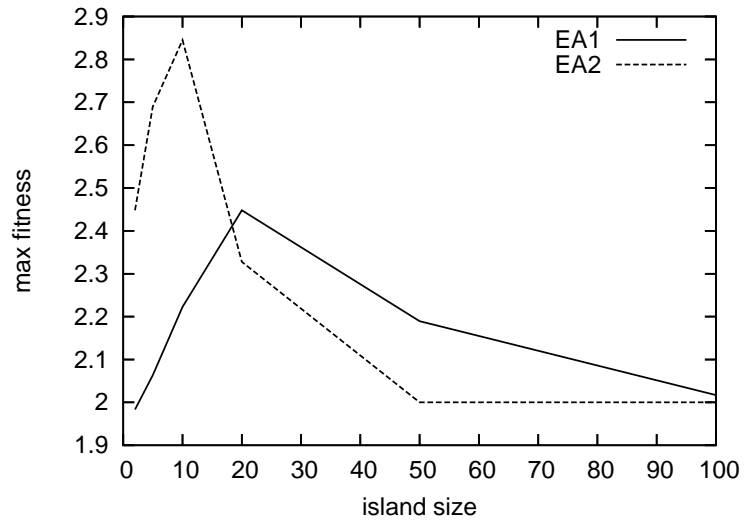


Figure 4.44: Best-in-run fitness values in runs with different island sizes, the IM2' function. Too big islands prevent islands from converging to different optima to recombine and create the optimal solution.

beneficial. However, deceptive cases exist when islands may block each other in evolution.

- Islands must maintain reasonable sizes to maintain evolvability for some time before convergence. However, because migrations increase islands' diversity anyway, fixation can play an important role in diverging islands evolutionary paths and retesting different fixations can ultimately find good building blocks.
- The number of islands, and their size must be well balanced. Stronger selection seems to require a larger number of smaller islands.
- Migration parameters, although related to each other, each have a different significance as explained in the remaining bullets.

- The strength of each island’s selection pressure has a strong influence on IMs behavior and migration parameters often must be adjusted to it. With stronger selection, results generally get better, although additional experiments (not reported) show that too high of a selection decreases the performance again (most likely due to a convergence that is too fast).
- Depending on the selection pressure on the islands, migrations can play various “roles” in achieving good results by the algorithm:
  - When the selection pressure is uniform or very weak, migrations can serve as an additional selection mechanism by migrating good individuals often and in large quantities. Such migrations (short intervals, big sizes and strong policy) effectively increase the average best-in-run value. This is in agreement with Cantú-Paz studies.
  - When the selection pressure is medium to strong, migrations can serve as a method for exchanging genetic material between islands, so that it can be “picked up” in target islands. Such a migration would propagate genetic information but not flood the target island. The impact of the migration size and interval parameters on the best-in-run fitness seem reversed in this case, and the migration policy seem to play a minor role. Even small migrations can significantly contribute to the target island because evolution may amplify their influence by quickly increasing the percentage of their genes in the island. If this happens, diversity in the island grows, selection intensity is increased and the final results obtained are the best.

What is worth noting is that this is a setup that a practitioner would normally use.

- The choice of migration policy seems to be secondary, with the exception of weak selection with large and frequent migrations, when a strong migration policy is desired. The observed weak influence of migration policy may be caused by the fact that with strong selection islands converge, and so the choice of migrants is secondary. Firstly, all genomes are similar in this case, and secondly, what is best in a current generation can be average in the following generations, so intuitively migrating the best individuals can be equal to migrating random individuals just one generation later.
- Surprisingly, the topology parameter also had little impact. Possibly, I had too few islands, and experiments with hundreds islands would be needed to observe a stronger influence of topology. It is also possible that due to using dynamic topologies, I made them all sparse and changes would rather occur by making topologies denser.

Out of the two described possible roles of migrations, the second produced better results and used standard parameters of evolutionary algorithms on islands (as opposed to e.g. very weak selection). It is more easily distributed because the interaction between the islands is smaller. Evolutionary processes on islands are more independent and the new individuals are effectively mixed into the old solutions. This should be useful for hierarchical problems, when there is no gain from a better individual dominating a new island, and rather mixing its genetic material with one existing in the target island, produces better results.

In the next chapters, I will study the increased evolvability following migrations more thoroughly. In Chapter 5, the analysis will be done in the context of genetic operators. In Chapter 6, I will study the migration process in detail to show how the increased evolvability may happen at genomic level. Finally, I will search for this behavior across various fitness landscapes and analyze if we can enhance it.

## Chapter 5: Migration as a genetic operator

The inter-island evolution significantly changes the dynamics of intra-island evolution. This happens as a result of migration between islands, which can be treated as yet another genetic operator, interacting with standard operators. Treating migration as an operator is common for biologists, who classify it (often under a name of *gene flow*) as one of a few major evolutionary mechanisms.

In this chapter, I will analyze what the role of migration is and how it is related to the other operators. The main difference between migration and other operators is that the latter operate only at intra-island, *local* level, whereas migration operates at the inter-island, *global* level. In addition, we will see that one of migration's roles is to connect the two levels of evolution long-term, in addition to a relatively weak direct impact it creates on target islands.

### 5.1 Migration

I will start by analyzing the migration operator alone, which means that there will be no selection or reproduction. I performed simulations with a simplified GA model (like EA-1) in which I represented individuals by unique IDs and tracked them through generations. This was possible, because with no mutation and recombination individuals do not change. Also, with absence of these operators, no representation of genomes is necessary, and without selection, no fitness values were needed. The

model become more complex later in this chapter, when I study the other genetic operators. For the infinite population I used a simple mathematical model computing percentages of alleles.

Unless stated otherwise, the experiments in the whole chapter were run with the parameters settings as follows. I used 20 islands of 20 individuals, or a single population of size 20 when experimenting with no migration. The migration probability for each individual was  $p = 0.2$  (it resulted in  $\alpha = 0.2$  for unidirectional migrations, and  $\alpha = 0.4$  for bidirectional migrations). For migration and no other operators, the whole population is cloned from generation to generation. Therefore, it makes no sense to vary the migration interval and I assume a migration in every generation ( $i = 1$ ). Migration interval will be increased when I analyze the interaction between migration and other operators — and migration interval plays a role.<sup>1</sup> Genome lengths (when used) are provided for each experiment.

As mentioned before, in this chapter I choose a target island independently for each migrant. This results in migrants migrating more evenly in the system, to all potential neighbors and at smaller rates. Because I do not use any selection (other than stochastic), the interaction between migrants and locals is not as sensitive to parameters, in particular to migration size. When studying operators alone, the choice of migration mode should also have smaller impact.

---

<sup>1</sup>To compare the results with later experiments it would be enough to insert periods of no change (horizontal lines in charts) of a length equal to migration intervals between each two consecutive points on the plots.

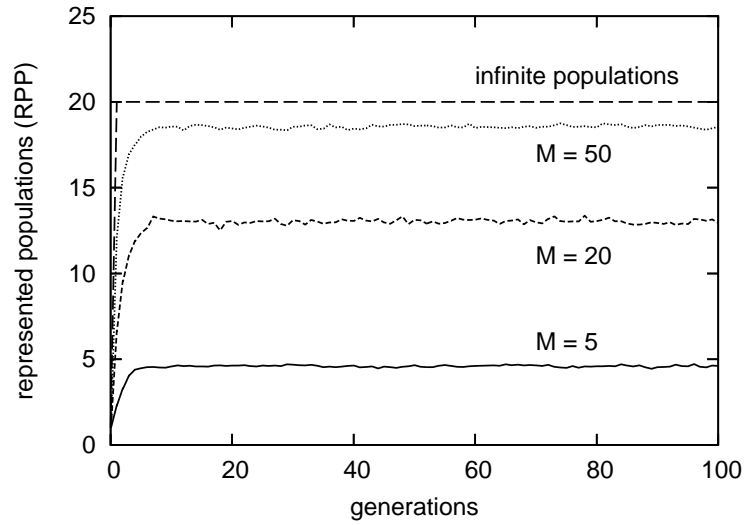


### 5.1.1 Bidirectional random–random migration

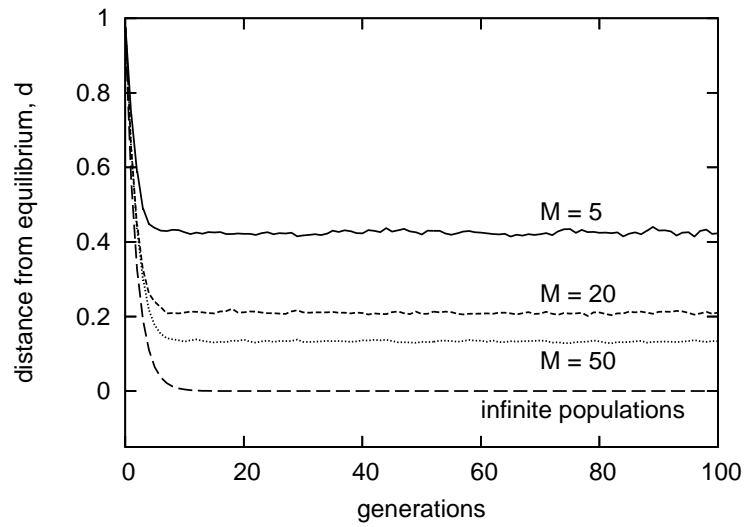
Let me begin the study with a bidirectional migration. It maintains  $w(t) = W$  and  $m(t) = M$  (number of distinct individuals globally and locally) for each island because no individual is lost. However, the contents of each island changes. If the island population was infinite, already after the first migration there would be a fraction of individuals coming from every other island (assuming a fully connected topology). In a finite island population, however, the island size  $M$  limits how many individuals of different origins can coexist. This limit will be rarely reached, even if  $N > M$ , because it is improbable to have exactly one representative from each island present. On the other hand, if  $M$  is big compared to  $N$ , the result will quickly show that all islands have their representative in all other islands (even if the exact number of representatives is different). So, for the number of islands having their representatives in any island we have  $RPP \leq \min(M, N)$  (see Appendix B for definition of  $RPP$ ). In Fig. 5.1a, I show an average  $RPP$  for different sizes of islands. For finite populations islands need some time to converge, because of granularity and stochasticity.

Using infinite populations and bidirectional migration, with time, each island should approach the equal percentage of individuals originating from every other island,  $\frac{1}{N}$  (as long as there is a migration path between the islands). Let us call this state a migration-linkage *equilibrium*. This convergence is intuitive, since a constant mixing of genetic material should distribute it evenly in the end. In Appendix E, I prove this more rigorously.

In Fig. 5.1b, I plot the distance  $d$  from equilibrium (see Appendix B) for both infinite population islands and finite population islands of different size ( $p = 0.2$ , as mentioned). We see that in the experiments, islands indeed converge toward the



(a) RPP



(b) Distance from island equilibrium,  $d(t)$

Figure 5.1: Bidirectional migrations mix individuals from different islands.

equilibrium point. One can see that as the island size becomes smaller, the distance from the equilibrium becomes bigger, because the number of representatives must be an integer. It may be impossible to reach the equilibrium point for finite populations because of too small of a number of individuals to distribute individuals evenly.

I conclude that, as expected, migrations mix the contents of islands, eliminating a “migration-linkage disequilibrium” resulting from certain individuals being evolved only in some islands. This drives the system toward migration-linkage equilibrium, where individuals originating from all islands evolve next to each other. As a result, bidirectional migrations significantly decrease inter-island diversity (by making all islands similar) and at the same time preserve local diversity (or possibly even increase it, if initial parents were not chosen randomly).

### **5.1.2 Unidirectional random–random migration**

With infinite populations there is not much difference between bidirectional and unidirectional migrations, assuming that the migrations still spread individuals to all other islands. This is because with infinite populations no individual is lost as a result of migration with random immigration, and only the individuals in the island changes. Of course, infinite populations are impossible in practice. When I use unidirectional migration with finite populations in islands, each time migrants replace some individuals in the target island, those individuals are lost. In a system where such random migrations are the only operator, losing individuals causes drift which finally leads to a situation with only one individual being left.

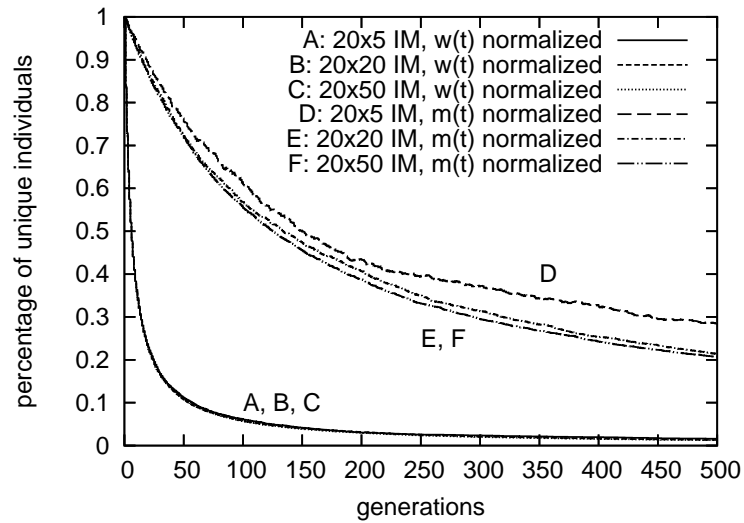
Since in each migration  $\alpha$  individuals of an island are replaced with migrants, it is tempting to assume that in every generation an  $\alpha$  part of individuals in the system

are gone, resulting in an exponential decay. While this model holds reasonably well in the beginning of a run, it does not properly describe the situation in the later generations, when copies of the same individuals circulate and a more complex model is needed. In fact, individuals disappear much slower in later generations, as shown in Fig. 5.2a for experiments I performed with various  $M$  values.

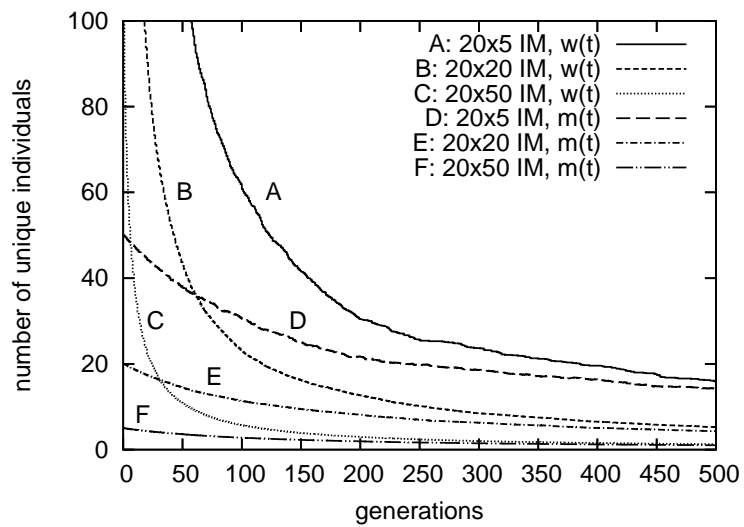
We can observe that  $\bar{w}(t)$  drops relatively quickly already in the beginning of a run, while  $\bar{m}(t)$  degrades slowly. This situation leads to a state in which a subset of individuals have survived in multiple copies and make up the entire content of all islands, thereby making the islands similar. In fact, if we compare the number of unique individuals in the system  $w(t)$  and an average number of unique individuals in an island  $m(t)$ , we see that they come close to each other, meaning that islands must share copies of the same individuals that survived in the system. This is shown for three different settings in Fig. 5.2b.

To understand why this happens, I analyze a migration of a single individual, assuming that all migrations can be decomposed to or approximated by repetitive migrations of single individuals if there are no operators acting in between. For simplicity, I also assume that such a migration randomly chooses an individual from the pool of all individuals and substitutes a randomly chosen individual in the whole pool. In such a case, for an individual to disappear from the system, there must be a single copy of it, and it must be chosen to be overwritten with a migrant. If an individual with more copies is chosen, it does not disappear immediately, and may still increase its presence later as a result of migrations (and prolong the time until convergence).

The number of each individual's copies in the system may be modeled by a Markov



(a) Global,  $\bar{w}(t)$ , and local,  $\bar{m}(t)$ , percentage of unique individuals



(b) Global,  $w(t)$ , and local,  $m(t)$ , number of unique individuals

Figure 5.2: Local diversity decreases slower than global diversity. Unidirectional random-random migrations, for different  $W$ .

chain. The states in the Markov chain are different quantities of the given individual. For this quantity to increase in a migration (from  $k$  to  $k + 1$ ), the individual must be chosen (with probability  $\frac{k}{W}$ ) and must replace another individual (with probability  $\frac{W-k}{W}$ ), which has a combined probability  $p_{k \rightarrow k+1} = \frac{k(W-k)}{W^2}$ . Similarly, to decrease the number of an individual's copies by one, a different individual must be chosen and replace one of the copies of the given individual, which occurs with the same probability of  $p_{k \rightarrow k-1} = \frac{k(W-k)}{W^2}$ . The last option, which maintains the number of a given individual's copies in the system, is that either copies of the individual will be chosen both for emigrating and immigrating, or a different individual will be chosen and will replace another individual different from the given one. Such situation has a probability of  $p_{k \rightarrow k} = \frac{k^2 + (W-k)^2}{W^2}$ . Those three options' probabilities sum to one. I create the transition matrix for the Markov chain, and compute probabilities  $p_k$  that a given number  $k$  of copies of the individual exists (a probability that the system is in the state  $k$ ). Since this number can change only by one (so there are always only two neighboring states), the equation for  $p_k(t + 1)$  is the following:

$$\begin{aligned}
p_k(t + 1) &= p_{k-1}(t)p_{k-1 \rightarrow k} + p_{k+1}(t)p_{k+1 \rightarrow k} - p_k(t)p_{k \rightarrow k-1} - p_k(t)p_{k \rightarrow k+1} = \\
&= p_{k-1}(t) \frac{(k-1)(W-(k-1))}{W^2} + \\
&\quad p_{k+1}(t) \frac{(k+1)(W-(k+1))}{W^2} - 2p_k(t) \frac{k(W-k)}{W^2}
\end{aligned}$$

We saw that after islands exchanged individuals, diversity stays high significantly longer than the predictions of a *naïve* model exponentially eliminating individuals.

Due to a possible random walk of the quantity of a given individual, its expected time to reach zero is much longer when compared to the case when this quantity would constantly decrease. To illustrate this, in Fig. 5.3, I show a simulation of a random walk, assuming that the individual has already reached  $p_k = 0.1$  (with  $W = 1000$ ) and using the probabilities from the model above. It shows how a given individual may remain in the system for quite a long time, even though its percentage temporarily decreases. It is easy for an individual to get eliminated early in the run, but if it is “lucky” to stay longer, it becomes more and more difficult to get rid of it later.

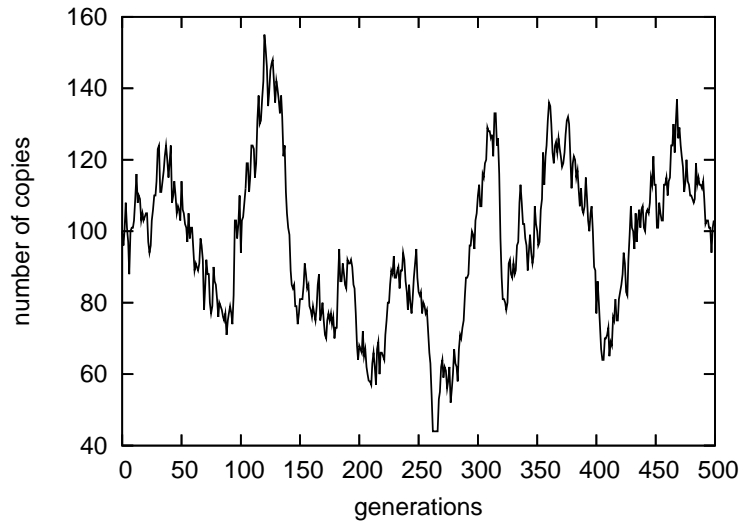


Figure 5.3: A sample simulated random walk of the number of copies of an individual, assuming it has reached  $p_k = 0.1$  with  $W = 1000$ .

The numbers  $p_k$  obviously may be understood also as the expected number of individuals having  $k$  copies. Therefore the expected number of unique individuals in the system are equal to  $w^*(t) = \sum_{k=1}^W W p_k(t) = W(1 - p_0(t))$ . In Fig. 5.4, I show a comparison of  $w^*(t)$ , and the average observed number  $w(t)$  in an actual experiment with single-individual migrations, for  $W = 1000$ .

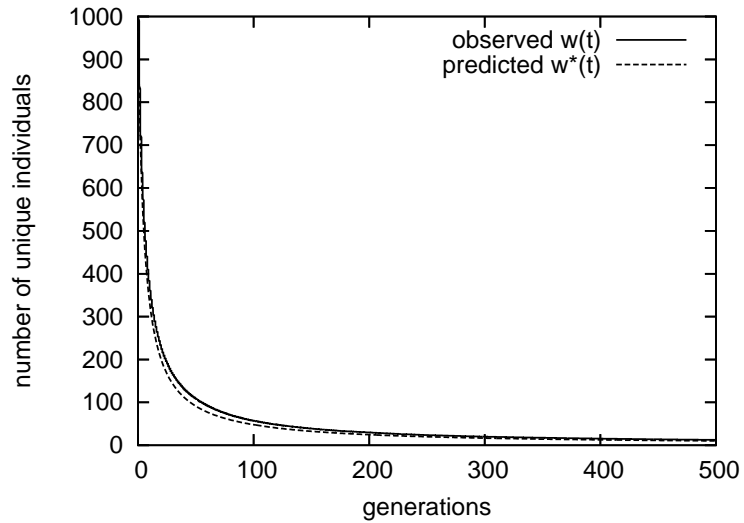


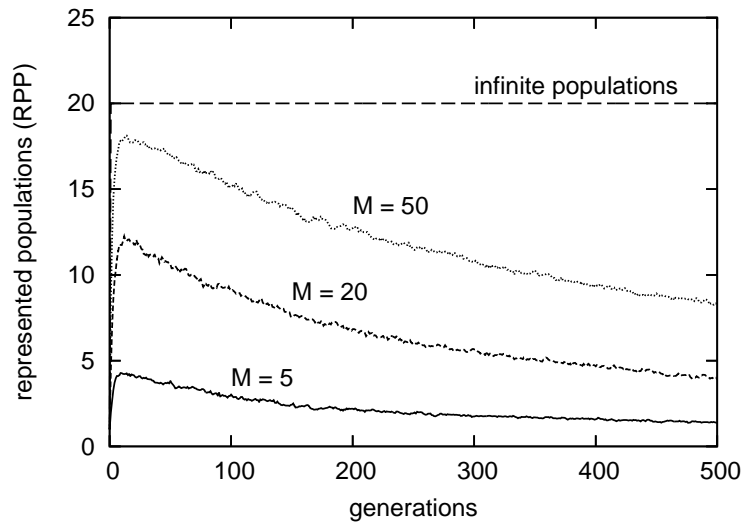
Figure 5.4: The expected  $w^*(t)$ , and observed  $w(t)$ , number of unique individuals compared. Unidirectional random-random migrations,  $N = 20$ ,  $M = 50$ .

It is important to understand how  $RPP$  changes during the run because, as we know, for certain problems mixing genetic material from different islands can be a path toward a better solution. For bidirectional migrations we saw, that  $RPP$  increases during the run and stabilizes to the number of all islands, if possible. In the presence of drift, when islands converge to single individuals,  $RPP$  also must drop to one. In Fig. 5.5a, I show how  $RPP$  changes in time for different island sizes, including a simulation of a model of an infinite population.

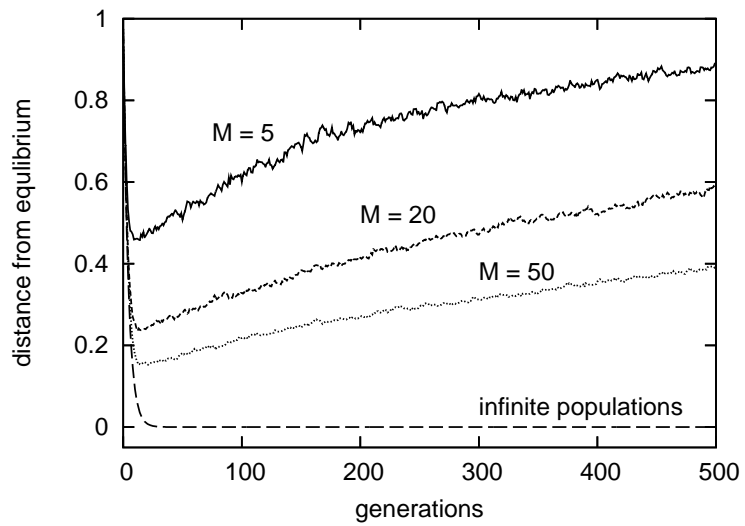
An unavoidable effect of islands converging to a single individual is a steady drift away from the equilibrium point. This is depicted in Figure 5.5b, again for different island sizes. Of course, no drift is happening for the infinite population.

I conclude that even though unidirectional migrations lose some individuals due to random replacement, they also quite effectively mix island populations, making them similar to each other. They preserve local diversities, and whereas bidirectional





(a) *RPP*



(b) Distance from island equilibrium,  $d$

Figure 5.5: Unidirectional random–random migrations first mix individuals, then cause a drift.

migrations just regroup individuals, unidirectional migrations result in a subset of initial individuals to fill all islands. In the second stage, stochastic replacement of individuals by migrants cause islands to slowly lose the local diversities, and as a result island populations become gradually less mixed, in the sense of individuals originating from different islands.

### 5.1.3 Unidirectional selective migration

Let us assume that all individuals in the system have some fitness values from the range  $[0, 1]$ . In general, this assumption is enough to perform experiments with rank-based selection strategies practically without any loss of generality (I assume that cases, with the same fitness accidentally assigned to two individuals, are negligible). In particular, one can experiment with migrations where migrants are not chosen randomly, but rather based on their fitness. It is known (Cantú-Paz, 2001) that such migrations cause an increase of selection pressure in the system and a shorter convergence time. I will use tournament–random migrations, because a tournament allows me to easily control the selection pressure by changing its size.

Using a positive selection pressure in migrations has a very visible effect. Islands no longer lose individuals due to a random drift, but rather because of repeated selection. This results in a relatively fast convergence of the whole system to a single individual, regardless of a configuration, as confirmed by the experiments shown in Fig. 5.6a for binary tournament–random migration policy.

There is a question as to whether changing  $M$  and  $N$ , but keeping  $W$  constant has any influence. One could expect such influence, since choosing the best or good individuals in smaller islands should be less selective. In particular with islands of size

one any emigration policy must be equal to random emigration. In my experiments, however, the smallest islands were of size ten and produced effects similar to the larger islands.

When I draw plots of  $\bar{w}(t)$  and  $\bar{m}(t)$  in Fig. 5.6b,<sup>2</sup> , and compare with Fig. 5.2, we see that with selective migrations local convergence is similar to the global convergence. Worse individuals from any island have a lesser chance of being migrated, but the same chance of being replaced. Therefore, the probability of increasing their number is much less than the probability of decreasing their number ( $p_{k \rightarrow k+1} < p_{k \rightarrow k-1}$ ), and their random walk is biased toward zero. Under this situation not only do islands become similar to each other converging to better individuals, but the worse individuals are quickly discarded.

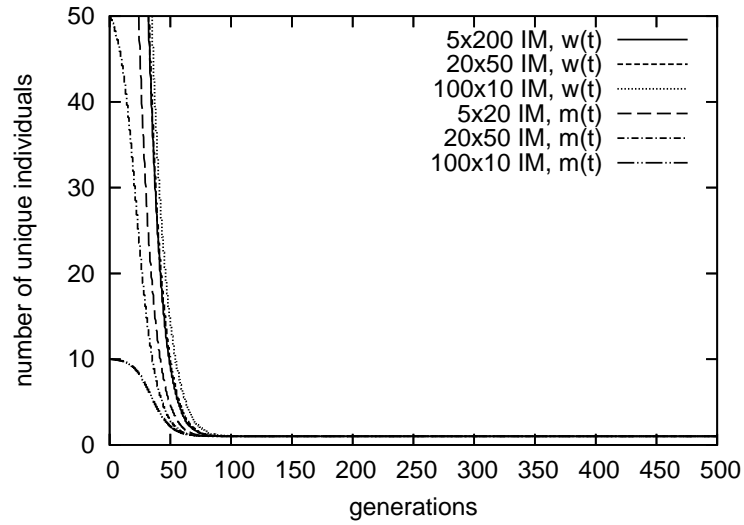
I conclude that a strong emigration policy causes a fast convergence to a single individual, and random policy served much better for inter-island mixing.

## 5.2 Selection

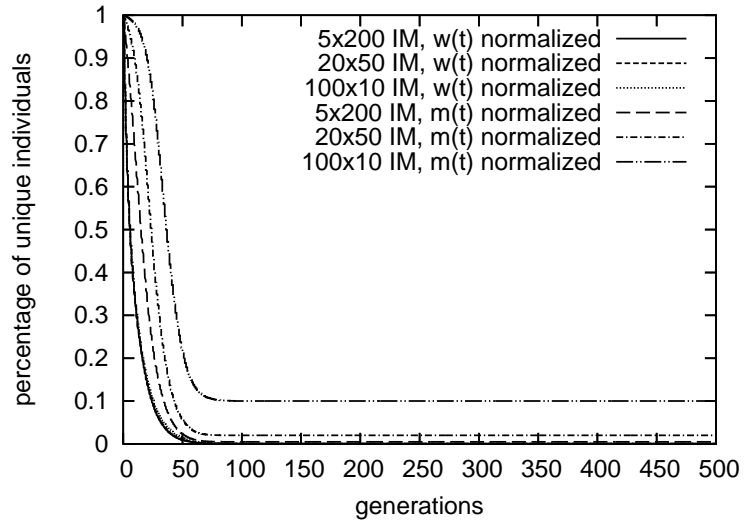
So far, I have assumed that nothing happens inside islands between migrations. Let us now see the effect of selection operating within islands. For completeness, I will first analyze selection alone, which means that we will have a set of isolated islands. Such an analysis is obviously very similar to existing studies of takeover times for single populations. In the absence of reproduction operators there is no difference between parent selection and survival selection — in each generation some individuals are selected to be copied into the next generation.

---

<sup>2</sup>Note, that since  $\bar{m}(t)$  cannot drop below  $\frac{1}{M}$ , smaller islands seem to converge to higher values — however we know that they all converge to a single individual, so I treat it as an artefact.



(a) Global  $w(t)$  and local  $m(t)$  number of unique individuals



(b) Global  $\bar{w}(t)$  and local  $\bar{m}(t)$  percentage of unique individuals

Figure 5.6: Strong (binary tournament–random) migration policy causes a fast loss of both global and local diversities (in absence of other operators). Unidirectional selective migrations shown, for different number/sizes of islands,  $W = 1000$ .

### 5.2.1 Uniform stochastic selection

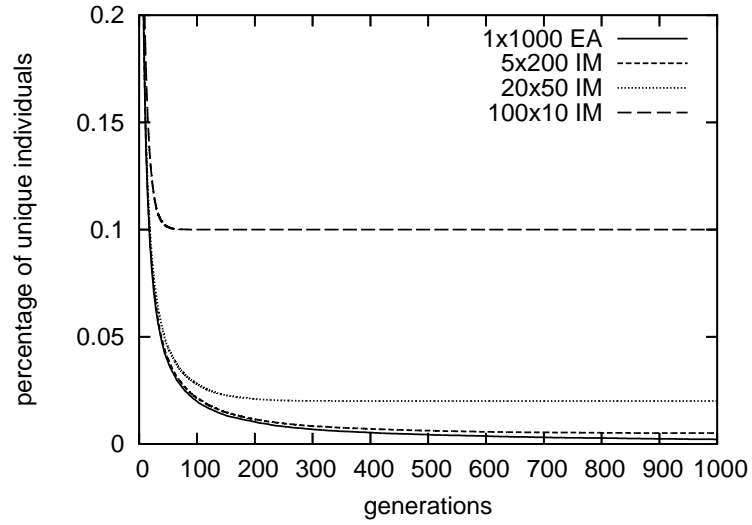
As we know from the studies of single population models, even a random (uniform stochastic) selection causes drift and convergence of populations. What is worth stressing with IMs, however, is that in the absence of migrations, islands usually converge to different individuals (although of course  $w(t) \leq \sum_i m_i(t)$ ). Therefore, the average number of distinct individuals in a converged system is close to the number of islands. An IM naturally maintains a higher level of inter-island diversity in the system. This is shown in Fig. 5.7a.

### 5.2.2 Tournament selection

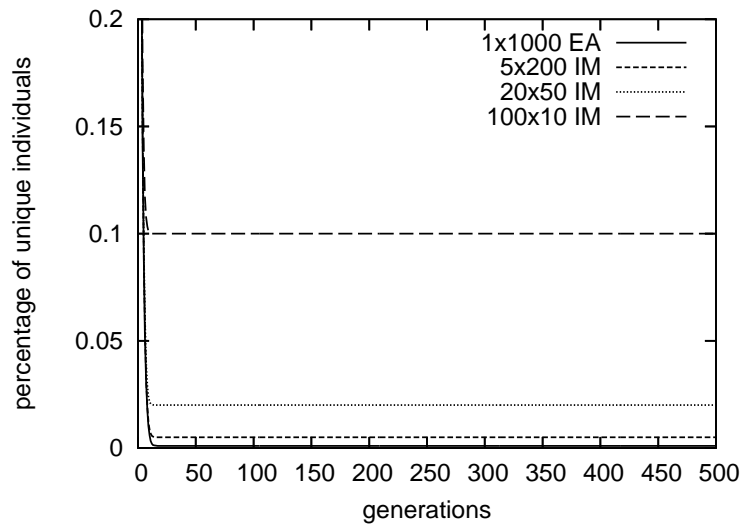
An island model with a positive selection pressure, for example in the form of a tournament selection, also behaves similarly to its single population counterpart. The convergence occurs faster than in the case of uniform stochastic selection. Again, we observe that the number of unique individuals in the system  $w(t)$  remains close to the number of islands, which is shown in Fig. 5.7b.

In practice, when we measure the diversity as a spread of individuals, rather than their number, the diversity will often be high too, because the individuals to which particular islands independently converged can be quite distant from each other.

Selection preserves inter-island diversity and quickly eliminates the local diversity. Note, that this effect is exactly opposite to migration, as explained in Section 5.3.



(a)  $\bar{w}(t)$  with uniform stochastic selection



(b)  $\bar{w}(t)$  with binary tournament selection

Figure 5.7: Selection causes fast local convergence, but preserves inter-island diversity. Experiments for different number/sizes of islands,  $W = 1000$ .

## 5.3 Selection with migration

As we have seen, both migrations and selection cause the loss of losing genetic diversity, although they act in different ways. Selection affects diversity inside islands, which decreases quickly when islands converge. However, because they converge to different individuals, the inter-island diversity is still maintained. Migrations act differently, affecting inter-island diversity, which decreases due to the islands becoming similar to each other. In this case, local diversity is still maintained for some time. One can expect that with selection and migrations acting together, the convergence will occur even faster, both at a global and local level. This is in fact observed, and shown in the following subsections.

### 5.3.1 Uniform stochastic selection

I performed a set of experiments in which I used uniform stochastic selection (just producing drift), with a range of migration types, from bidirectional random migration (the weakest), through unidirectional random–random, to unidirectional binary tournament–random policy (the strongest). In all of them I obtained similar results,<sup>3</sup> and I will show only representative experiments.

First, global convergence effects of both the selection and migration accumulate. Even in the case of bidirectional migration, simply mixing island populations is enough so that selection brings the global diversity to a single individual. Figure 5.8 shows a similar global convergence for unidirectional migrations, in which the drift resulting from migrations plays an additional (although minor) role. I have plotted the drift to various combinations of  $N$  and  $M$ , but kept  $W = 1000$  and the migration interval

---

<sup>3</sup>Stronger policies cause a little bit faster convergence.

of 10. As expected,  $\bar{w}(t)$  drops all the way towards zero, quickly going past the value of  $\frac{N}{W} = \frac{1}{M}$ , which was the limit in the case with no migrations — although when the number of islands is larger,  $w(t)$  remains higher for longer time.

We see that convergence occurs in two stages. In the first stage (around 50 generations), the selection dominates migration, causing a decrease in the number of individuals, which is similar to the case without migrations. The effect of migration is visible in the later stage, after reaching  $\frac{1}{M}$ , when the islands are mainly getting mixed.

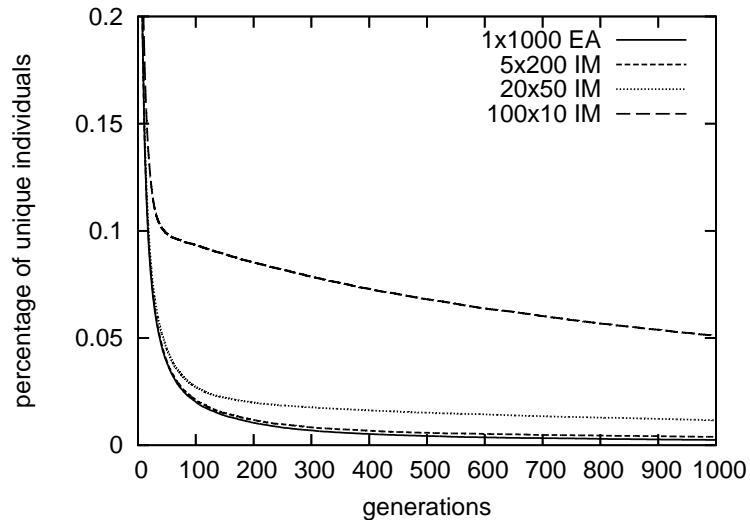


Figure 5.8: The percentage of unique individuals ( $\bar{w}(t)$ ). Uniform stochastic selection and unidirectional migration for different numbers/sizes of islands,  $\alpha = 0.01$  and  $i = 10$ .

In Fig. 5.9, I show curves for both the local and global number of individuals. We see that, according to our previous observations, migration is acting in an opposite way to selection (even though they ultimately cause both the local and global convergence). While selection would keep the inter-island diversity high, migration



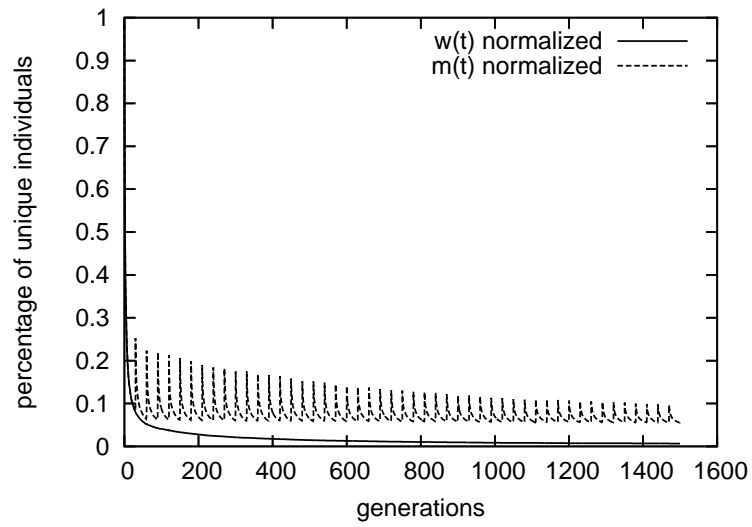
makes it impossible and ultimately causes the global convergence. At the same time, while selection causes islands to converge, migration is actively preserving local diversity (the “spikes” in the charts).

If an individual is replaced by a migrant new to an island, the number of unique individuals does not change. However, the decrease of  $m(t)$  may result from either sending copies of the same individuals to one target island, or sending individuals to another island and then back. This is less likely with more islands, since the migrants can be sent to any other island, rather than back to the original one.

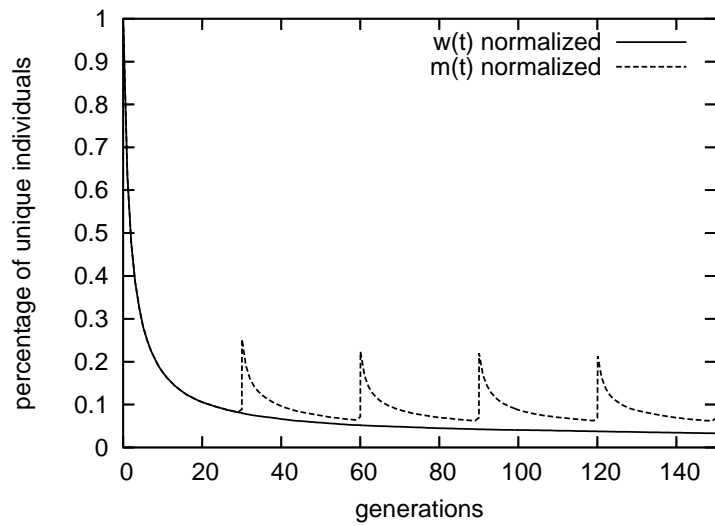
In Fig. 5.10, I show experiments with differing numbers of islands (2, 5, 20 and 500). The island size was kept constant and each individual in the system was unique. In this trial, I have noticed a faster loss of locally unique individuals for smaller  $N$ , particularly for 2 and 5 islands (Fig. 5.10). With fewer islands converging, one may find many copies of the same migrants. When  $N$  is smaller, it is more likely that an individual will visit the same island again, which creates multiple copies of it and decreases the number of unique individuals.

### 5.3.2 Binary tournament selection

Another set of experiments was performed for a binary tournament selection, and they all gave comparable results for various migration policies, which differ a little bit, however, from the results seen so far for uniform selection. I show a representative chart in Fig. 5.11. A stronger selection causes a quick convergence of each island to single individuals. A migration causes a temporary increase in diversity, but soon after this happens, in the next few generations continuous selection causes the diversity to decrease below the level preceding the migration, reducing the global number of



(a)  $\bar{w}(t)$  and  $\bar{m}(t)$



(b) close-up

Figure 5.9: Uniform stochastic selection and unidirectional migration act opposite to each other. Interval of 30.

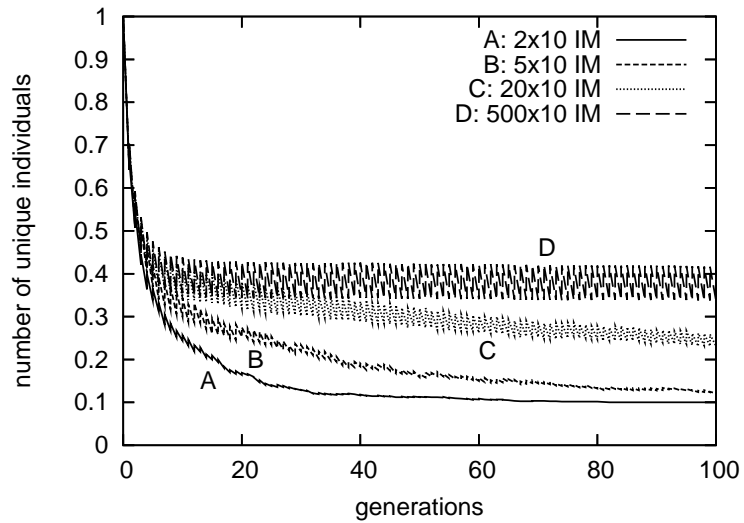


Figure 5.10: The number of locally unique individuals ( $m(t)$ ), with different  $N$  and constant  $M = 10$ . Uniform stochastic selection and unidirectional migration,  $\alpha = 0.1$ .

unique individuals  $w(t)$ . The whole system converges to a single individual much sooner than in the case of uniform stochastic selection.

## 5.4 Recombination

The third main force in EAs are reproduction operators, whose role is commonly understood as exploring the search space. There are two traditional operators: recombination and mutation. They operate locally in islands. In the following sections I will study how they interact with migration.

Whereas it was possible to arbitrarily assign fitness to individuals in previous sections, such experiments are in general impossible when using reproduction operators. A fitness value of each individual after mutation and recombination would have to be defined — however, this is precisely what fitness landscapes do! In other words, the way fitness changes when using reproduction operators is always the main

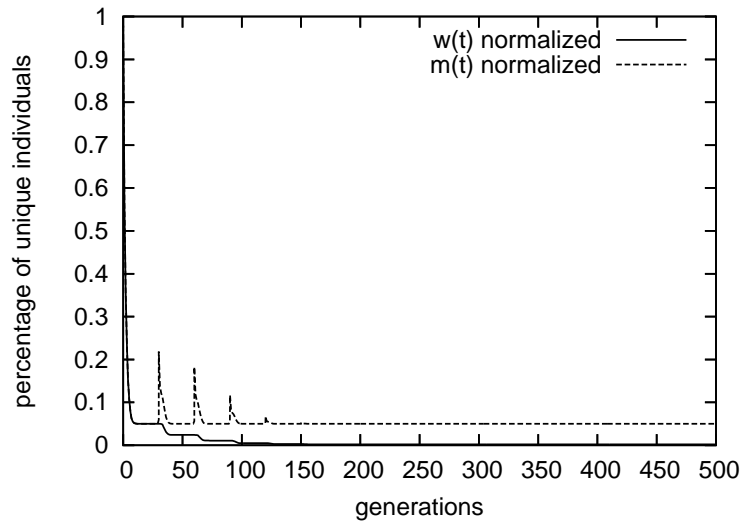


Figure 5.11: Global and local percentage of unique individuals ( $\bar{w}(t)$  and  $\bar{m}(t)$ ), binary tournament selection and unidirectional binary tournament–random migration, with interval of 30.

question for each problem we would like to solve with EAs. This relation changes with each problem, for many of them (e.g. for engineering domains) it requires lengthy simulations, and generally if we knew the formula beforehand, we could probably solve the problem analytically and not need EAs at all. Therefore, I show experiments with uniform stochastic selection and random migrations, which do not use any fitness values. To see what the relation is between reproduction, other operators and fitness, we will need concrete functions or problems, and this analysis is deferred to Chapter 8. For experiments with recombination I extended my simulation model to operate on actual genomes — I tracked alleles, each of which had a unique ID.

#### 5.4.1 Two-children deterministic uniform recombination

I will first show some experiments with recombination alone. This analysis is known from existing studies of EAs, but serves as a reference point in this context.

When reproduction is the only operator in a population, it results in mixing the genes of individuals and losing linkage information. An infinite population converges toward a so-called Robbins equilibrium (Geiringer, 1944), where the percentage of each individual that could be created by recombination from initial parents, is equal. Finite populations converge to a point in the neighborhood of the equilibrium.<sup>4</sup>

A recombination operator may produce either one child, or two children, genetically complementary to each other. The second case has an advantage of preventing the loss of genetic material in populations. With one child recombination, genetic diversity gradually decreases and the population slowly drifts away from its equilibrium.

To preserve all the genetic material, not only recombination must produce two children, so that no alleles are lost from two given individuals, but also each individual must be chosen for reproduction deterministically. I took special care to ensure it. Although individuals were paired randomly, each individual in a population was chosen exactly once for recombination. In Fig. 5.12, I show how *RII* grows and stabilizes for a different combination of population size and genome length, for two-children uniform recombination. As expected, recombination ultimately mixes alleles coming from multiple initial individuals.

## 5.5 Recombination with migration

Using both reproduction and migration operators, I observed a mixing of genes from all islands. This is analyzed in the following subsections.

---

<sup>4</sup>Do not confuse this equilibrium with the migration-linkage equilibrium defined earlier by myself.

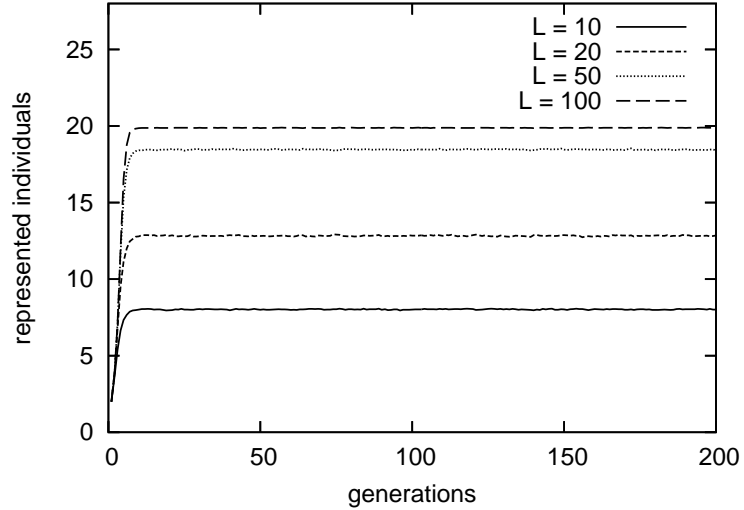
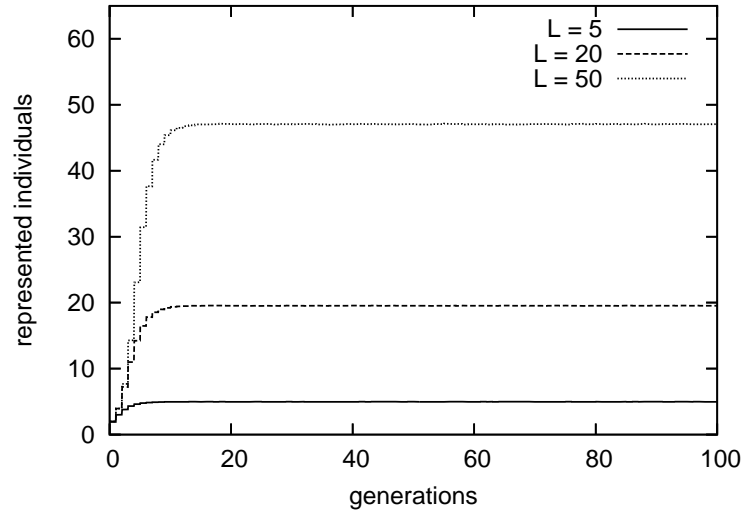


Figure 5.12: The number of represented individuals per individual ( $RII$ ), using two-children deterministic recombination, for various genome lengths. Single population.

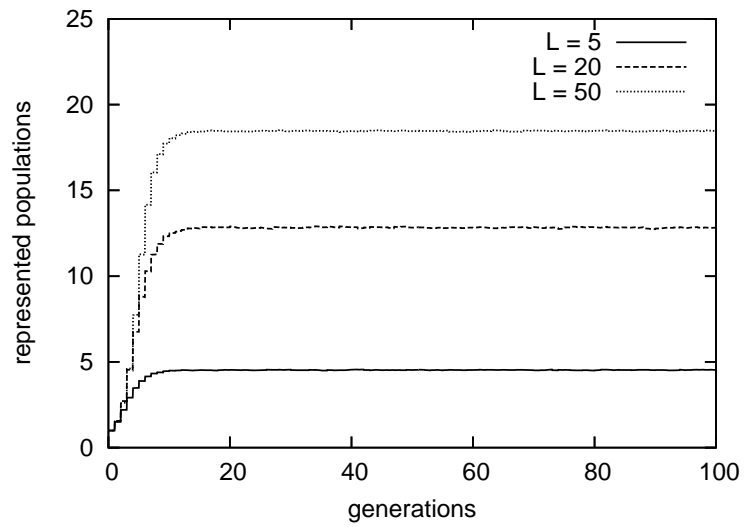
### 5.5.1 Two-children deterministic uniform recombination and bidirectional random migration

When one uses two-children reproduction and bidirectional migrations, no loss of genetic information occurs. In Fig. 5.13a, I show the increase of  $RII$  far above island size  $M$ , and limited by the genome length. Individuals naturally inherit genes from representatives of many islands, even when a small migration size is used. The spread of genes between islands is directly shown in the next Fig. 5.13b, where I plot the  $RPI$  number. We see that the  $RPI$  number quickly increases to the number of all islands  $N$  (unless the length of genomes  $L$  is too short, as the inequality  $RPI \leq \min\{N, L\}$  holds).

Both  $RPP$  and  $RIP$ , accordingly, show fast growth toward  $M$  or  $W$  (Fig. 5.13c and 5.13d). The genome length influence is less visible in the case of  $RPP$  since the number of islands is for all  $L$  much lower than the total number of genes in all

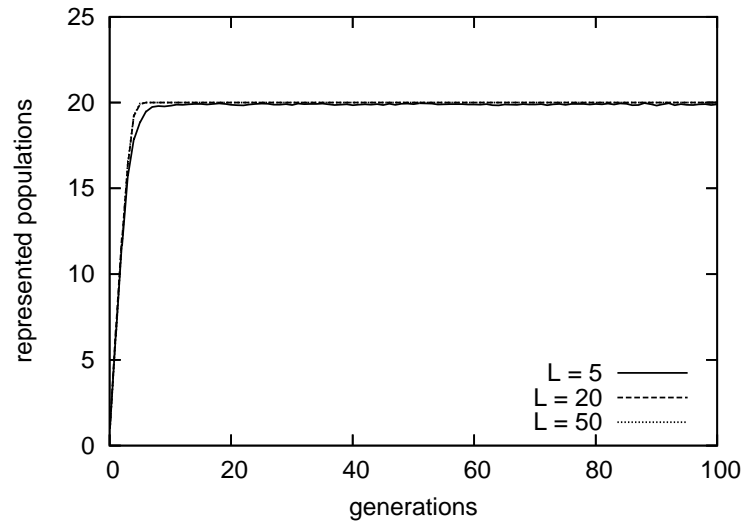


(a) *RII*

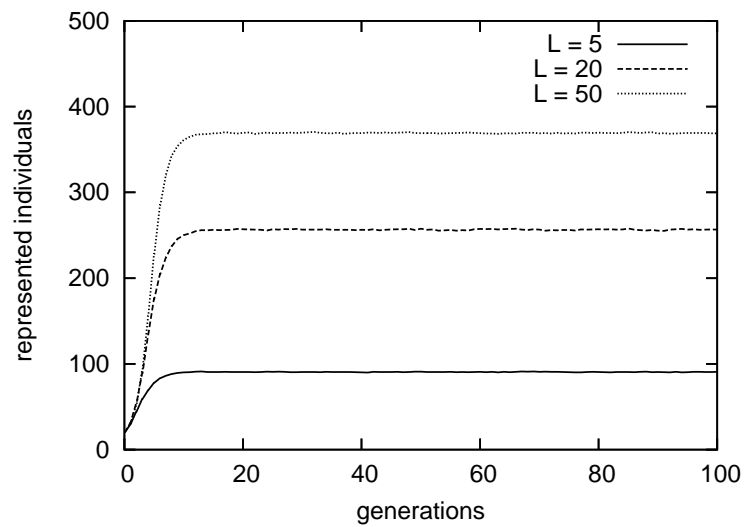


(b) *RPI*

Figure 5.13: Two-children recombination and bidirectional migrations mix alleles from all islands, for different genome lengths.



(c) *RPP*



(d) *RIP*

Figure 5.13: Two-children recombination and bidirectional migrations mix alleles from all islands, for different genome lengths.



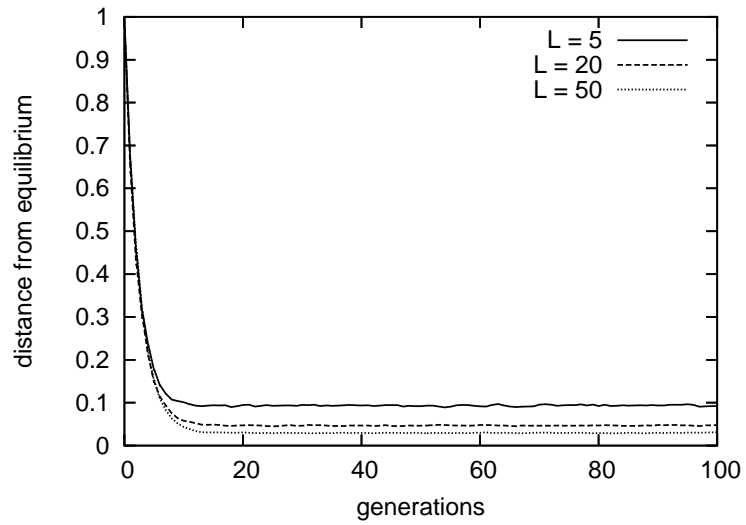
individuals. In the case of no recombination,  $RIP$  is obviously constant and equal to  $M$ . In my experiments the  $RIP$  number seems to be close to  $RPI \cdot M$ . This relation could be studied deeper, as it seems that it may be approximated by the above equation under certain conditions. In Fig. 5.14a and 5.14b, I plot corresponding  $d$  and  $d'$  measurements, which also show the mixing of individuals.

### 5.5.2 Two-children deterministic uniform recombination and unidirectional random migration

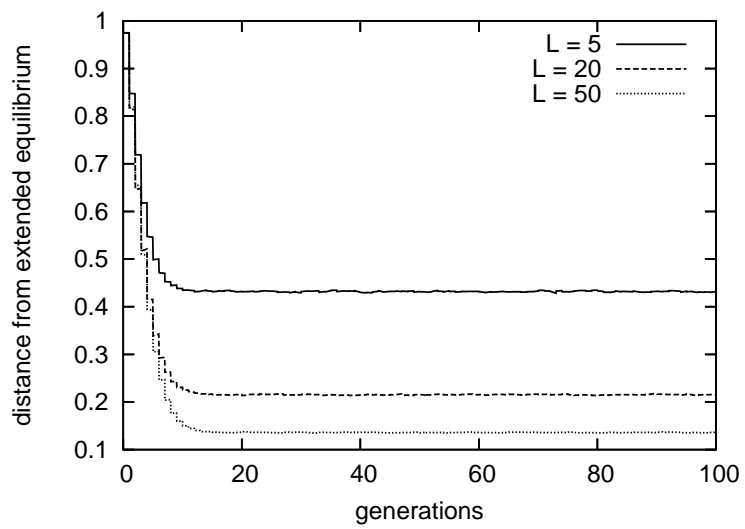
In the previous section, no loss of alleles occurred because both recombination and migration were only swapping alleles or individuals. However, we know that other migration types decrease the global diversity. This section will show how the behavior changes when those operators are more stochastic.

A unidirectional migration alone causes a slow loss of genes in the target island. In Fig. 5.15a we observe that recombination quickly recovers  $\bar{m}(t)$  back to the maximum. This result is because recombination is able to keep constructing new individuals from a decreasing number of available alleles. As a result, both  $d$  and  $d'$  seem to stabilize at low values, as shown in Fig. 5.15b, and provide a good mixture of genes from different islands.

Of course, using recombination to counteract drift cannot last infinitely. The decrease of  $w(t)$  occurs much later though (after thousands of generations in my experiments), as shown in Fig. 5.16a. Note, that in this case I have set the migration interval to 1 and extended the experiments to 4000 generations. In each run, each island converges to a random individual. Therefore, an average value from a non-converged island is equal to an average value of final individuals from multiple runs,

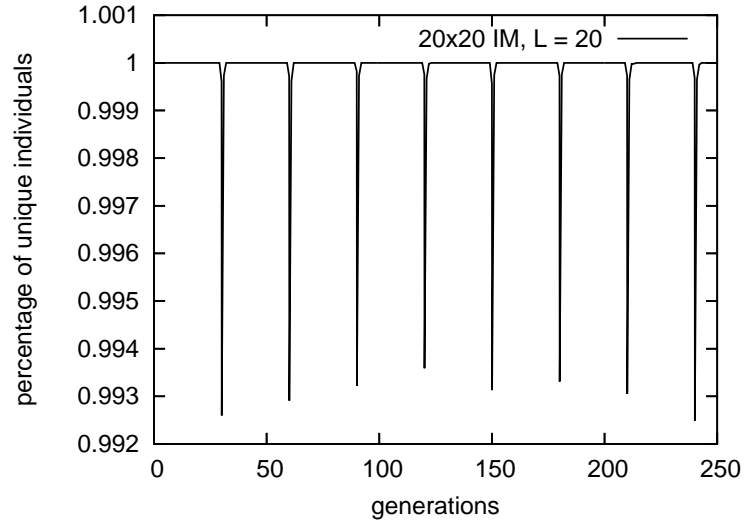


(a)  $d$

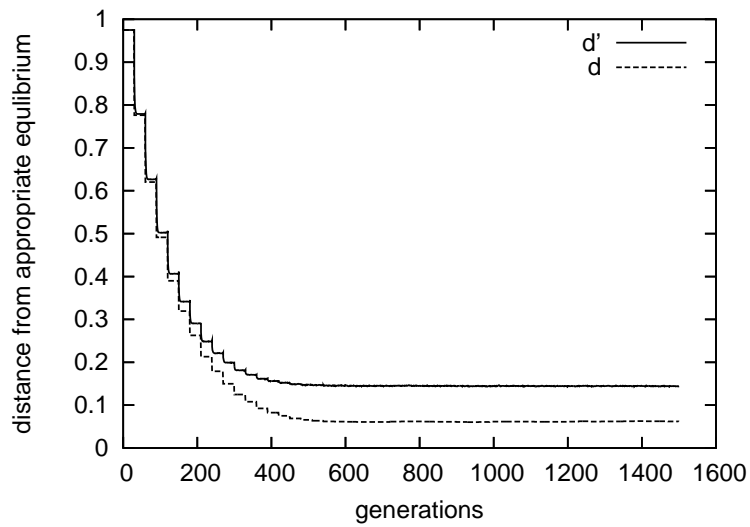


(b)  $d'$

Figure 5.14: Distances from island equilibrium. Two-children recombination and bidirectional migrations, for different genome lengths.



(a)  $\bar{m}(t)$ , initial phase



(b) initial stabilization of  $d$  and  $d'$  (long genomes)

Figure 5.15: Two-children recombination counteracts drift from a unidirectional migration (initial generations), interval of 30.

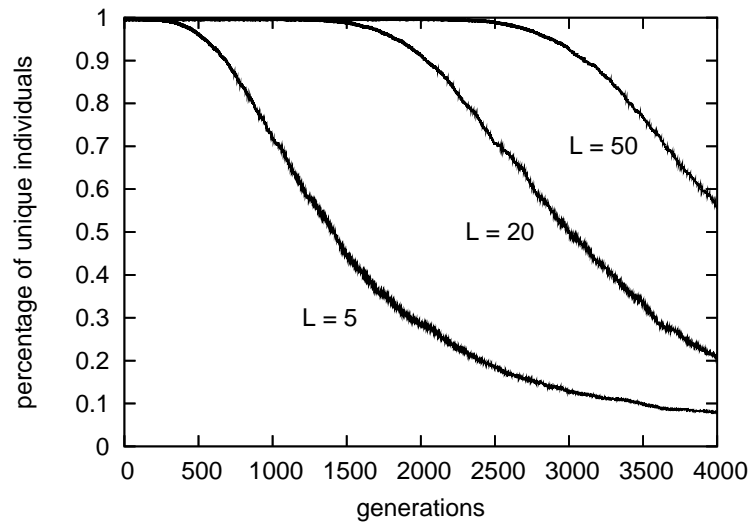
and distance  $d'$  remains constant. Distance  $d$ , however, describes the state of all individuals in an island, and with the convergence of the island it must grow and approach the  $d'$  distance, which describes an average single individual, as shown in Fig. 5.16b. Note that thanks to the recombination, even though the islands converge to single individuals, they still have representative genes from many islands.

## 5.6 Recombination with random selection

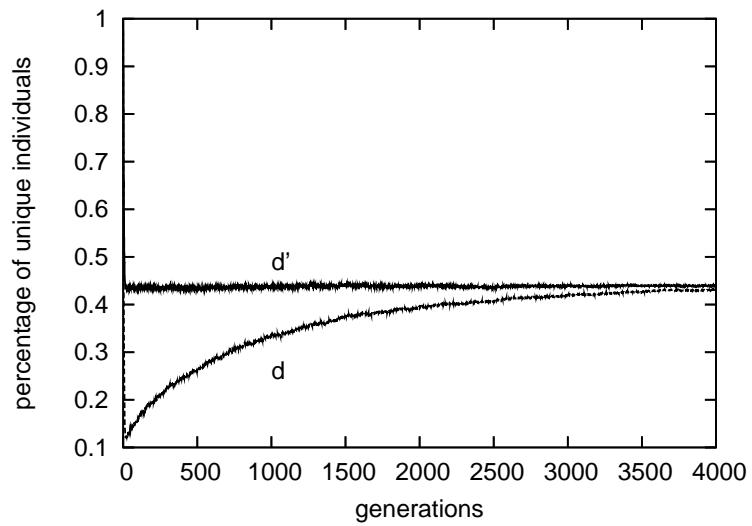
Selecting parents randomly for recombination is effectively equivalent to using a uniform stochastic selection in addition to recombination. Due to the limitations described earlier, I will limit the analysis to uniform stochastic selection. This study is included for completeness, since of course both recombination and selection have been studied for standard EAs.

Uniform stochastic selection and recombination cause the loss of alleles. There are two reasons for that. The first one is the choice of parents — if only some individuals in a population are selected to be parents (as opposed to choosing them in a pre-defined order), genetic diversity is obviously partially lost. Another reason for the loss of alleles is when only one child of a given pair of parents is created, because the unused alleles will be lost (unless they happen to be selected in another recombination). Regardless of the above, both of these setups are commonly used in practice, based on the ease of implementation.

Losing alleles ultimately makes the number of distinct individuals in a population decrease. How fast this happens is dependent on a number of factors. In Fig. 5.17a, I show a decreasing value of  $\bar{m}(t)$  for various genome lengths, using one-child stochastic uniform selection of parents. If genomes are longer, they have more genes to differ



(a)  $\bar{m}(t)$  ultimately drops



(b)  $d$  increases and approaches  $d'$ ,  $L=5$ .

Figure 5.16: Drift due to unidirectional migration occurs in later generations even though two-children recombination counteracts it.

between each other, and so the number of different individuals remains higher (even if each locus loses the alleles at the same pace as in the case of shorter genomes).

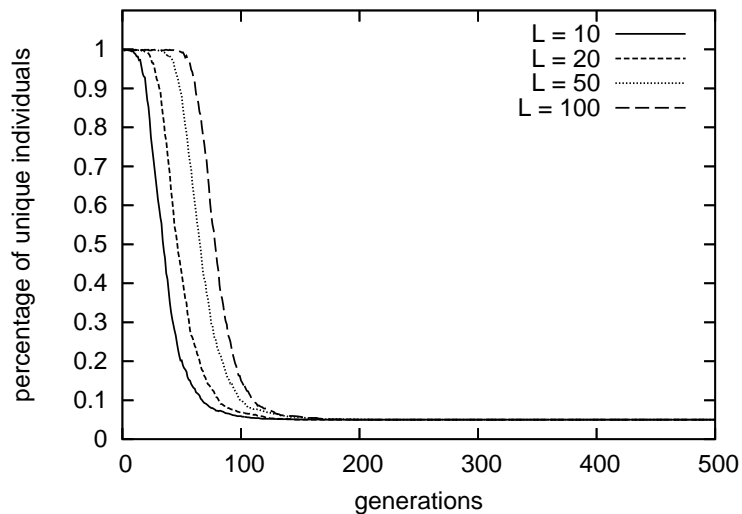
Increasing the size of populations (which often means decreasing the number of islands in IMs) makes convergence occur slower as shown in Fig. 5.17b. Similarly, using a uniform *deterministic* parent selection (each parent is chosen exactly twice) results in a slowing down of the convergence, as shown in Fig. 5.17c. Note, that in any case of one-child recombination, drift occurs much faster than in the case of two-children deterministic recombination with migration.

As an effect of reduced diversity, the *RII* converges to lower values than it was observed in prior tests (Fig. 5.18).

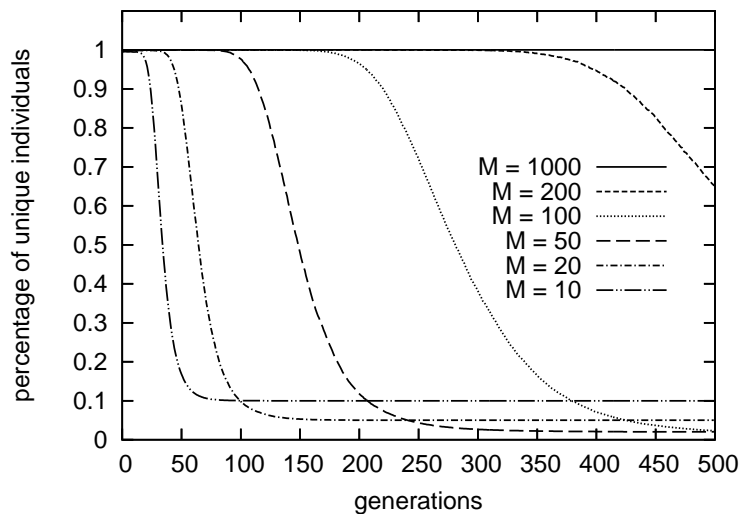
## 5.7 Migration-recombination-selection scenario

After seeing migration, selection and recombination working separately, I will now analyze their joint behavior. When selection (even as weak as uniform stochastic) is present, recombination is not able to compensate for the decrease in  $\bar{w}(t)$  as it was in the case of only migrations causing drift. In Fig. 5.19a, I show a drop in the number of unique individuals in the islands, under these conditions. A decrease in the number of unique individuals may occur at a pace dependent on many factors, including genome length  $L$  and island size  $M$ , but is inevitable.

Migration increases local diversity. This is visible in Fig. 5.19b, where I show a close-up of a part of a previous figure, which is similar to the already analyzed case of migration and selection only, but different from the case with recombination. To explain this situation intuitively: if the local diversity drops faster than the inter-island diversity (e.g. due to selection), migrations insert individuals not present in a

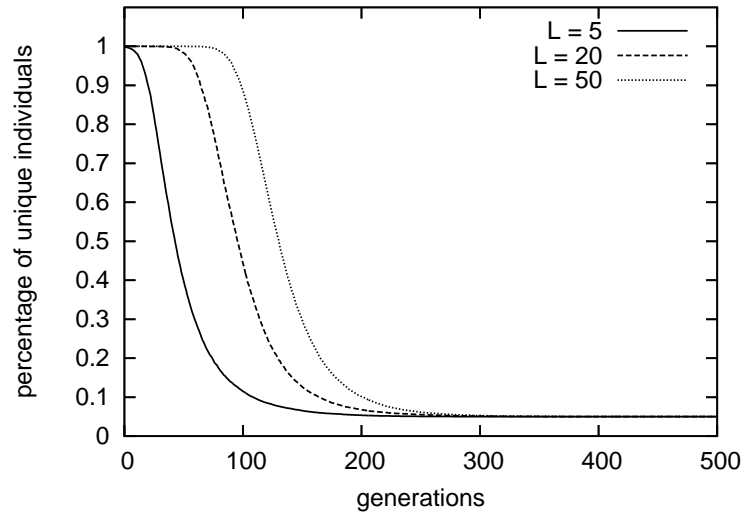


(a)  $\bar{m}(t)$ , one-child stochastic recombination, different genome lengths,  $M=20$



(b)  $\bar{m}(t)$ , one-child stochastic recombination, different population sizes,  $L=50$

Figure 5.17: Uniform stochastic selection of parents, and one-child recombination results in a drift. The percentage of locally unique individuals ( $\bar{m}(t)$ ), using one-child recombination.



(c)  $\bar{m}(t)$ , one-child *deterministic* recombination, different genome lengths,  $M=20$

Figure 5.17: Uniform stochastic selection of parents, or one-child recombination results in a drift. The percentage of locally unique individuals ( $\bar{m}(t)$ ), using one-child recombination.

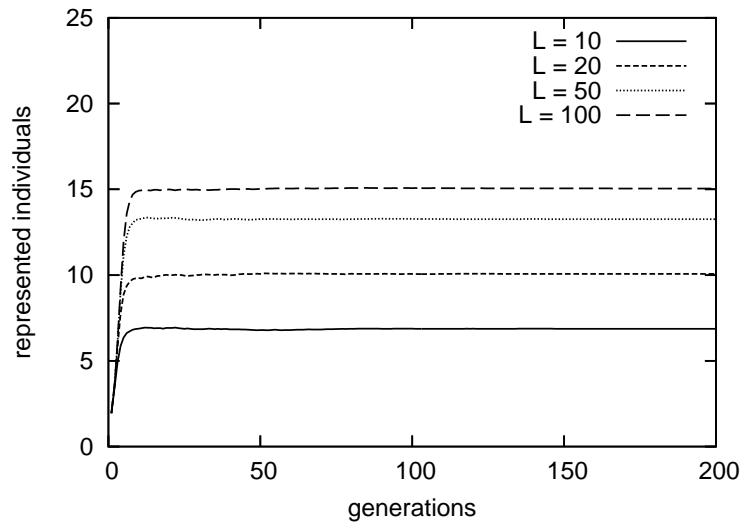


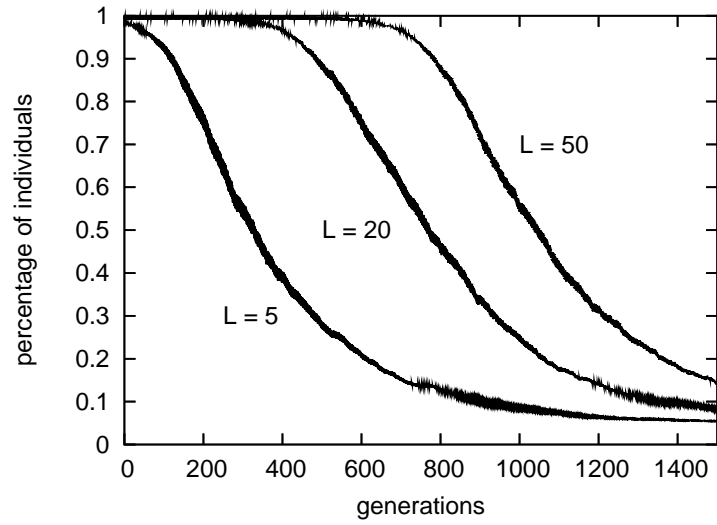
Figure 5.18: The number of represented individuals per individual ( $RII$ ) using one-child recombination, for different genome lengths.



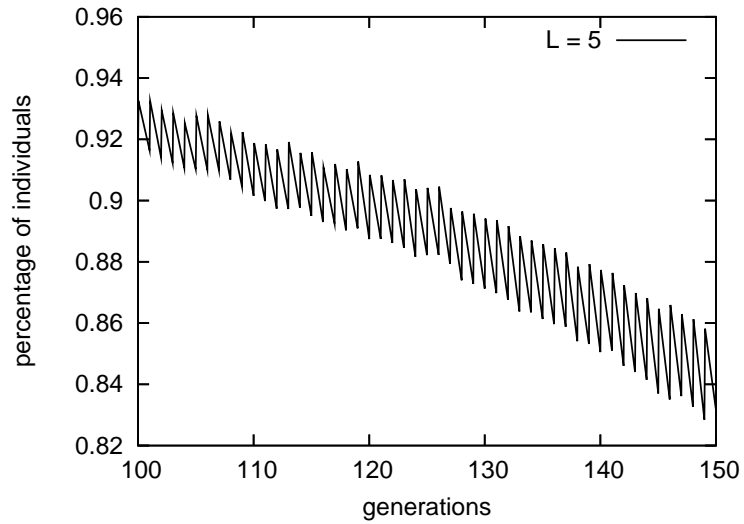
given island, and increase local diversity. If local diversity drops slower than inter-island diversity (e.g. due to recombination keeping it high), then islands become similar and migrations insert the same or similar individuals as the ones already present in the island and decrease local diversity. Of the different types of operators analyzed so far, selection (even uniform stochastic) causes the fastest allele loss, one-child recombination causes a slower loss, then migration is responsible for even slower loss, and finally two-children deterministic recombination causes no allele loss. Migrations, coupled with any “stronger” operator will slow down convergence, and coupled with “weaker” ones will speed up convergence (or make it exist at all). Therefore, theoretically, migration can decrease local diversity for some special cases. In any real situation, however, when selection is present (and usually it is stronger than uniform selection), migration will always cause a local increase of diversity.

When I increase the migration interval to 10, we see the effect of operators more clearly. Overall, drift occurs slightly slower, as shown in Fig. 5.20a and recombination is apparently able to increase the number of unique individuals temporarily, producing unknown (or forgotten) combinations, as shown in Fig.5.20b. After several generations due to random allele loss, the overall diversity drops to its level prior to any migrations, or even lower. In real applications, even though the diversity drops, some novel individual could have been created in the meantime, and selected to survive on the island.

The whole dynamics suggest a certain order in which operators act. First, migrations inject new alleles into the target island, then recombination mixes those alleles together with the local ones, producing a number of new individuals, and then



(a)  $\bar{m}(t)$



(b) close-up for  $L=5$ , operators interacting

Figure 5.19: Migration plus recombination slows down drift from selection, but cannot compensate it. One-child stochastic recombination and unidirectional migrations, different genome lengths, migration interval=1.

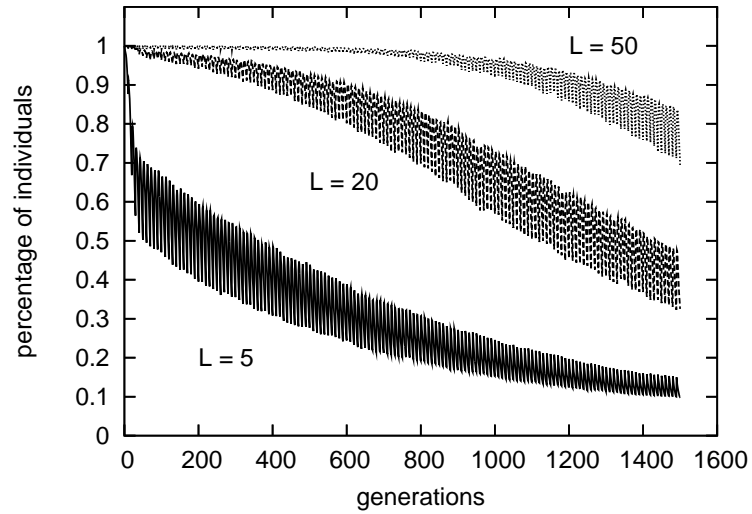
selection picks the new fit individuals. I will refer to this phenomena as *migration-recombination-selection scenario*. In this interpretation migration locally behaves similar to mutation in a GA, creating “material” for recombination.

Coming back to experiments with migrations in every generation (where we can observe global drift easier), in Fig. 5.21 – 5.22, I plot charts showing the mixing of islands. Due to faster convergence, the measurements reach slightly lower values. However, it seems that most of the mixing manages to occur before stochastic effects preclude it. When drift dominates,  $RPP$  converges to  $RPI$ , and  $RIP$  converges to  $RII$ , since each island becomes a set of clones of the surviving individual. Because of the same reasons,  $d$  converges to  $d'$ .

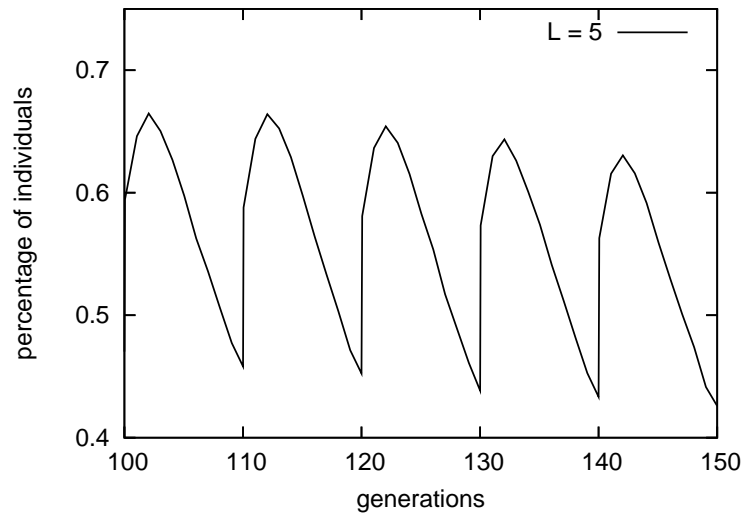
## 5.8 Mutation

Mutation’s influence on single-population EAs is fairly well understood (Spears, 1998). Mutation is an operator responsible for local changes to a given individual. It does play an important role in evolutionary algorithms, either as an exploratory operator in the absence of recombination, or as a source of genetic diversity. In fact, mutation is a source of new alleles that, if successful, spread in the whole system, and as such is probably one of the more important aspects of every evolutionary algorithm.

I think that in IMs mutation plays quite a similar role. Because it operates on a single individual regardless of where this individual is located, splitting individuals into islands does not change the way it operates. As a result, it is hard to predict a new significant role for this operator in IM. While providing new alleles is important, it fits better into incremental evolution, in which islands evolve solutions separately

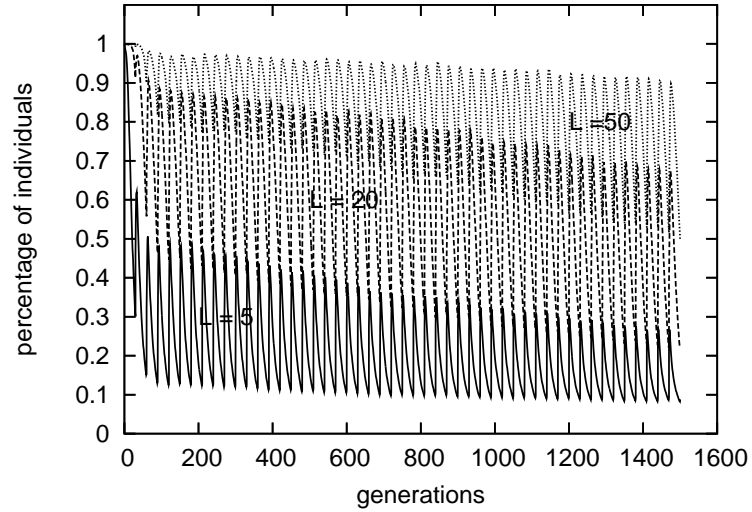


(a)  $\bar{m}(t)$

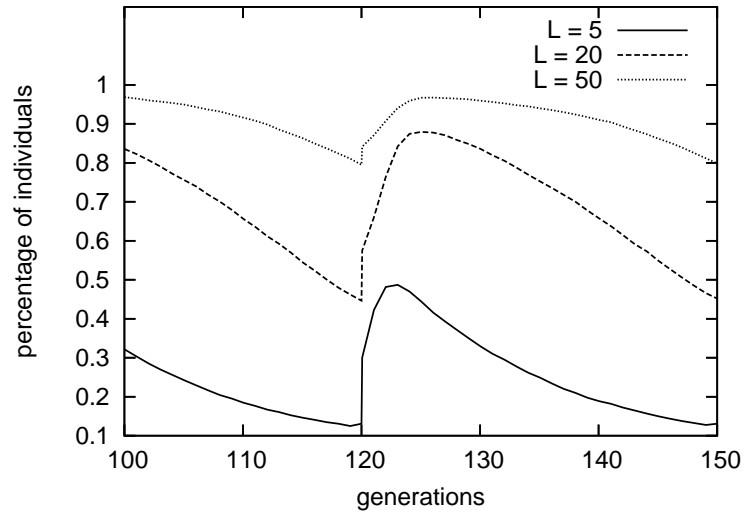


(b) close-up, operators interacting

Figure 5.20: Recombination increases the positive impact of migration on local diversity, but soon selection (or just stochastic sampling here) brings the diversity down, globally producing a slow drift. One-child stochastic recombination and unidirectional migrations, interval of 10.

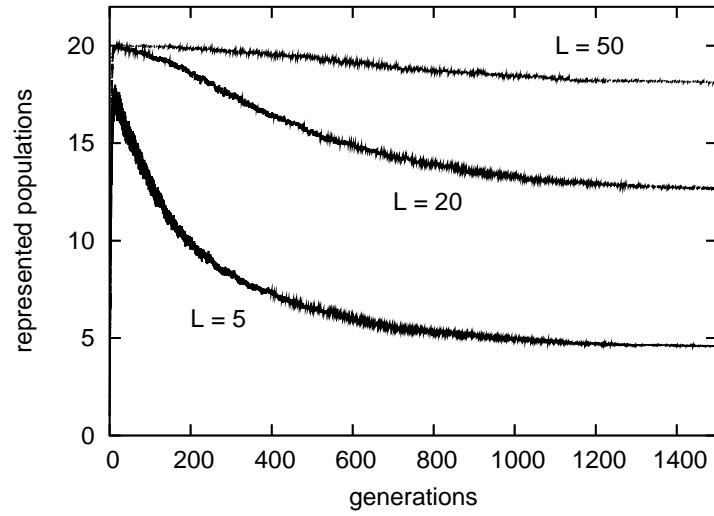


(c)  $\bar{m}(t)$

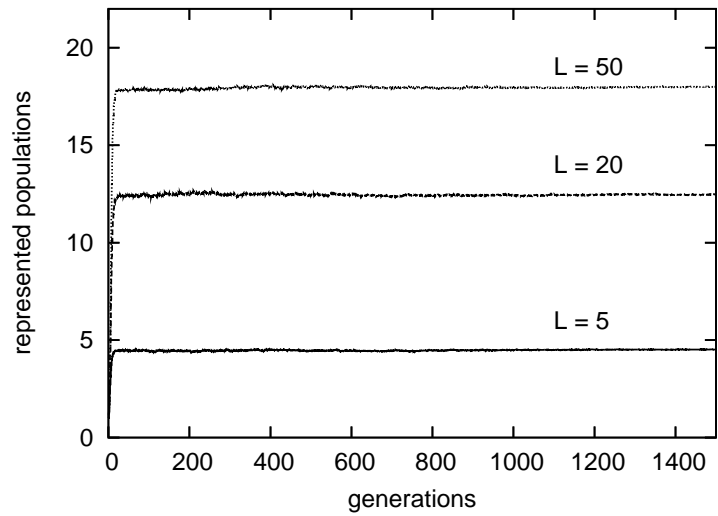


(d) close-up, operators interacting

Figure 5.20: Recombination increases the positive impact of migration on local diversity, but soon selection (or just stochastic sampling here) brings the diversity down, globally producing a slow drift. One-child stochastic recombination and unidirectional migrations, interval of 30.

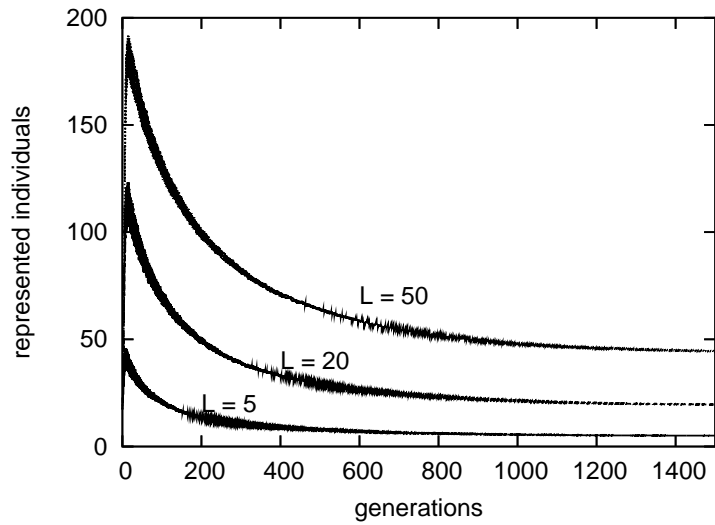


(a) *RPP*

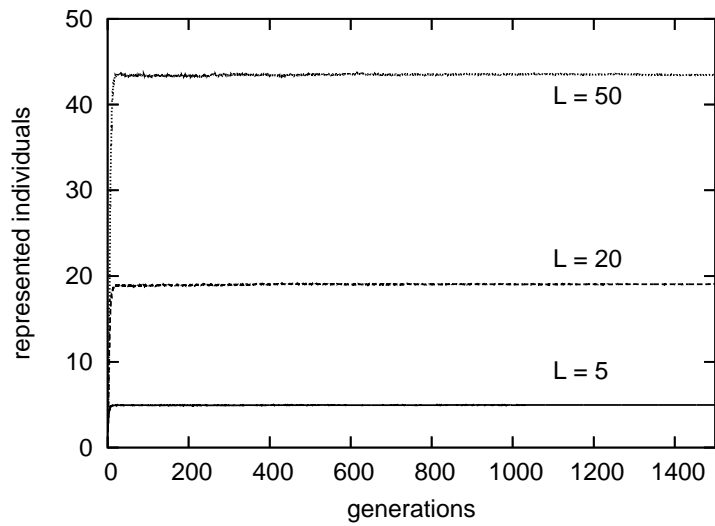


(b) *RPI*

Figure 5.21: Island-wide measurements converge to individual-wide measurements, but long after the mixing inside individuals took place. One-child stochastic recombination and unidirectional migrations, for different genome lengths.

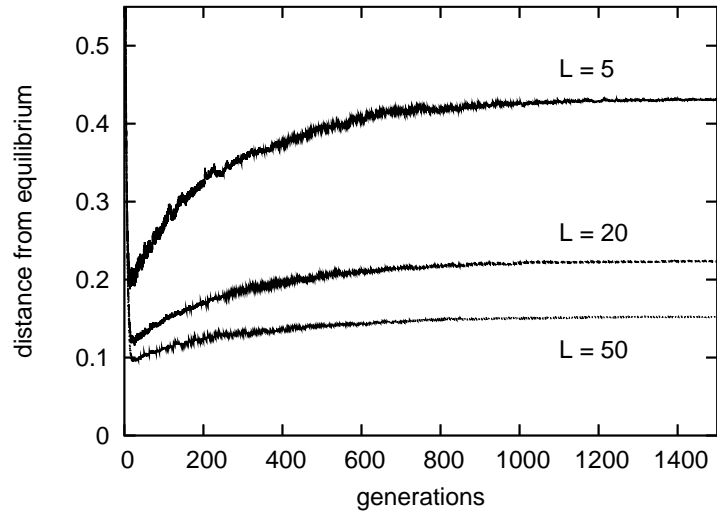


(c) *RIP*

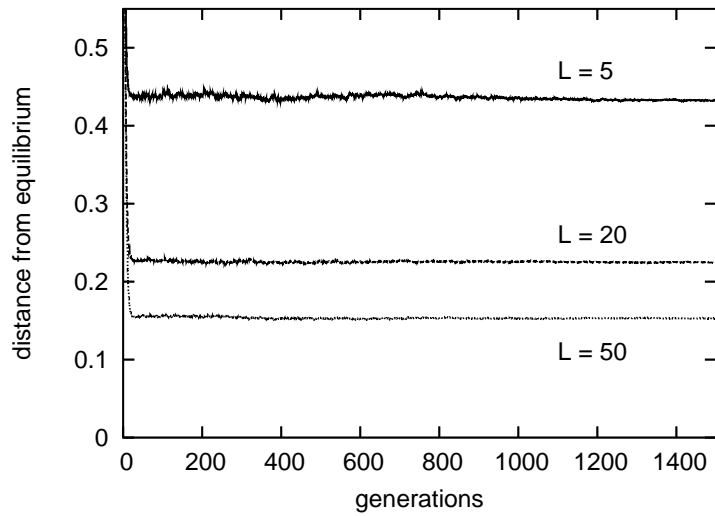


(d) *RII*

Figure 5.21: Island-wide measurements converge to individual-wide measurements, but long after the mixing inside individuals took place. One-child stochastic recombination and unidirectional migrations, for different genome lengths.



(a)  $d$



(b)  $d'$

Figure 5.22: Island distance from migration equilibrium converges to average individual distance from the equilibrium. One-child stochastic recombination and unidirectional migrations, for different genome lengths.



and compete. I am more interested in situations where islands frequently combine solutions (i.e. compositional evolution at the inter-island level), and mutation does not contribute to such a framework directly. Therefore, in this dissertation I focus mostly on recombination and I experiment little with mutation.

Several measurements used in analyzing other operators, would fail with mutation. The number of unique individuals, both globally and locally, would never converge and would always produce numbers close to the total number of individuals because individuals produced by mutation, even if similar to their parents, would still be different. The number of representatives, both at the level of individuals and genes would either yield the same results as previously, if I decided to count mutated genes as still originating in initial island populations, or would quickly drop to zero, if I decided to not count the mutated genes. A possible measurement is to measure the percentage of genes created (by mutation)  $k$  generations ago, for different  $k$ . Such a measurement would show whether mutations are successful and survive or not in the system — however, it does not seem to be influenced directly by migrations and as such, is not directly related to my research.

A few experiments I performed (not shown) suggest that in the case of stochastic migrations and one-child, stochastic recombination, even a very small amount of mutation helps keep the diversity high infinitely. With selection present, higher amounts of mutation would be required. One should, however, be careful, because a larger amount of mutation may remove any subtler effects of EAs, including the effect of recombination enhancing the impact of migrations (migration-recombination-selection scenario).

## 5.9 Summary

In this chapter I have analyzed the basic forces driving the dynamics of IMs. I have focused on three operators: migration, selection and recombination. While the role of the last two are quite well understood in traditional EAs, the role of the first is still vague. More importantly, I have studied relations between those operators.

We can understand the role of migration as a connection between the local and global levels of evolution. The impact of global evolution on local evolution is dual — both exploitative and explorative, as described below:

On one hand, migration serves as an additional selection pressure. In particular, in the absence of any other operator, migration can cause a convergence of the whole system to a single individual. Migration tends to act in two stages — in the beginning it makes all islands similar to each other, making the system converge more on a global level, rather than at an island level. Once this is achieved, in the second stage, migration slowly (stochastically) eliminates individuals until only one remains. A focus on global convergence is opposite to the way selection works. Selection brings islands into a converged state, but retains inter-island diversity, by acting locally and not interacting with other islands. Migration “transfers” local convergence to the inter-island level.

On the other hand, the selective role of migration is very weak and helps to keep the local diversity high. This was seen in both experiments with selection and with recombination. The way migration keeps the local diversity high is to utilize diversity at the inter-island level, where it works similarly to recombination. Even though it does not create new alleles, it mixes genetic material between islands, preventing

selection and drift as long as possible. The unidirectional versus bidirectional choice resembles a similar issue with recombination, where only producing two complementary children assures the preservation of the genetic material. As we know, recombination breaks the linkage between genes, so that they are not inherited together in surviving individuals. One could say that in a similar way, migration prevents solutions from being inherited within a single island in which they evolved.

The experiments with the recombination operator show that not only is migration a direct source of an instant diversity increase, but it also triggers longer-term exploration at the local level. Migration indirectly causes recombination to keep creating multiple new individuals for several generations after migration. This is a very promising observation because it shows that migration can cause a stabilized island to regain evolvability and possibly create novel solutions. In this *migration-recombination-selection scenario*, migration brings a set of new individuals, recombination mixes them with local individuals, producing multiple hybrids, and creates an opportunity for selection to choose good individuals and improve average fitness.

Summarizing, migration works both ways, transferring dynamics between the levels of evolution. Thanks to migration, what happens inside islands, i.e. local effects of selection or recombination, has global, inter-island implications. The impact of the local level on the global level is complex, and we will study it in Chapter 6.

## Chapter 6: Dynamics of after-migration evolution

In Chapter 4 we observed an increase in fitness, diversity and generally evolvability occurring *after* migrations. In this chapter I will try to understand what happens in a population at the gene level after a migration. We will see how these local behaviors correspond to interaction between islands at the global level.

In a traditional setup, interaction between islands is important *after* evolving solutions. In a setup which promotes inter-island compositional evolution (e.g. many small islands) proper interaction is necessary *during* the evolution of solutions.

To maximize the probability of creating novel solutions of higher fitness, for many complex domains one should be able to combine qualitatively different solutions. Because it is quite probable that in each island a different (sub)solution was evolved, it is important to measure how much islands and individuals from various islands are getting mixed with each other.

Recombination is a crucial operator to promote cooperation between islands. Without recombination the only way migrants can interact with locals is by competing for survival, and the better group dominates the other. An IM without recombination may only develop different solutions in different islands and then let the best ones spread and take over the others. An intuitive understanding of IMs suggest that their true power should be in the ability to discover various solutions or parts thereof in different islands, and create even better solutions as a result of placing them next to each other, and recombining them. I believe that such recombinations can increase

the evolvability in islands and build another level of solutions, which could in turn become successful migrants and seeds for further recombinations. In this chapter we will see models explaining how recombination may “propagate” to the inter-island level.

## 6.1 Allele survivability

The level of interaction between islands results from changes that migrations make to the contents of the target island, which is a result of interaction between migrants and locals. Migrants’ genes can either survive or get eliminated from the target island. When migrants’ alleles survive in the population, we will say that migrants are *accepted*. They could either dominate the population, or mix with the local gene pool. When migrants’ alleles are eliminated, we will say that migrants are *rejected*. Note, that even if migrants are accepted, their alleles may be overwritten with consecutive immigrations.

I will start with some fitness-related reasons for these situations to show that, for a successful mixing, the domain must be of a certain nature. There are at least two phenomena occurring after migration:

- If the average fitness of migrants is different than the average fitness in the target island, parent selection will favor either migrants or locals.
- If the average fitness of offspring with migrant genes is different than the average fitness of offspring with local genes, survival selection will favor one or the other.

The first situation may happen if the average fitness values of the source and target islands differ, or if we use a selective migration policy (e.g. using best–random policy

I expect that the average fitness of the migrants will be higher than the average fitness of the locals). However, even if migrants are preferred for recombination by parent selection, their alleles have a small chance of survival if their offspring have a bad fitness value — which is stated in the second bullet. Therefore, good recombination and survival selection is very important as well.

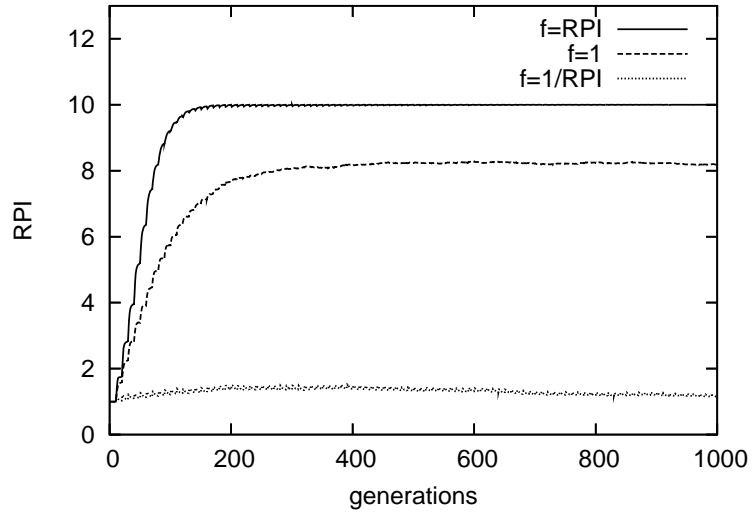
I analyze an artificial model, in which fitness is assigned not based on the alleles, but based on the origin of the alleles. Let us consider three simple cases. In the first case the fitness of an individual is implemented as equal to  $RPI$ . Therefore, migrations are encouraged because they carry gene values of different origins into a population. In the second case, the fitness of an individual is computed as  $1/RPI$ . In this case, individuals with alleles of the same origin have the highest fitness, and therefore migrations are discouraged. For reference I compare these two cases with the third, where migrants' allele survivability is random (fitness is constant for all individuals).

Not surprisingly, when we plot the average  $RPI$  obtained in these three cases, we see that when migrants are accepted,  $RPI$  converges toward  $N$ ; when migrants are rejected, the  $RPI$  stays around one, and in the random case (as already seen in the previous chapter), the  $RPI$  slowly grows and will stabilize at some level smaller than  $N$ . This is observed for both EA-1<sup>1</sup> (Fig. 6.1a) and EA-2 (Fig. 6.1b).

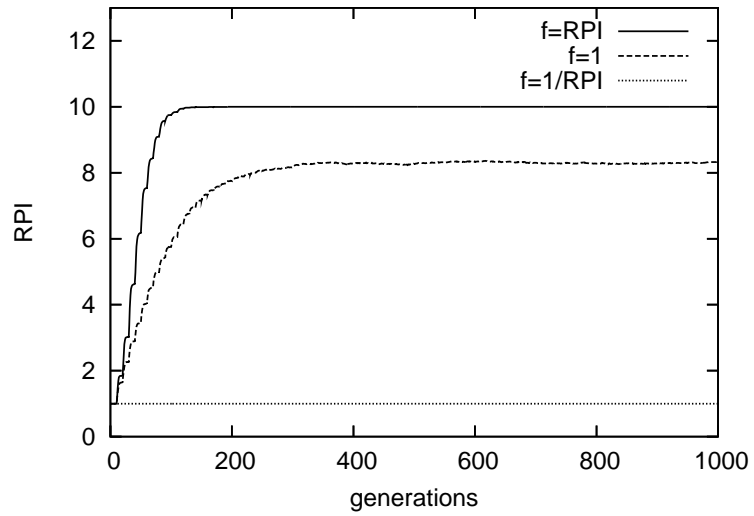
If all migrants are rejected (i.e. their genes would not spread in the target islands), the migrations should hardly affect the target islands. Migrants would then temporarily take the place of some other individuals (usually either random, or worst ones) and the local diversity would be decreased and drift created. However, the

---

<sup>1</sup>See Chapter 3 for description of EA-1 and EA-2 setups. Simplifying, EA-1 has a weaker selection policy.



(a) EA-1



(b) EA-2

Figure 6.1: The number of represented populations in an individual ( $RPI$ ), when migrants are accepted, neutral or rejected.

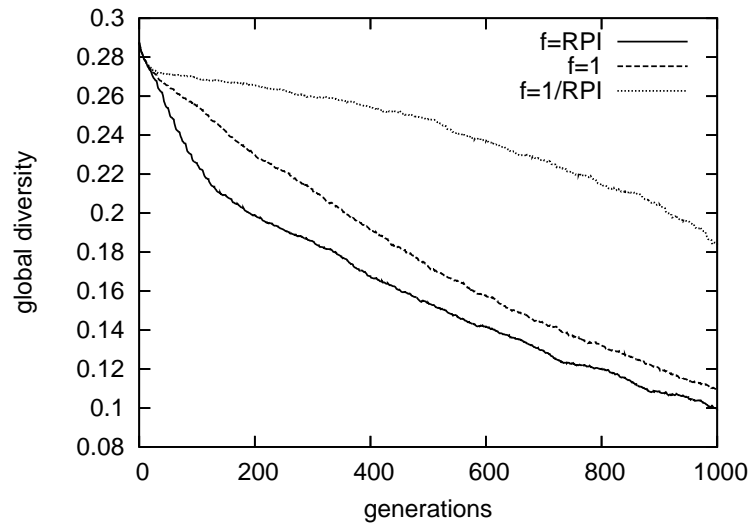
global diversity would stay high, as islands would remain focused on their regions of the search space. Looking at diversity plots (Fig. 6.2a and Fig. 6.2b), we see that when migrants are rejected, the system's diversity is generally higher. In case of EA-2, we observe a full rejection of migrants.

It is important to realize that it is not only migration parameters that decide the interaction between islands. Fitness landscape, EA, representation and operators used in islands have significant impact on the mixing of islands as well. In other words, the same migration parameters may result in very different behavior from case to case. In particular, even big and frequent migrations may have little effect on evolution if migrants are rejected. The system should then behave similarly to an isolated model.

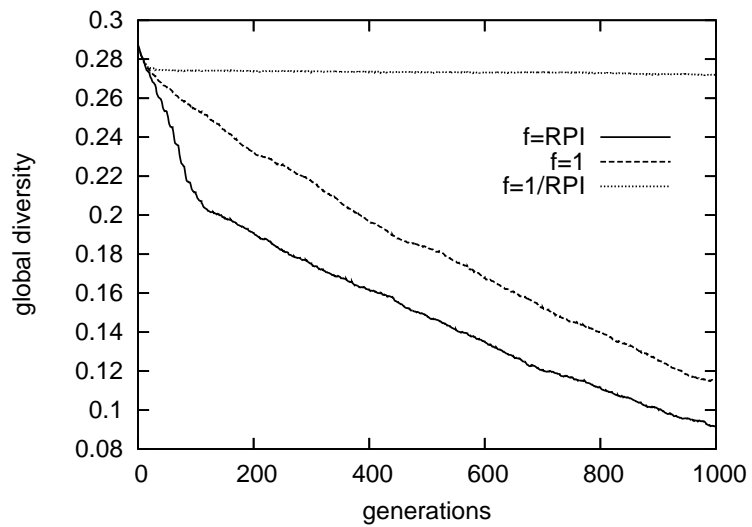
## 6.2 After-migration dynamics models

In Chapter 5 we saw the migration-recombination-selection scenario, in which recombination is able to temporarily create a “rich” mixture of individuals using alleles inserted by migration and the best individuals created during these periods of change can be then selected. Therefore, we can rightfully expect, that depending on the constructive ability of recombination, migrants will either be successfully mixed with locals (provided that both have good fitness), or the two types of individuals will remain separated, until one of them dominates. I will show how a proper recombination operator is crucial for the acceptance of migrants' alleles, and therefore for successful combination of subsolutions in IMs. I show that there are only a few possible types of behavior in islands after migration, and that recombination is one of the most





(a) EA-1



(b) EA-2

Figure 6.2: Global diversity when migrants are accepted, neutral or rejected.

important operators to choose from. For reference, in Appendix D, I review some important general aspects of recombination.

According to the building blocks hypothesis, a success of genetic algorithms (and other EAs using recombination) is due to good BBs being combined into larger ones. Good BBs are those that on average contribute to a good fitness of an individual. Linked genes form building blocks (BB), and proper BBs are composed together to form blocks at the next level. Several issues are related to the survival of BBs (Cantú-Paz, 1999; Sastry and Goldberg, 2002). First they need to be created (*BB supply*). In IMs this should be done both in a standard evolutionary way (by EAs inside each island), and by proper fixing of genes, as mentioned before. Then proper BBs must be chosen if multiple choices are possible for a given subset of genes (*BB decision*). Finally, once BBs are present in individuals of a given population, they must be properly combined together (*BB mixing*).

The migration-recombination-selection scenario was relatively easy to observe, because so far I have assumed that recombining individuals is an easy task, provided a supply of correct sub-solutions. In Chapter 4 this was achieved by using real-valued genes, which could not be damaged by recombination (I used BBs of length one). In Chapter 5, a flat fitness landscape was used, so there could be no discussion of good BBs.

What happens when recombination is more destructive than constructive? The longer the BBs become, the more likely they are to break when recombining individuals. For one-point or two-point crossovers, genes having a closer position are more likely to survive together, but the distant ones are much more prone to separation. For a uniform recombination the situation is even worse, since gene

positions play no role, and every two genes are equally likely to be separated. So, how can long BBs survive? And can migrants successfully mix with locals?

In this section I focus on BB mixing. Not only will I analyze what happens with genomes after migration, but also how an increased length of BBs (which decreases the ability of recombination to create good offspring) affects the dynamics. To show this, I mostly use uniform recombination, which has several favorable properties. It is easy to analyze, regardless of its randomness it still requires at least a few generations to mix alleles in a population and it does not facilitate BB heritability, so it is a good example of a case when we do not know the BBs in our problem, yet we want to use some arbitrary recombination.

I will use special models, that I call  $\alpha\beta\gamma\delta$  models, and which I will gradually introduce in the following subsections.

### 6.2.1 Building blocks competing

In Chapter 4 we have seen benefits of using a framework, in which islands are small and converge fast. Let us take such framework, and assume that a migration consists of some copies of the same individual, and similarly the target island is also converged to a single individual. Thanks to prior evolution these individuals may carry good BBs. Further, let us assume that we have some weak selection EA, for example an EA-1 (binary tournament selection) and no mutation for simplicity. Finally, let us denote by  $\alpha_0$  the percentage of better individuals (*alpha* individuals, or “alphas”) after migration, regardless if these are migrants or locals,<sup>2</sup> and by  $\beta_0 = 1 - \alpha_0$  the percentage of the worse individuals (*beta* individuals, or “betas”). This corresponds

---

<sup>2</sup>Although in later sections we will mostly think of a migration of better individuals.

to either a migration of  $\alpha_0 M$  individuals better than those in the target island, or a migration of  $\beta_0 M$  worse individuals ( $M$  being the size of the target island). Evolution will change these proportions, and I will refer to them by  $\alpha_t$  and  $\beta_t$  after generation  $t$  (or dropping the index, when the context is obvious).

As we know from the standard takeover analysis,  $\alpha$  will increase in time, converging toward the value of one (and reaching it quite quickly for finite populations). For binary tournament selection, this growth is given by the following equation

$$\alpha_{t+1} = 1 - (1 - \alpha_t)^2 = \alpha(2 - \alpha).$$

The situation looks slightly different if the effect of recombination is included. Let us assume that a recombination of alpha and beta individuals results in breaking their BBs, and therefore the offspring, *gamma*, has an even lower fitness than any of the parents. This is quite a realistic situation, especially if alphas and betas evolved independently. I will denote the percentage of gamma individuals by  $\gamma$ . The state of the system may be described either by giving an  $(\alpha, \beta)$  pair constrained by  $0 \leq \alpha, \beta \leq 1$  and  $\alpha + \beta \leq 1$ , or by an  $(\alpha, \beta, \gamma)$  triple being on a standard 2-simplex.<sup>3</sup>

Let us denote the length of individuals by  $L$ . A short note is necessary. If parents share some alleles, recombination leaves them untouched. For uniform recombination with no loss of generality we can disregard such genes, and assume that all genes in *alphas* and *betas* are different. In IMs, fixation and convergence will cause more and more of such alleles to be shared (although in IMs convergence oscillates locally), and

---

<sup>3</sup>An n-simplex is a set  $\{x_1, \dots, x_{n+1} : \sum_{i=1}^{n+1} x_i = 1 \text{ and } 0 \leq x_i \leq 1\}$ .

so  $L$  should effectively decrease during evolution, which may affect recombination. For this analysis, however, I assume a fixed  $L$ .

The probability of a uniform crossover creating a gamma individual is  $p_{\alpha,\beta\rightarrow\gamma} = 1 - 2(\frac{1}{2})^L$  (which equals all offspring except the two clones of the parents). For  $L = 1$  we have  $p_{\alpha,\beta\rightarrow\gamma} = 0$ , since no mixing is possible, no gammas will be created and the traditional analysis of takeover applies. For  $L = \infty$  we have  $p_{\alpha,\beta\rightarrow\gamma} = 1$ , and every offspring of alpha and beta is of type gamma. Also the probability of creating an alpha or a beta individual from two gamma type individuals is 0. In such cases the offspring will be of the alpha type only if both parents are of the alpha type. Therefore, we have the following equation for binary tournament selection:

$$\alpha_{t+1} = (1 - (1 - \alpha_t)^2)^2 = \alpha_t^2(2 - \alpha_t)^2,$$

where the subscript  $t$  denotes a generation number. In the more general situations, when recombination occurs with probability  $\rho$  (and otherwise we clone individuals),  $\alpha_{t+1}$  percentage is given by:

$$\alpha_{t+1} = (1 - \rho)\alpha_t(2 - \alpha_t) + \rho\alpha_t^2(2 - \alpha_t)^2.$$

This function is shown in Fig. 6.3a for various values of  $\rho$ , and is compared with identity function. For  $\rho = 0$  it simplifies to cases without recombination. As we see, for larger values of  $\rho$  there exist a critical point  $\lambda(\rho)$  such that if  $\alpha_0 < \lambda(\rho)$ , then the percentage of alphas will decrease in consecutive generations, and will ultimately converge to zero. This means that the best individuals will actually *not* survive in the population.

We can find the  $\lambda(\rho)$  point analytically by solving:

$$(1 - \rho)\alpha(2 - \alpha) + \rho\alpha^2(2 - \alpha)^2 = \alpha.$$

After expanding and factoring, the left hand side is equal to  $\alpha(\alpha - 1)(\rho\alpha^2 - 3\rho\alpha + 2\rho - 1)$ ; comparing the last component to 0, and solving the quadratic equation we find that  $\Delta = \rho(4 + \rho)$ , so solutions exist for all  $\rho \geq 0$ , and they are given by  $1.5 \pm \frac{\sqrt{\rho(4+\rho)}}{2\rho}$ . The smaller one is the  $\lambda(\rho)$  value. In particular, for  $\rho = 1$  we have  $\lambda(1) = \frac{3-\sqrt{5}}{2} \approx 0.382$ . The lambda function is shown in Fig. 6.3b. For  $\rho < 0.5$  we have  $\lambda(\rho) < 0$ , which means that with less than a 0.5 rate of recombination alpha individuals will always dominate the population.

Beta individuals also will survive only if both parents are of beta type. Gamma individuals obviously make up the rest of the population. The formula to describe this is as follows:

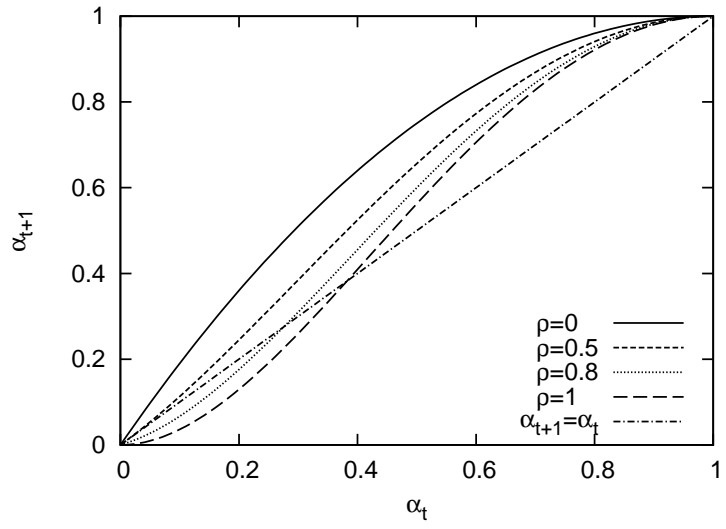
$$\beta_{t+1} = (\beta_t^2 + 2\beta_t\gamma_t)^2 = (\beta_t^2 + 2\beta_t(1 - \alpha_t - \beta_t))^2 = \beta_t^2(2 - 2\alpha_t - \beta_t)^2,$$

$$\gamma_{t+1} = 1 - \alpha_{t+1} - \beta_{t+1}.$$

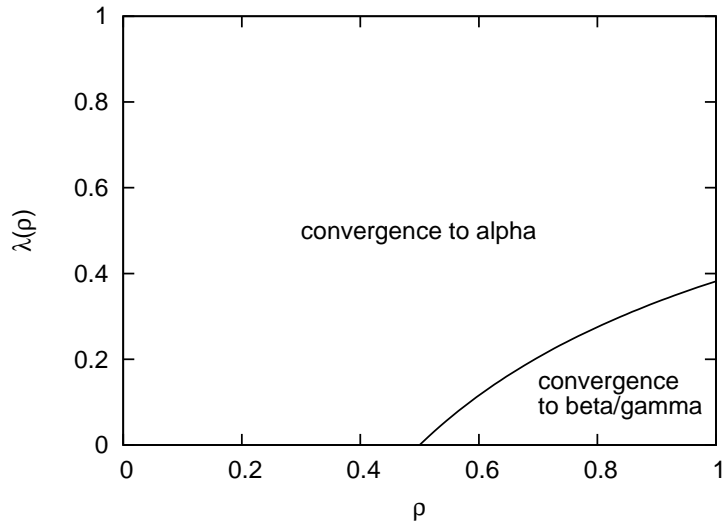
These formulas can be applied repeatedly, creating the first  $\alpha\beta\gamma$  model which approximates the percentages of migrants, locals and hybrids in a target island. The model predicts three possible outcomes, shown in Fig. 6.4. If  $\alpha_0$  is very small,<sup>4</sup> beta individuals will take over. If  $\alpha_0$  is of medium size, a lot of gamma individuals will

---

<sup>4</sup>The limit was experimentally measured to be around 0.17. It could be approximated by finding a boundary in initial conditions of a proper differential equation.



(a)  $\alpha_{t+1}$  for different  $\rho$ , and  $L = \infty$



(b)  $\lambda(\rho)$ , a critical point for  $\alpha_0$

Figure 6.3: Convergence to alpha individuals.

be created as an effect of recombination of alphas and betas. This will lead to the ultimate domination of gamma individuals (and the decrease of the average fitness!). Finally, if  $\alpha_0$  is sufficiently large (greater than  $\frac{3-\sqrt{5}}{2}$ ), then alpha individuals will dominate the population. A corresponding vector field, with the trajectories of the three cases, is shown in Fig. 6.5.

When  $1 < L < \infty$ , the interaction between alpha, beta and gamma type individuals becomes slightly more complicated. Let us assume that each gene of a gamma type individual has a probability of 0.5 of being equal to an alpha allele, and the same probability of being equal to appropriate beta allele. Let us use two auxiliary constants

$$q_2 = \left(\frac{1}{2}\right)^L$$

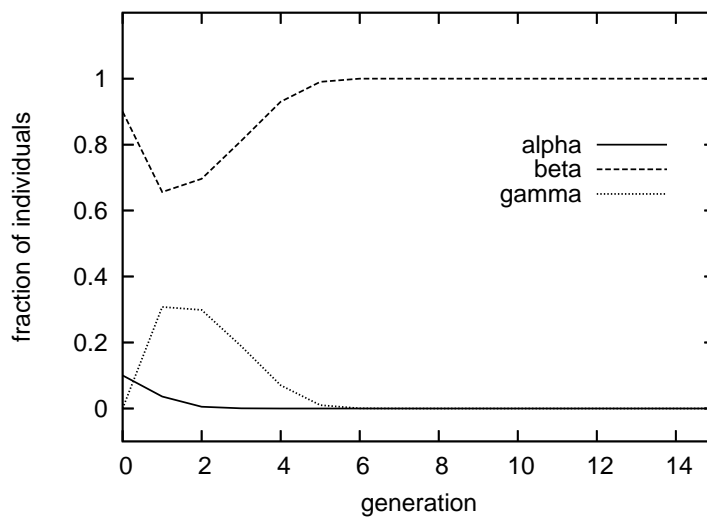
$$q_3 = \left(\frac{3}{4}\right)^L .$$

The probabilities of creating an alpha individual by recombining two other individuals can be summarized in the following symmetrical matrix  $A$ , where both rows and columns are sorted in alpha, beta, gamma order. I denote by  $p_{x,y \rightarrow z}$  the probability of creating  $z$  by recombining  $x$  and  $y$ , and  $A$  is written as:

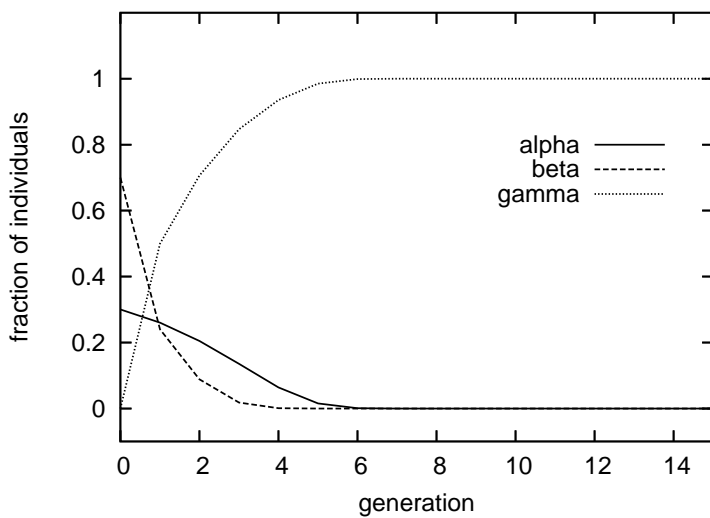
$$A = \begin{bmatrix} p_{\alpha,\alpha \rightarrow \alpha} & p_{\alpha,\beta \rightarrow \alpha} & p_{\alpha,\gamma \rightarrow \alpha} \\ p_{\beta,\alpha \rightarrow \alpha} & p_{\beta,\beta \rightarrow \alpha} & p_{\beta,\gamma \rightarrow \alpha} \\ p_{\gamma,\alpha \rightarrow \alpha} & p_{\gamma,\beta \rightarrow \alpha} & p_{\gamma,\gamma \rightarrow \alpha} \end{bmatrix} = \begin{bmatrix} 1 & q_2 & q_3 \\ q_2 & 0 & 0 \\ q_3 & 0 & q_2 \end{bmatrix} .$$

For example, if an alpha individual is recombined with a gamma individual, half



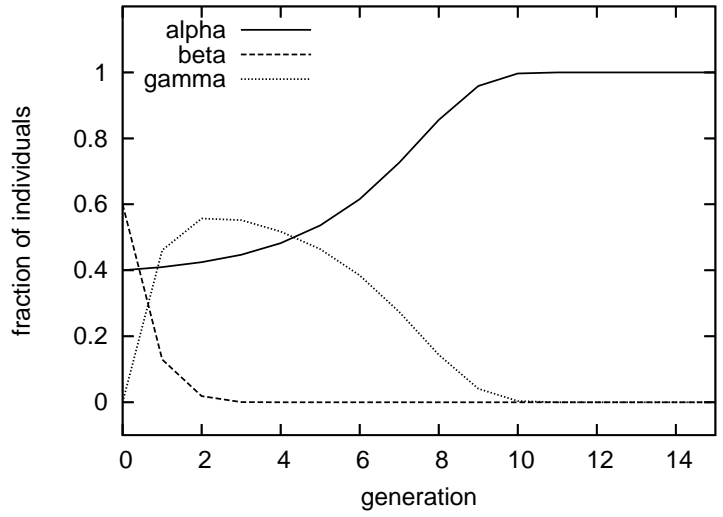


(a)  $\alpha_0 = 0.1$ , *beta* individuals taking over



(b)  $\alpha_0 = 0.3$ , *gamma* individuals taking over

Figure 6.4: Different  $\alpha_0$  (percentage of best individuals) lead to the domination of different individuals. Results from the model for  $L = \infty$ .



(c)  $\alpha_0 = 0.4$ , *alpha* individuals taking over

Figure 6.4: Different  $\alpha_0$  (percentage of best individuals) lead to the domination of different individuals. Results from the model for  $L = \infty$ .

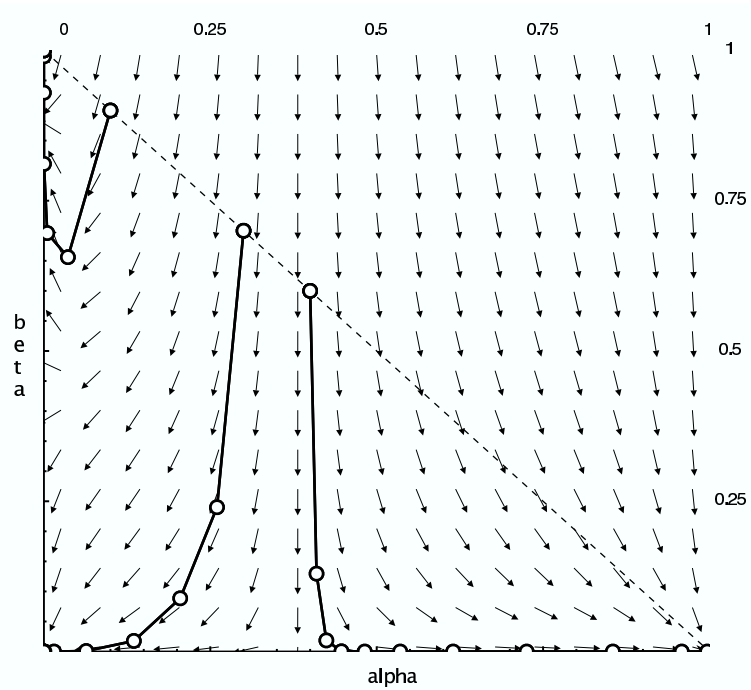


Figure 6.5: The  $\alpha$ - $\beta$  vector field, with paths of the three cases converging to different individuals. Results from the model for  $L = \infty$ .

of the time the alpha gene is chosen, and other half of the time there is a probability of 0.5 for choosing the alpha gene. Hence  $p_{\alpha,\gamma\rightarrow\alpha} = (\frac{1}{2} + \frac{1}{2}\frac{1}{2})^L = (\frac{3}{4})^L$ . On the other hand  $p_{\beta,\gamma\rightarrow\alpha} = 0$  because a gamma individual must have some beta genes, and they will survive when recombining with beta.

Analogically, we can write the  $B$  matrix to denote the probabilities of creating a beta individual as follows:

$$B = \begin{bmatrix} p_{\alpha,\alpha\rightarrow\beta} & p_{\alpha,\beta\rightarrow\beta} & p_{\alpha,\gamma\rightarrow\beta} \\ p_{\beta,\alpha\rightarrow\beta} & p_{\beta,\beta\rightarrow\beta} & p_{\beta,\gamma\rightarrow\beta} \\ p_{\gamma,\alpha\rightarrow\beta} & p_{\gamma,\beta\rightarrow\beta} & p_{\gamma,\gamma\rightarrow\beta} \end{bmatrix} = \begin{bmatrix} 0 & q_2 & 0 \\ q_2 & 1 & q_3 \\ 0 & q_3 & q_2 \end{bmatrix} .$$

The probabilities of choosing respectively alpha, beta and gamma (knowing that the fitness of alpha is larger than the fitness of beta, which in turn is larger than gamma's fitness) as a parent in a binary tournament are:

$$p_{\alpha} = 1 - (1 - \alpha)^2 = \alpha(2 - \alpha) ,$$

$$p_{\beta} = 1 - (1 - \alpha - \beta)^2 - p_{\alpha} = \beta(2 - 2\alpha - \beta) ,$$

$$p_{\gamma} = \gamma^2 = (1 - \alpha - \beta)^2 .$$

If we denote  $p = (p_{\alpha}, p_{\beta}, p_{\gamma})$ , we can write:

$$\alpha_{t+1} = p_t A p_t^T ,$$

$$\beta_{t+1} = p_t B p_t^T .$$

We could create a similar (although somehow more complicated) matrix for gamma, but we know that  $\gamma_{t+1} = 1 - \alpha_{t+1} - \beta_{t+1}$ .

Having this model, we can now create a convergence map showing what type of individual the system will converge to, for various values of  $\alpha_0$  and  $L$ . Whereas, it is difficult to analytically compute the borders of convergence regions, we can run the model for various combinations to get a general view. I performed such experiments and changed  $\alpha_0$  from 0.05 to 0.95, with a step of 0.05, and  $L$  from 1 to 25, with a step of 1. This results in a *convergence map*, shown in Fig. 6.6.

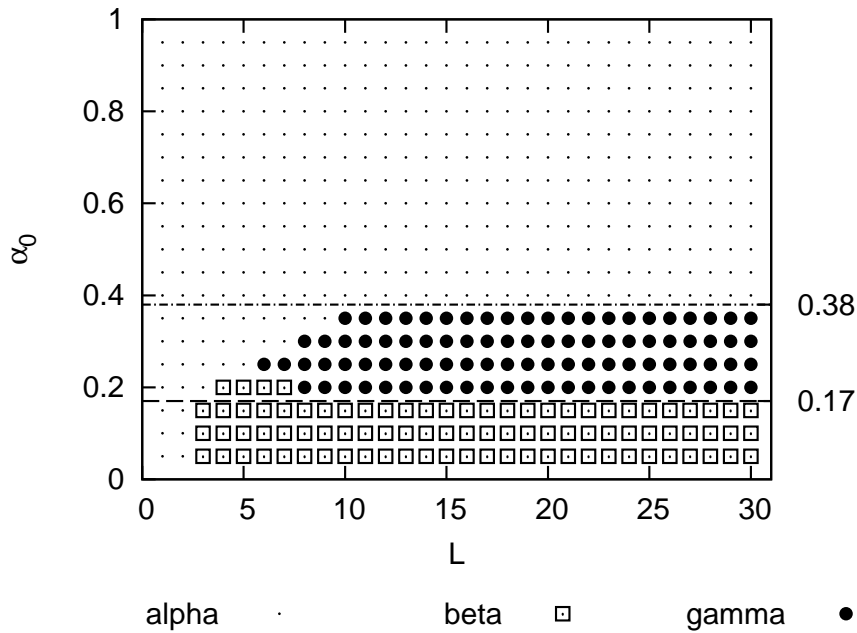


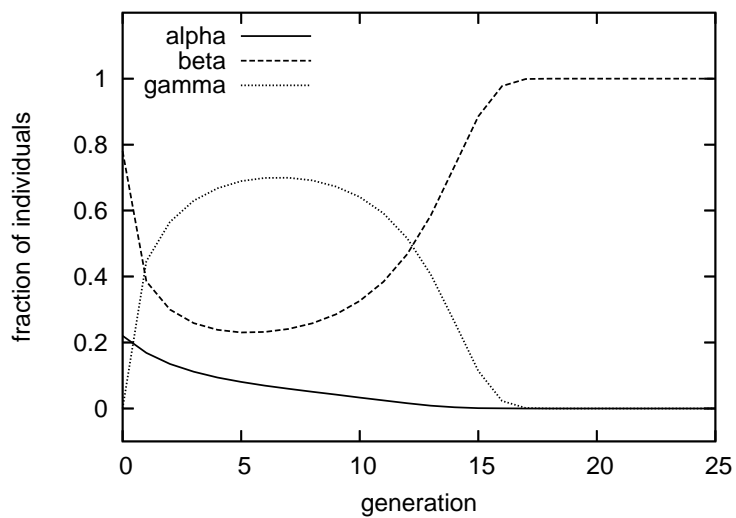
Figure 6.6: Convergence map for the model with  $\alpha=10$ ,  $\beta=01$ ,  $\gamma=**$ .

We see that for  $L \leq 5$  gamma individuals never dominate (at least for the parameter values that I checked). When individuals are so short, the probability of randomly creating alpha, or beta individuals using gamma individuals is high enough to counteract the growth of  $\gamma$ . A model of two such runs with very similar  $\alpha_0$  values,

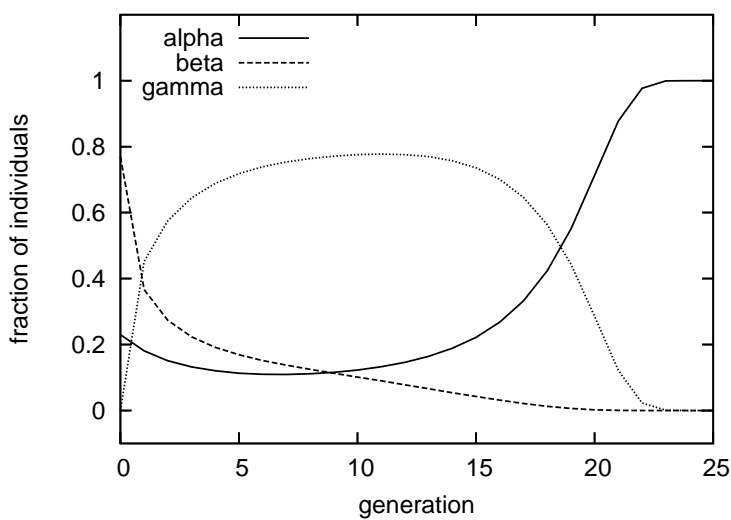
yet with different outcome is shown in Fig. 6.7. We can see how gammas decrease in number, even though in both cases they start to dominate in the beginning.

To verify the model, I have performed some simple experiments with a single population of size 100, starting from a mixture of alpha and beta individuals, as it is after migration (I will refer to this and further similar experiments as a *simulation*, because it is not a full IM setup). I have used a binary encoding of a length  $L$  and considered BBs of length equal to  $L/2$ . A remark about notation is necessary here. To denote strings of a BB's length ( $L/2$  in our case) of 1s, 0s, or random genes, I use the following notation with the bold font:  $\mathbf{1}=1^{L/2}$ ,  $\mathbf{0}=0^{L/2}$  and  $\mathbf{*}={*}^{L/2}$ . I assume that alpha is  $\mathbf{10}$  and beta is  $\mathbf{01}$ . The important thing is that alpha and beta individuals have different alleles for each gene. Any individual different from alpha or beta was considered a gamma individual. I denote it by  $\mathbf{**}$ , which notation I will use to denote “every combination of alleles except for the ones used in other types of individuals.” Nevertheless, I approximate that each gene in the  $\mathbf{*}$  part has equal probability of being “0” or “1.” This is a small inconsistency, and in particular from this assumption one could compute a non-zero probability for a combination that is equal to alpha or beta. This error, however, is very small for a larger  $L$ , in which we are primarily interested. In case of an odd  $L$ , I assume the first part of length  $\frac{L+1}{2}$  and the second of length  $\frac{L-1}{2}$ .

For each configuration, the EA-1 algorithm without mutation was run 100 times for 500 generations. Again as expected, I observed convergence to all individual types (alpha, beta and gamma). However, it is hard to define exact borders in the parameters space, with regard to the type of individual the system converges to. Rather, the borders are imprecise, because for parameters in between the regions the



(a)  $\alpha_0 = 0.22$ , *beta* individuals taking over



(b)  $\alpha_0 = 0.23$ , *alpha* individuals taking over

Figure 6.7: Short genomes prevent gamma individuals from dominating. Results from the model for  $L = 5$ .

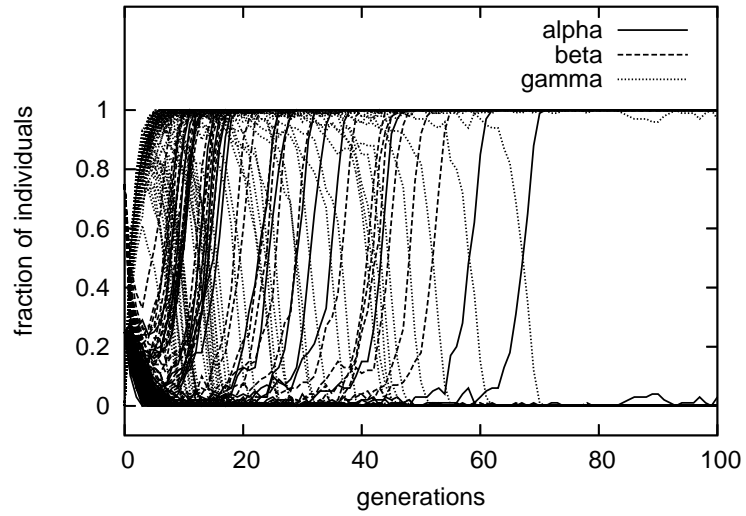
system may produce different outcomes each time. This is shown for  $\alpha_0 = 0.25$  and  $L = 12$  in Fig. 6.8. The cases of convergence to more than one individual type occur only on the borders between regions and do not change the overall picture.

I have decided that if the system converges to a given individual more than 50% of the time, I mark this individual as the outcome for given settings of parameters. If no individual reached 50%, I left the outcome blank. A convergence map is shown in Fig. 6.9. As we can see, gamma individuals take over less often, than in the prediction model. However, the general shape of the map is similar. For low  $\alpha_0$ , beta individuals take over, for big  $\alpha_0$ , alpha individuals take over, and for larger  $L$  ( $L > 14$ ), an intermediate interval exists for  $\alpha_0$  and this is when gamma individuals are dominating. A longer  $L$  is important because it represents more complex problems. Note that turning on mutation would give even higher chances to gamma individuals for dominating the population, since it would easily mutate alpha and beta individuals into gamma ones.

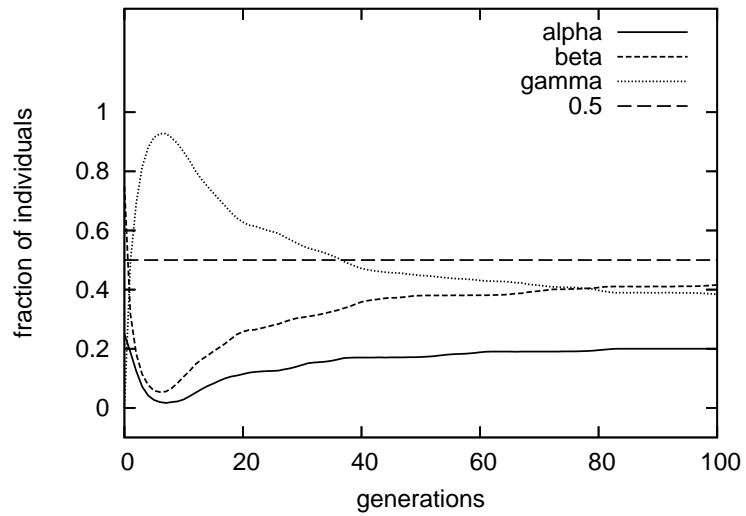
Although the above model and simulation was created for binary tournament selection, I suppose that any weak selection that allows selection of the worst (gamma) individuals would result in a similar picture. Even elitism, which keeps the best individual (an alpha individual) discovered throughout the run would not necessarily lead to the ultimate domination by alpha individuals — because as we have seen small  $\alpha$ s are not strong enough to take over the population.

## 6.2.2 Combining building blocks

We have just seen how individuals compete for survival after migrations. In most cases, either copies of one or copies of the other, will take over the population, and so



(a) All 100 runs



(b) Average

Figure 6.8: Different individuals dominate the population in different runs, for  $\alpha_0 = 0.25$  and  $L = 12$ . Results from a simulation.



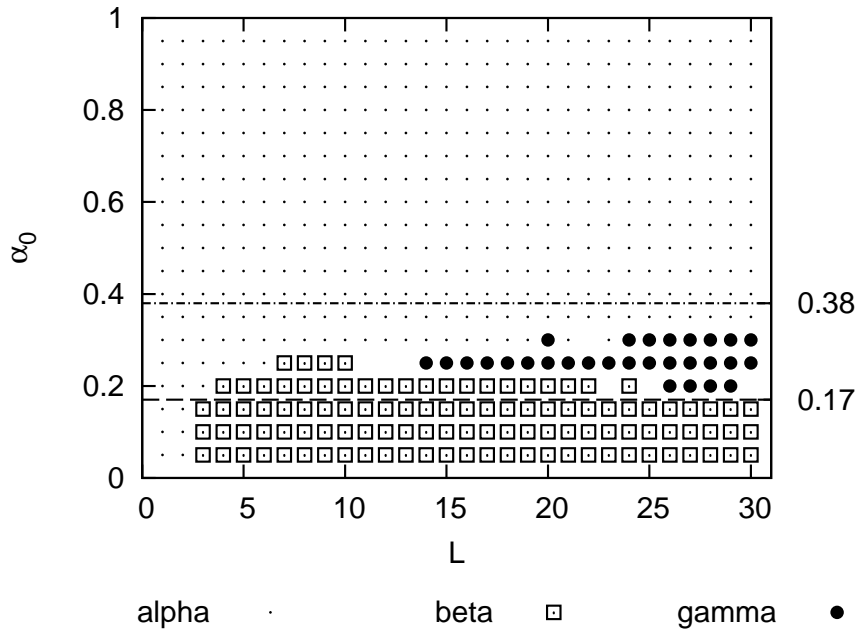


Figure 6.9: Convergence map for a simulation with  $\alpha=10$ ,  $\beta=01$ ,  $\gamma=**$ .

the BBs contained within the “winning” individual will survive. We have also seen that for some intermediate settings of  $\alpha_0$ , individuals having some genes from both alpha and beta individuals may “win,” even if fitness-wise they are the worst. In such cases, most BBs in both individuals would fail to survive since it would be broken by recombination.

In this section I will study how individuals can cooperate after migrations, by successfully combining their BBs. Indeed, not all recombinations of alpha and beta individuals must be worse than their parents. Both groups carry different potentially good BBs. For many hierarchical problems a special combination of some BBs creates individuals that have better fitness than their parents.

It is very important to know how likely recombination is to create successful combinations. To show this, let us perform an analysis similar to the one in the

subsection 6.2.1. Let us assume that apart from alpha, beta and gamma individuals, there exists a fourth individual, *delta*, with a genome that is a special combination of alpha and beta genomes (so it would have been counted as a gamma previously). I assume the delta of form **11**, and appropriately extend the meaning of **\*\*** to exclude delta. The delta individual benefits from recombining building blocks (**1s**) from its parents, and therefore has a higher fitness. Again, let us assume that we start from a population consisting in  $\alpha_0$  part of “alphas,” and  $\beta_0 = 1 - \alpha_0$  part of “betas.” Will the delta individual take over the population, and if so, for which configurations?

For  $L = 1$  only alpha and beta individuals exist, and the standard analysis applies. For  $L = \infty$  the analysis is analogical to the situation without the delta, since the probability of creating the delta is zero. For  $2 \leq L < \infty$ , I will proceed in a manner similar to the previous analysis for competing BBs.

Alpha, beta, gamma and delta individuals are defined by the presence or absence of proper BBs. Probabilities of creating a given individual type from the others are multiplications of appropriate probabilities for their BBs. I use the following notation:<sup>5</sup>

$$q'_1 = \left(\frac{1}{4}\right)^{L/2},$$

$$q'_2 = \left(\frac{1}{2}\right)^{L/2},$$

$$q'_3 = \left(\frac{3}{4}\right)^{L/2}.$$

---

<sup>5</sup>The following holds:  $(q'_2)^2 = q_2$  and  $(q'_3)^2 = q_3$ .

This notation facilitates writing probabilities, for example:

$$p_{\alpha,\gamma\rightarrow\alpha} = p_{\mathbf{10},**\rightarrow\mathbf{10}} = p_{\mathbf{1},*\rightarrow\mathbf{1}}p_{\mathbf{0},*\rightarrow\mathbf{0}} = (q'_3)^2.$$

Let us define the three matrices A, B and D, shown below.

$$A = \begin{bmatrix} 1 & (q'_2)^2 & (q'_3)^2 & q'_2 \\ (q'_2)^2 & 0 & 0 & 0 \\ (q'_3)^2 & 0 & (q'_2)^2 & q'_3q'_1 \\ q'_2 & 0 & q'_3q'_1 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 & (q'_2)^2 & 0 & 0 \\ (q'_2)^2 & 1 & (q'_3)^2 & q'_2 \\ 0 & (q'_3)^2 & (q'_2)^2 & q'_3q'_1 \\ 0 & q'_2 & q'_3q'_1 & 0 \end{bmatrix},$$

$$D = \begin{bmatrix} 0 & (q'_2)^2 & q'_3q'_1 & q'_2 \\ (q'_2)^2 & 0 & q'_3q'_1 & q'_2 \\ q'_3q'_1 & q'_3q'_1 & (q'_2)^2 & (q'_3)^2 \\ q'_2 & q'_2 & (q'_3)^2 & 1 \end{bmatrix}.$$

A state of the system (current percentages of individual types) can be described by a quadruple  $(\alpha, \beta, \gamma, \delta)$  belonging to the standard 3-simplex. The probabilities of choosing respective individuals for a parent (with binary tournament) are now:

$$p_\delta = 1 - (1 - \delta)^2 = \delta(2 - \delta),$$

$$p_\alpha = 1 - (1 - \delta - \alpha)^2 - p_\delta = \alpha(2 - 2\delta - \alpha),$$

$$p_\beta = 1 - \gamma^2 - p_\delta - p_\alpha = \beta(2 - 2\delta - 2\alpha - \beta),$$

$$p_\gamma = \gamma^2 = (1 - \delta - \alpha - \beta)^2.$$

If we denote by  $p = (p_\alpha, p_\beta, p_\delta, p_\gamma)$ , then the expected percentage of individuals in generation  $t + 1$  can be computed by

$$\alpha_{t+1} = p_t A p_t^T,$$

$$\beta_{t+1} = p_t B p_t^T,$$

$$\delta_{t+1} = p_t D p_t^T,$$

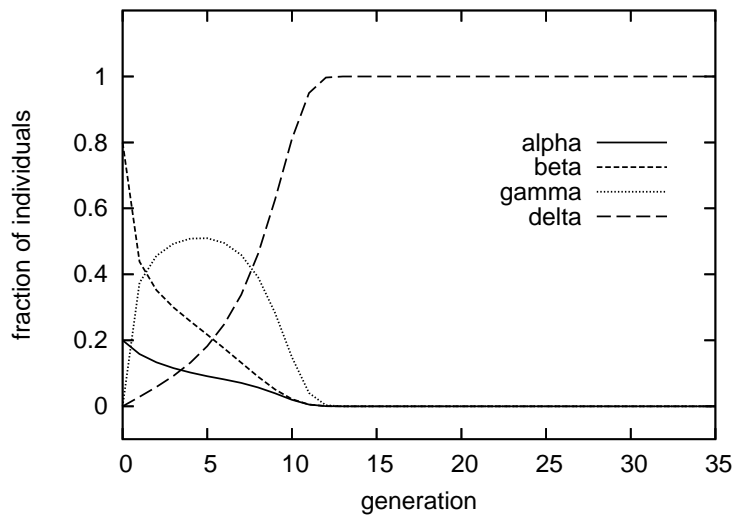
and we know that  $\gamma_{t+1} = 1 - \delta_{t+1} - \alpha_{t+1} - \beta_{t+1}$ .

The model now predicts four possible outcomes, corresponding to the four types of individuals. An example of a case that converges to delta is shown in Fig. 6.10. I also included a simulation of a fitness change,<sup>6</sup> to which I will come back later.

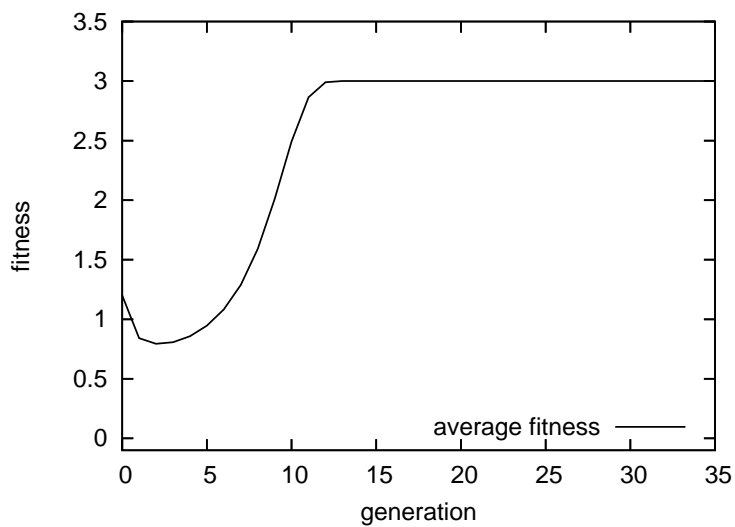
With the extended model at hand, we can create convergence maps as before. The map resulting from the above model is shown in Fig. 6.11a and one from a simulation on a 100-individual population is shown in Fig. 6.11b.

---

<sup>6</sup>Fitness plot assuming  $f(\alpha) = 2$ ,  $f(\beta) = 1$ ,  $f(\gamma) = 0$ ,  $f(\delta) = 3$ .



(a) percentages of individuals



(b) average fitness

Figure 6.10: *Delta* individuals take over, with  $\alpha=10$ ,  $\beta=01$ ,  $\gamma=**$ ,  $\delta=11$ , for  $\alpha_0 = 0.2$  and  $L = 5$ .

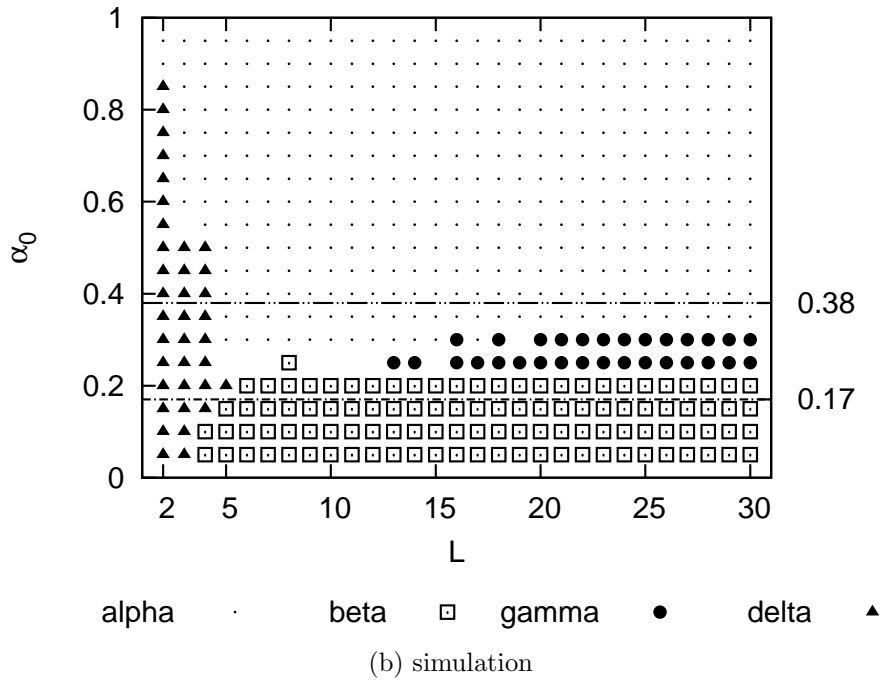
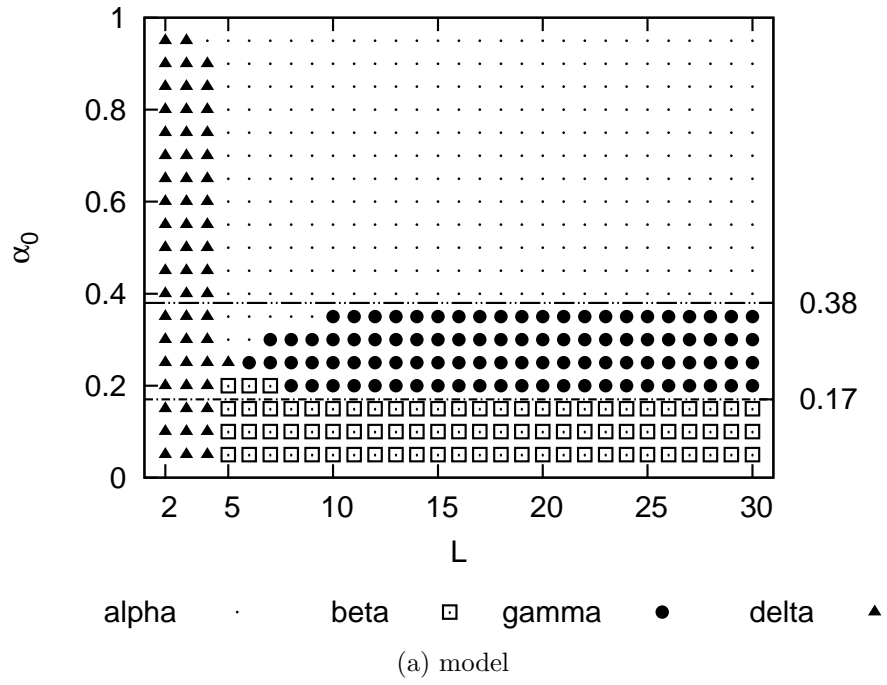


Figure 6.11: Convergence maps with  $\alpha=10$ ,  $\beta=01$ ,  $\gamma=**$ ,  $\delta=11$ .

Again, the effects predicted by the model are compatible with the observation, even if the regions where gamma or delta individuals dominate are a bit smaller than predicted.

We can clearly see that delta individuals take over the population only for  $L < 5$ , when the probability of creating a delta individual is relatively high. This confirms how important it is to design a proper recombination operator for real problems.

### 6.2.3 Concatenating building blocks

Let us look at a modification of the model. Previously I have assumed that all genes of alpha or beta are defined, and any change to them resulted in a worse fitness, unless a delta individual was created. This assumption corresponded to breaking existing longer BBs in order to create a new one out of shorter BBs. Let us now consider a case in which a concatenation of BBs, rather than a combination of BBs, creates a new BB.

Let us assume alpha individuals of form  $\mathbf{1^*}$  (that is we do not require the second part of the genome to be  $\mathbf{0}$ ), beta individuals of form  $\mathbf{*1}$ , delta  $\mathbf{11}$  and gamma  $\mathbf{**}$ . Nevertheless, I assume that initial alphas in generation 0 are defined by  $\mathbf{10}$ . Similarly, the initial beta individual is defined by  $\mathbf{01}$ . This way they do not share any genes. If we allow for random alleles in genes not defining the building blocks for alpha and beta individuals, then (assuming a binary encoding) on average, half of the genes in both alpha and beta would be same. This would effectively correspond to the same analysis (as the one following), but with genomes of length  $L/2$ , since the matching genes in alpha and beta could be disregarded.

To stress the need of at least two genes for combination, the analysis was done for  $L \geq 2$ . As before, for  $L = \infty$  the probability of creating delta is zero, as well as the probability of creating alpha or beta from parents different than alpha or beta, respectively. Therefore, the analysis is identical to the one performed in section 6.2.1, and below I assume  $2 \leq L < \infty$ .

Again, the probability of creating an individual as a result of recombination of two other individuals will be a multiplication of appropriate probabilities for its BBs, for example:

$$p_{\alpha, \gamma \rightarrow \alpha} = p_{1^*, ** \rightarrow 1^*} = p_{1, * \rightarrow 1} p_{*, * \rightarrow *} = p_{1, * \rightarrow 1} (1 - p_{*, * \rightarrow 1}) = q'_3 (1 - q'_2).$$

An analysis similar to the one above leads to the creation of the following matrices:

$$A = \begin{bmatrix} 1 - q'_2 & q'_3(1 - q'_3) & q'_3(1 - q'_2) & 1 - q'_3 \\ q'_3(1 - q'_3) & 0 & q'_2(1 - q'_3) & 0 \\ q'_3(1 - q'_2) & q'_2(1 - q'_3) & q'_2(1 - q'_2) & q'_3(1 - q'_3) \\ 1 - q'_3 & 0 & q'_3(1 - q'_3) & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 & q'_3(1 - q'_3) & q'_2(1 - q'_3) & 0 \\ q'_3(1 - q'_3) & 1 - q'_2 & q'_3(1 - q'_2) & 1 - q'_3 \\ q'_2(1 - q'_3) & q'_3(1 - q'_2) & q'_2(1 - q'_2) & q'_3(1 - q'_3) \\ 0 & 1 - q'_3 & q'_3(1 - q'_3) & 0 \end{bmatrix},$$

$$D = \begin{bmatrix} q'_2 & (q'_3)^2 & q'_2 q'_3 & q'_3 \\ (q'_3)^2 & q'_2 & q'_2 q'_3 & q'_3 \\ q'_2 q'_3 & q'_2 q'_3 & (q'_2)^2 & (q'_3)^2 \\ q'_3 & q'_3 & (q'_3)^2 & 1 \end{bmatrix}.$$

The probabilities of choosing particular individuals as parents are the same like



in the previous model, as well as equations describing the evolution of individuals' percentages. Therefore, plugging the above matrices into the model allowed me to generate a convergence map in Fig. 6.12a. An appropriate simulation corresponding to this model resulted in Fig. 6.12b.

Somewhat greater differences between the experiments and the model may have been caused by several factors. First, the probabilities were computed assuming that genes described by \* with equal probability may be of either value (0 or 1) — and this is not true when initializing the population with **10s** and **01s**. Therefore, the probabilities for creating delta individuals are higher in the model, than in reality. This may lead to actually modeling behavior for smaller  $L$  than assumed. Hence, in the model's convergence map, the region for delta stretches to higher  $L$ , compared with the experiments' convergence map. Secondly, both alpha and beta, which are now wider classes of individuals, survive more easily since they must only maintain half of the genome fixed. This results in a smaller region for gammas.

Overall, we again obtain qualitatively similar results to the previous experiments. Only if  $L$  is small, good combinations or BBs (delta individuals) survive. For higher  $L$  the situation resembles the previous cases, including a region in the parameter space, where the bad recombinations (gamma individuals) survive. Of course, for solving problems we are rather interested in possibly always creating good combinations (delta individuals), specially for larger  $L$ s. In Section 6.3 we will see that, for example, repeated migration in IMs may help with this task.

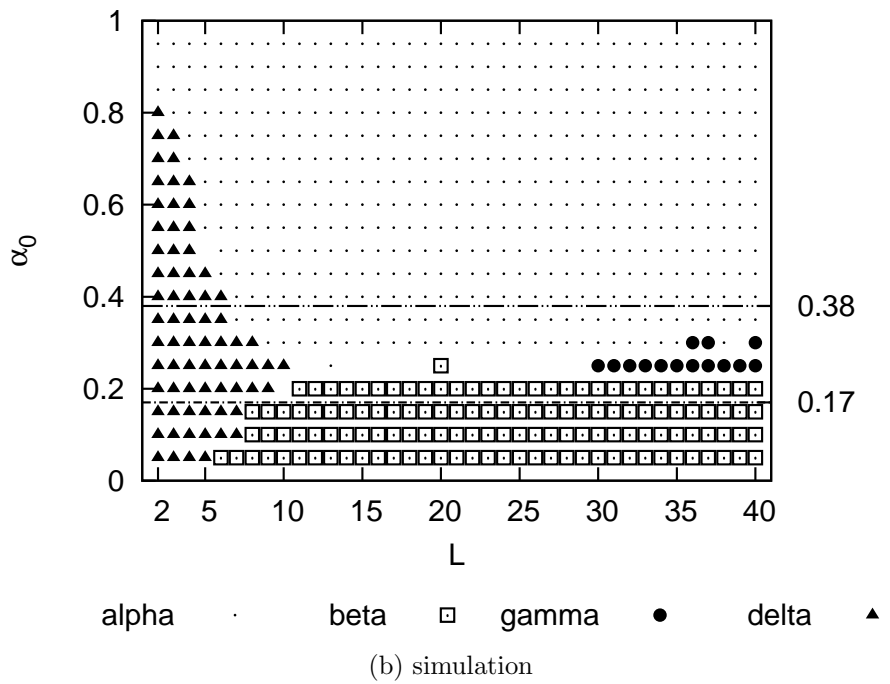
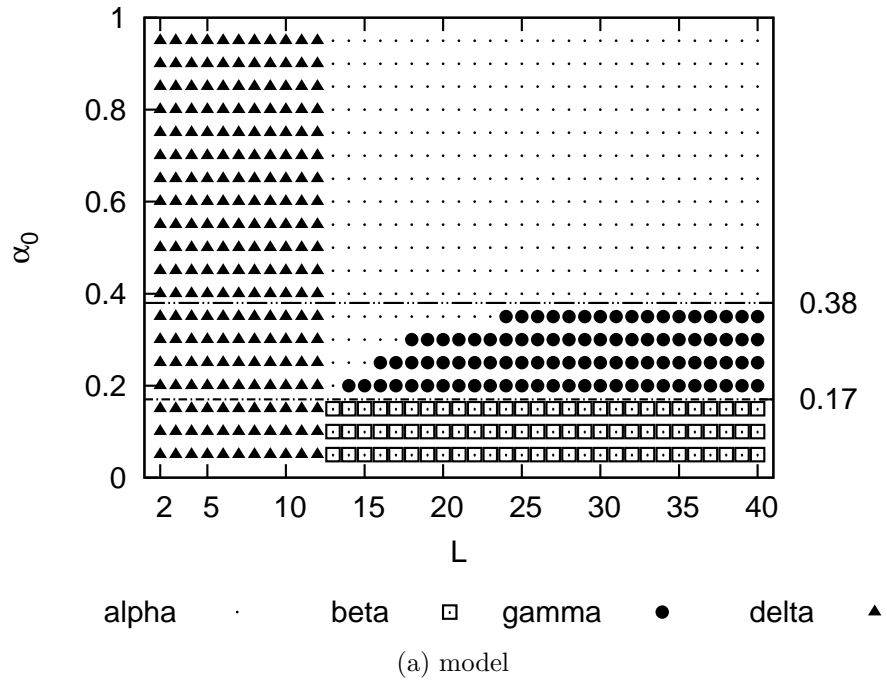


Figure 6.12: Convergence maps with  $\alpha=1^*$ ,  $\beta=^*1$ ,  $\gamma=^{**}$ ,  $\delta=11$ .

## 6.2.4 General model

In this subsection I try to simplify and generalize the  $\alpha$ - $\beta$ - $\gamma$ - $\delta$  model. We have seen on a few examples that the standard dynamics in a population, after migration goes through certain phases. At the beginning alpha and beta individuals coexist in relatively large quantities, followed by the creation of large quantities of gammas. Ultimately, either alpha, beta or gamma individuals dominate, or delta type individuals may be created and spread in the system.

The creation of a delta individual depends on the probability with which it can be created by recombination from other individuals. The survival and domination of deltas depend on the probability of preserving building blocks (and delta individuals in particular) by recombination. In the previous analysis the  $L$  parameter controlled the probability of a delta creation by recombination. Survival of delta individuals also depends on the selection pressure. A strong selection pressure would “notice” even a single delta, and a weak selection may allow an entire group of deltas to become extinct.

Looking at matrices used in sections 6.2.1–6.2.3, we notice that some entries are much larger (and significant for the model) than others. I assume that the effect of certain smaller entries can be neglected, and others can be simplified, as follows. A crossover of two parents of the same individual type will with high probability produce an offspring of the same type. Gamma individuals have no BBs, so I assume that creating a BB in a recombination involving a gamma individual is quite unlikely and I model it as impossible. All BBs used were of similar structure and length. Therefore, I assume a single parameter,  $p_s$ , describing the probability of a given BB to *survive*. The more destructive the recombination operator, the smaller this

probability will be. The probability of creating a delta individual from alpha and beta could in certain cases be modeled as a multiplication of probabilities of proper parent BBs surviving. However, in general these events are not independent, and the probability must be specified separately. Therefore, I use a second parameter,  $p_c$ , describing the probability to *construct* a higher level BB by composing shorter BBs. The more constructive the recombination operator, the higher this probability will be.

The above discussion can be shown in the following simplified matrices describing the probabilities of recombination creating the appropriate individual types:

$$A = \begin{bmatrix} 1 & p_s & p_s & 1 - p_s \\ p_s & 0 & 0 & 0 \\ p_s & 0 & 0 & 0 \\ 1 - p_s & 0 & 0 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 & p_s & 0 & 0 \\ p_s & 1 & p_s & 1 - p_s \\ 0 & p_s & 0 & 0 \\ 0 & 1 - p_s & 0 & 0 \end{bmatrix},$$

$$D = \begin{bmatrix} 0 & p_c & 0 & p_s \\ p_c & 0 & 0 & p_s \\ 0 & 0 & 0 & 0 \\ p_s & p_s & 0 & 1 \end{bmatrix}.$$

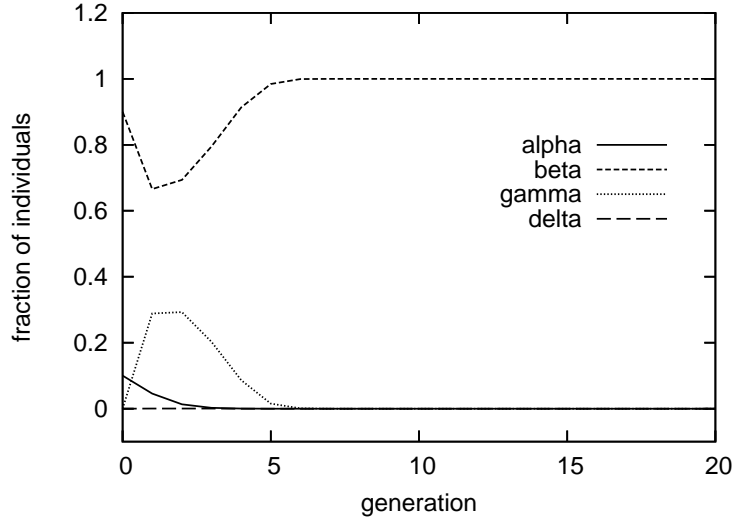
In the first two matrices,  $A$  and  $B$ , only  $p_s$  parameter is used, because alpha and beta individuals persist if appropriate BBs survive. In the third matrix,  $D$ , the  $p_c$  parameter is used to capture the probability of a delta individual being created, by combining an alpha and a beta. For the uniform recombination the probability of

preserving a BB is  $p_s = q'_2 = (0.5)^{L/2}$ . The probability of creating a new BB will be given  $p_c = (q')^2 = (0.5)^L$ . In Fig. 6.13 we see four possible cases of convergence for this model. In Fig. 6.14 we see a convergence map, which is very similar to the previous results (see Fig. 6.11) with uniform recombination.

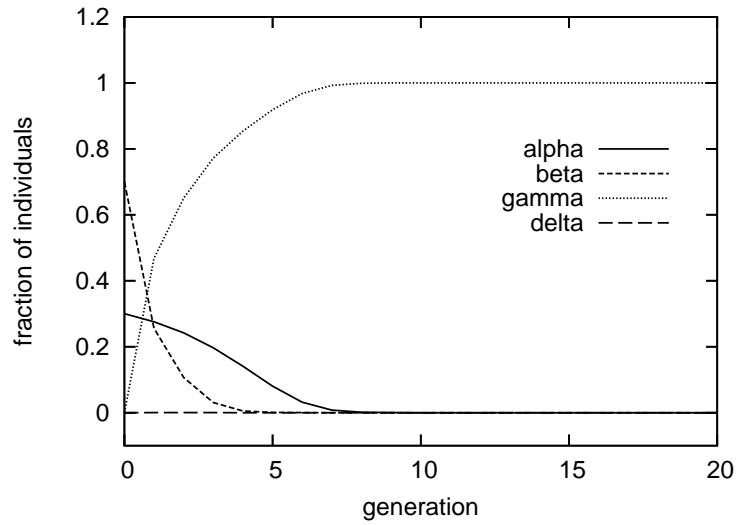
The behavior in the general model matches the one from more detailed models. This adds credibility to my understanding of after-migration dynamics, because it confirms that the general model captures the important aspects of earlier models.

If we assume concrete fitness values for individual types, we can plot how fitness changes after migration. Although the curves will be different for different models, or different fitness assignments, the general shape remains the same. In Fig. 6.15 we see the average fitness for the four cases that we have just analyzed, assuming that the individuals have the following fitness values:  $f(\alpha) = 2$ ,  $f(\beta) = 1$ ,  $f(\gamma) = 0$  and  $f(\delta) = 3$ . Compare also Fig. 6.10b.

In all cases we observe an interesting phenomenon, which is a (usually) temporary drop in average fitness during the transition phase. This helps understand why a selection that is too strong may be harmful for combining migrants with locals. Namely, this type of selection would prevent the intermediate mixture of alleles to survive, favoring more fit, yet less evolvable, individuals. Also, it confirms an assumption that separation of islands helps create local niches where new solutions may be created without an immediate competition with all of the individuals in the system.

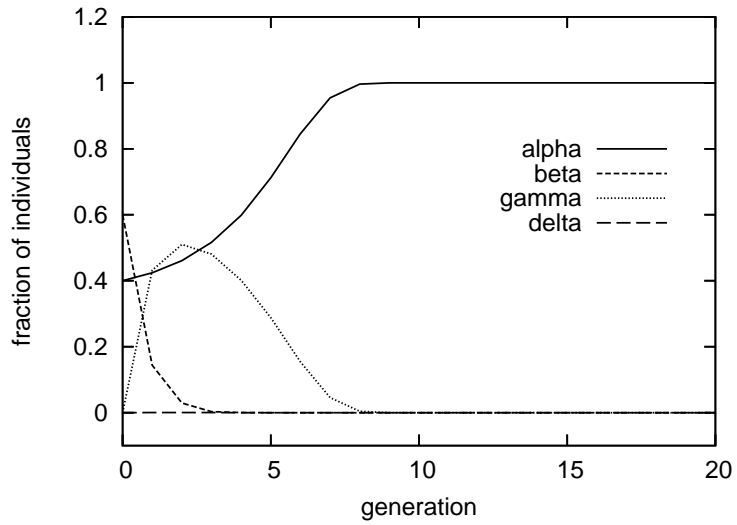


(a)  $\alpha_0 = 0.1$ ,  $L = 10$ , *beta* individuals taking over

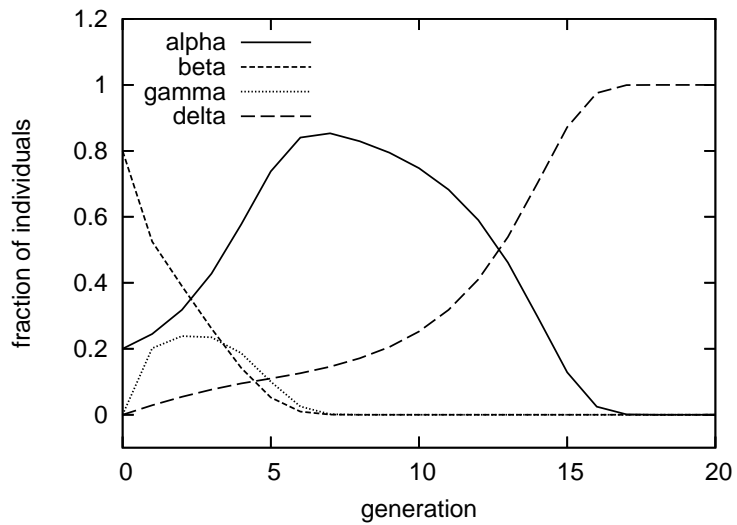


(b)  $\alpha_0 = 0.3$ ,  $L = 10$ , *gamma* individuals taking over

Figure 6.13: Different  $\alpha_0$  and  $L$  lead to the domination of different individuals. Results from the general model,  $p_s = (0.5)^{L/2}$ ,  $p_c = (0.5)^L$  (uniform recombination).



(c)  $\alpha_0 = 0.4$ ,  $L = 10$ , *alpha* individuals taking over



(d)  $\alpha_0 = 0.2$ ,  $L = 4$ , *delta* individuals taking over

Figure 6.13: Different  $\alpha_0$  and  $L$  lead to the domination of different individuals. Results from the general model,  $p_s = (0.5)^{L/2}$ ,  $p_c = (0.5)^L$  (uniform recombination).

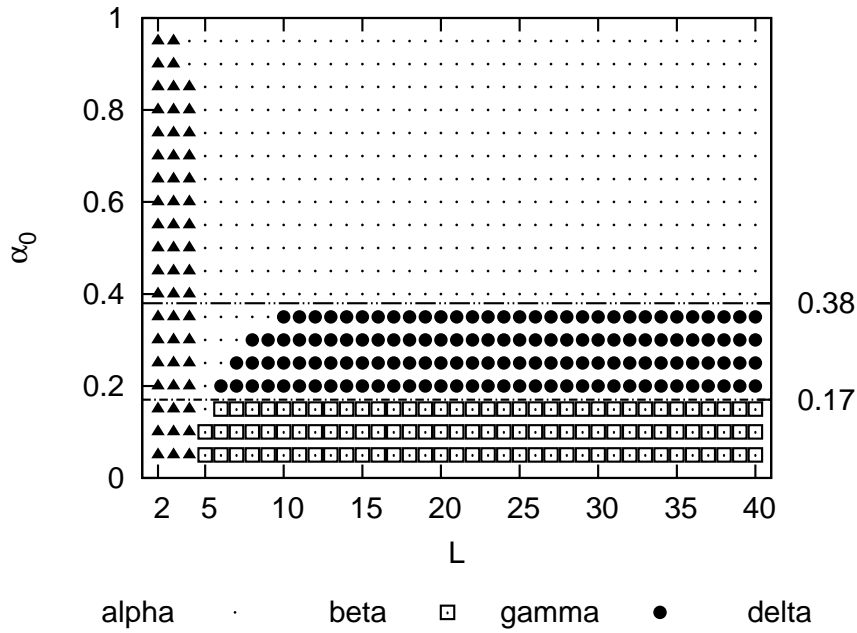
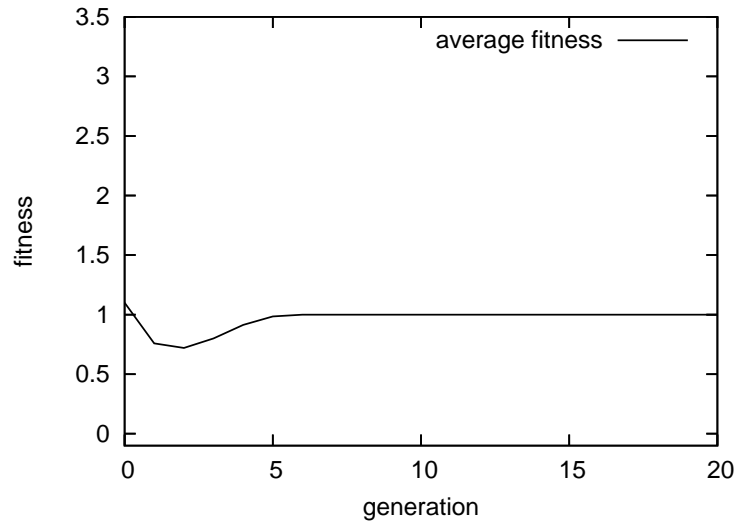


Figure 6.14: Convergence map with the general model,  $p_s = (0.5)^{L/2}$ ,  $p_c = (0.5)^L$  (uniform recombination).

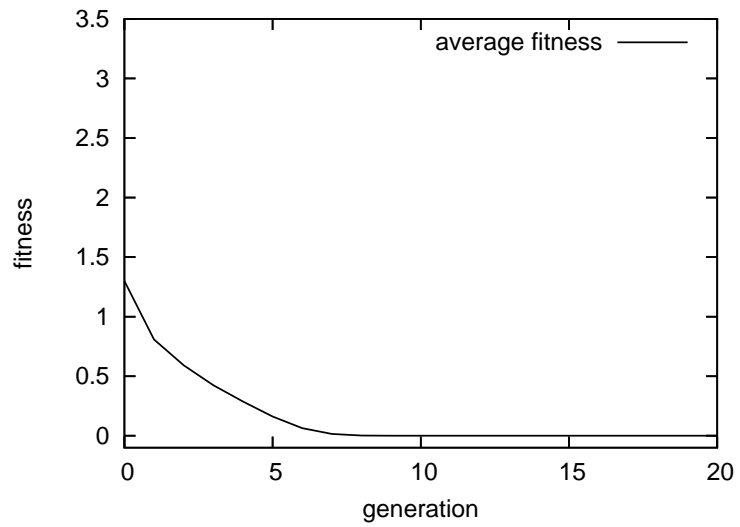
### 6.3 Improving migrations

Two methods of increasing the likelihood of creating delta individuals (and promoting the creation of new combined solutions) are as follows. The first one is to redesign the recombination operator (or use a different one), so that BBs are better preserved, and new BBs better created. The second is to try increasing the selection pressure in islands. Since I assumed a higher fitness for delta individuals, higher selection may enhance their ability to survive, however, it may be at the cost of decreasing the probability to create them initially.



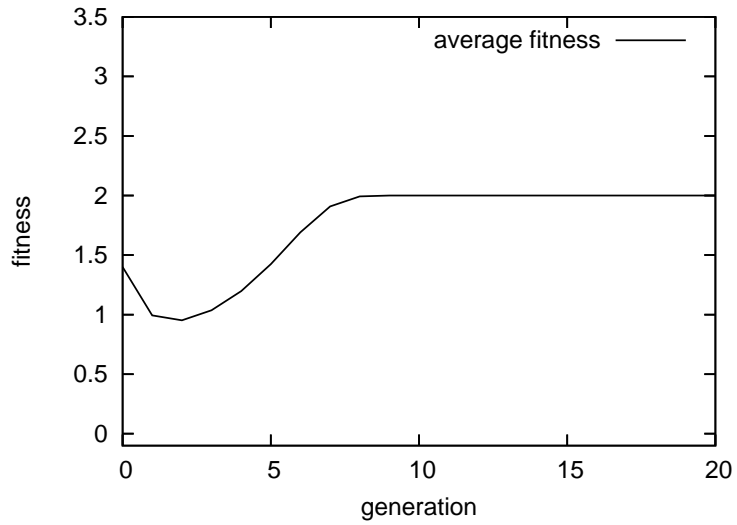


(a)  $\alpha_0 = 0.1$ ,  $L = 10$ , *beta* individuals taking over

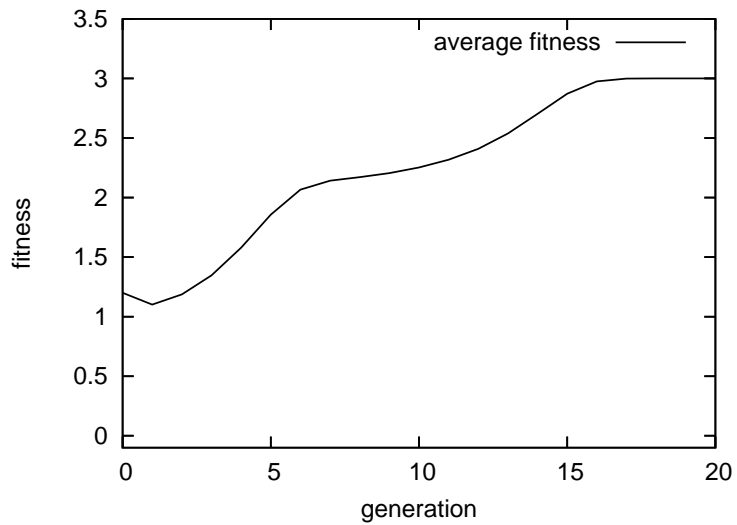


(b)  $\alpha_0 = 0.3$ ,  $L = 10$ , *gamma* individuals taking over

Figure 6.15: Average fitness after migration for different convergence scenarios. Results from the general model,  $p_s = (0.5)^{L/2}$ ,  $p_c = (0.5)^L$  (uniform recombination).



(c)  $\alpha_0 = 0.4$ ,  $L = 10$ , *alpha* individuals taking over



(d)  $\alpha_0 = 0.2$ ,  $L = 4$ , *delta* individuals taking over

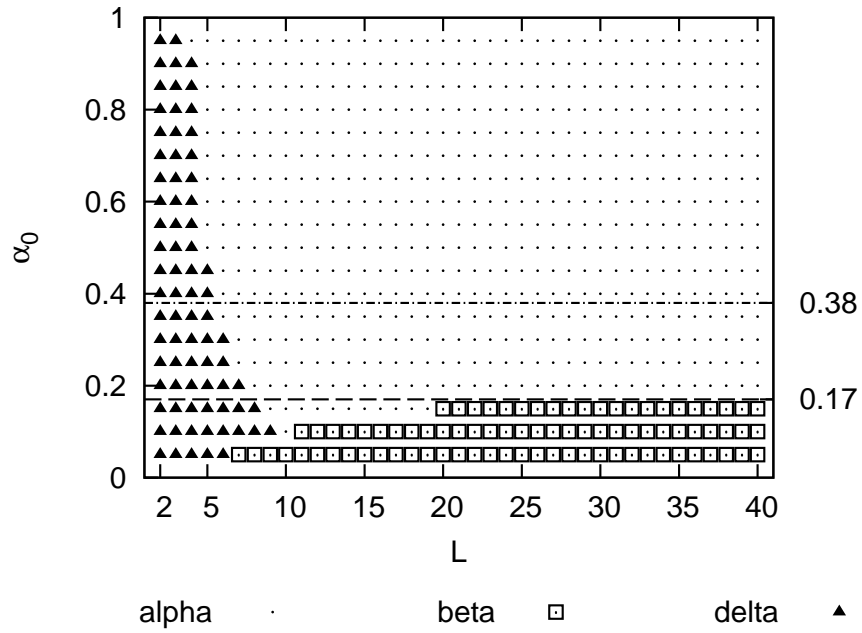
Figure 6.15: Average fitness after migration for different convergence scenarios. Results from the general model,  $p_s = (0.5)^{L/2}$ ,  $p_c = (0.5)^L$  (uniform recombination).

### 6.3.1 Redesigning recombination operator

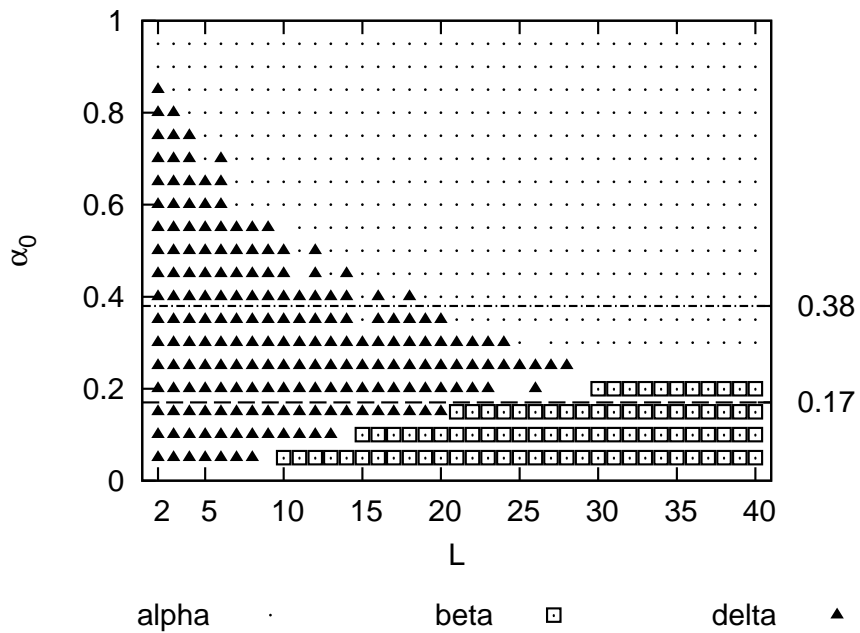
It is easy to observe that the domination of delta individuals strongly depends on the  $p_c$  probability. A general rule of thumb is the easier it is to create deltas, the more often it is that they will dominate. How to increase the probability of creating better individuals as a result of combining BBs is not obvious, however. One way to increase it in real systems is to use a recombination operator that maintains good linkage between genes.

A traditional one-point crossover is an example of a linkage-preserving recombination operator for the two-BBs problem and suits our needs. A probability of preserving a BB of length  $\frac{L}{2}$ , using one-point recombination is in fact  $p_s \geq 0.25$  regardless of  $L$ , and I assume it to be 0.25. This is because either of the two possible children may be chosen, and the crossover point may be chosen either inside or outside the BB (which is contiguous). The probability of creating a delta individual decreases with a higher  $L$ , because the crossover point must be placed between BBs (and the proper offspring must be chosen). Therefore I take  $p_c = 1/2L$ . A convergence map for the general model with these parameter values is shown in Fig. 6.16a.

In Fig. 6.16b I show a convergence map resulting from analogical simulation (on a 100-individual population), using one-point crossover. As we see, deltas are now created for much longer genomes. Although the match is not perfect, the increase of the maximal  $L$  is seen in both the model and the simulation (where maximal  $L$  is equal to 28, compared to a value of 5 before).



(a) general model,  $p_s = 0.25$ ,  $p_c = 1/2L$



(b) simulation with alpha=10, beta=01, gamma=\*\*, delta=11.

Figure 6.16: Convergence maps with a redesigned recombination (one-point crossover).

### 6.3.2 Selection pressure

As already mentioned, not only must deltas be created, but they must also survive. Since delta individuals are, by assumption, better than other individuals, a strong selection favors them.

Below I comment on strong and weak selection, based on experience gained from my cumulative experiments. Strong selection IMs are worse at combining BBs to create new ones. In this case, it is important to use very small migration sizes to make effective combinations possible, and/or recombination operators that are likely to effectively combine sub-solutions very fast. Larger migration sizes would only cause unnecessary additional selection pressure. A smaller migration size provides separation of the islands, which is beneficial; as we have seen in Chapter 4, separation of individuals stops more influential genes from enforcing the fixation of weaker genes through the hitch-hiking effect, and also allows creation of different sub-solutions (or BBs). Although migration intervals cannot be too short, generally a strong selection EA will converge faster locally, so longer migration intervals are unnecessary, if not detrimental to the performance because of the total loss of local diversity.

Weak selection IMs are better at combining sub-solutions and will benefit from relatively larger migrations (although still around only 10% for binary tournament, as we have seen) to help solve hierarchical problems. However, there may be a problem with the elementary BBs supply. Intuitively speaking, it seems that forming BBs (exploitation) in islands cannot proceed slower than combining/merging these BBs with other islands, which is achieved through migration, because diversity of sub-solutions will be lost. Long migration intervals, with higher migration sizes should result in even weak EAs in each island finally converging, and in successful

combination of BBs coming from different islands in the succeeding intervals.

A convergence map for a tournament of size 4, as opposed to binary tournament used for parent selection, is shown in Fig. 6.17. We see that in fact stronger selection helps delta to dominate, but as expected even smaller values of  $\alpha_0$  are required if we want to prevent the domination of alphas.

Finally, in Fig. 6.18, we see analogical results for one-point crossover together with 4-tournament selection. We observe an even further increase in the creation of deltas.

### 6.3.3 Repeated migrations

In the previous sections I have analyzed how a single population converges, given certain proportions of initial individuals. This corresponds to a situation after a single migration, when an island consists of two types of individuals (locals and migrants). As mentioned before, we could see the models as either  $\alpha_0 M$  of alpha individuals migrating, or  $\beta_0 M$  of beta individuals.

In this section I analyze some special effects resulting from repeating migration, as it does in standard IMs. I will assume that the migrants are better (so they are alpha individuals), because ultimately we are interested in spreading better solutions or parts thereof in the system. Under this assumption, after every migration interval, randomly selected  $\alpha_0$  fraction of individuals are substituted with alpha individuals, and I can describe these changes with the following equations:

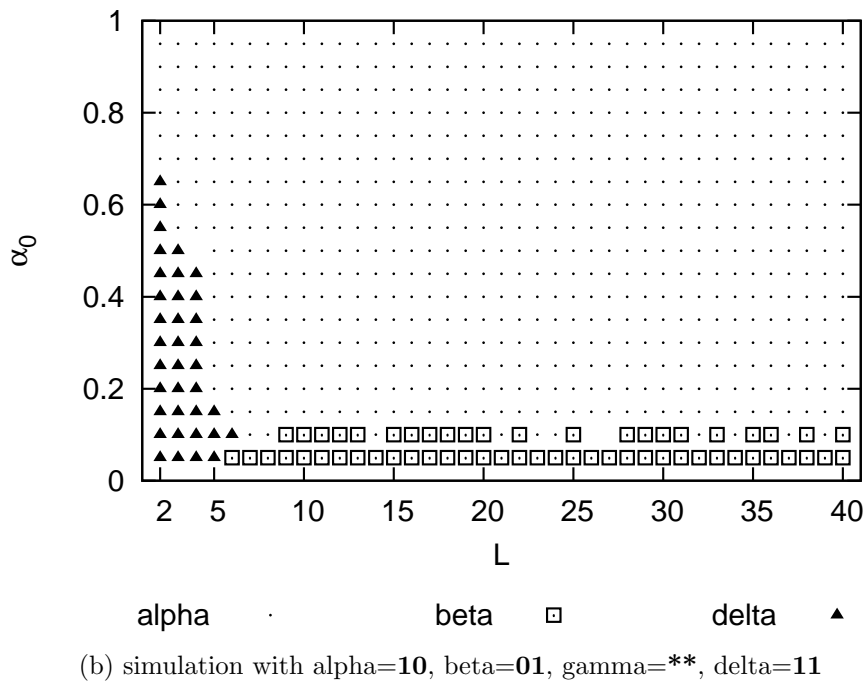
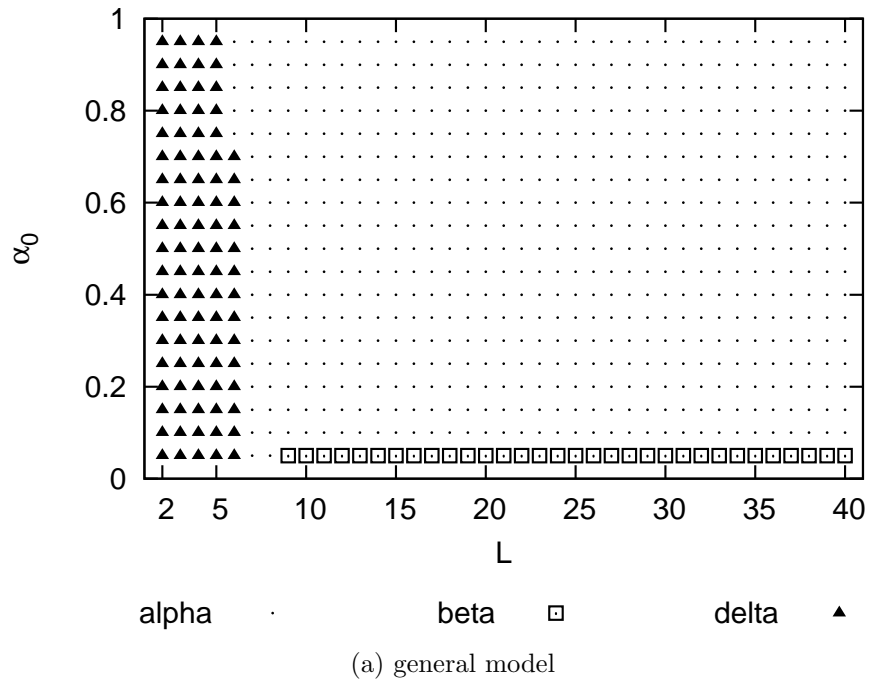


Figure 6.17: Convergence maps with an increased selection pressure (a 4-tournament parent selection).

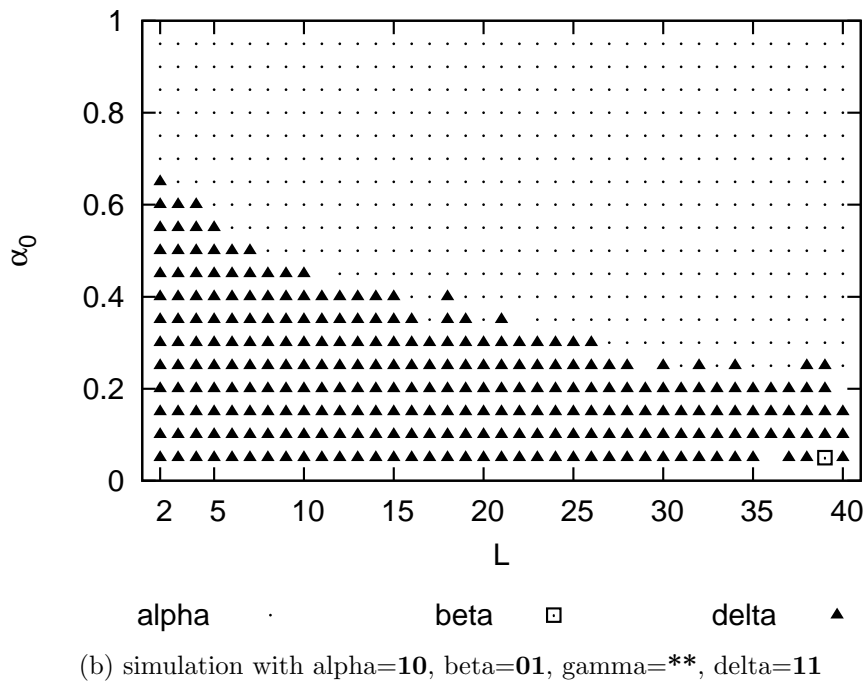
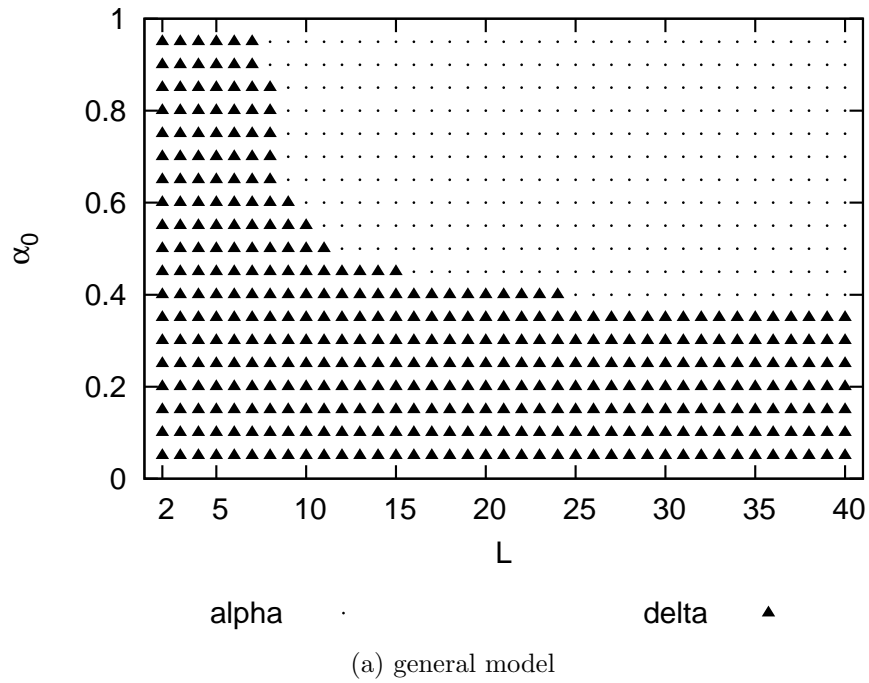


Figure 6.18: Convergence maps with both increased selection pressure (4-tournament parent selection) and a redesigned recombination (one-point crossover)



$$\alpha' = \alpha(1 - \alpha_0) + \alpha,$$

$$\beta' = \beta(1 - \alpha_0),$$

$$\gamma' = \gamma(1 - \alpha_0),$$

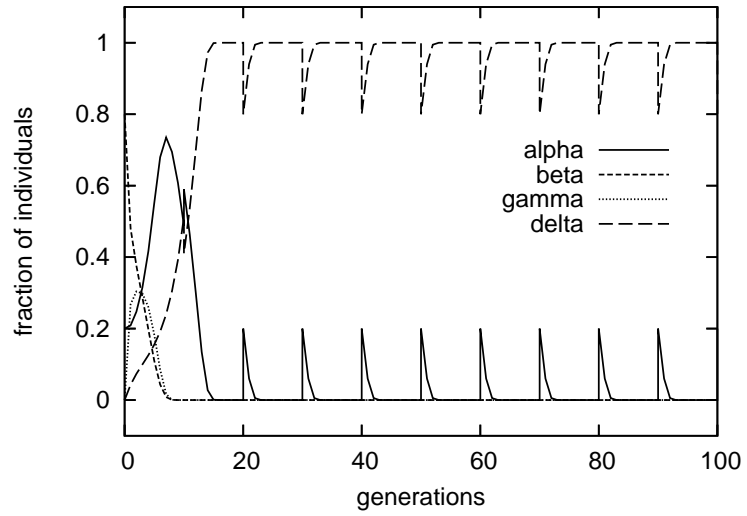
$$\delta' = \delta(1 - \alpha_0).$$

Let us analyze the four scenarios when different individual types dominate. If delta starts to dominate in the population, repeated migrations of alphas at reasonable rates should not affect this dominance, since alphas have lower fitness (and a recombination of alpha and delta is always one of these two individuals). The model prediction generally matches the experiments. This is shown in Fig. 6.19.

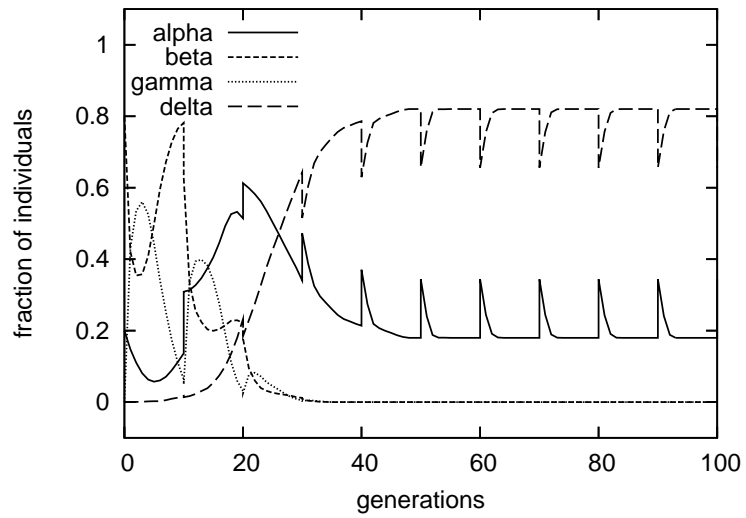
As we have seen for larger  $L$ , delta individuals are not created so easily. If migration size ( $\alpha_0$ ) is so large that it causes the population to converge toward alphas due to a single migration, then obviously repeated insertions of alphas will not change the situation (other than possibly speeding up the convergence). The model agrees with experiments again, and this is shown in Fig. 6.20.

On the other end of the spectrum, if migrations are really small, they have only a temporary influence on the behavior of the system, which quickly “recovers” back to all betas. Also in this case the predictions of the model seem to agree with experiments, as shown in Fig. 6.21 (why it is not exactly true, we will see later).

What happens for  $\alpha_0$  of an intermediate value? The model predicts a convergence to gamma. Although this is sometimes true, with repeated migrations it is observed

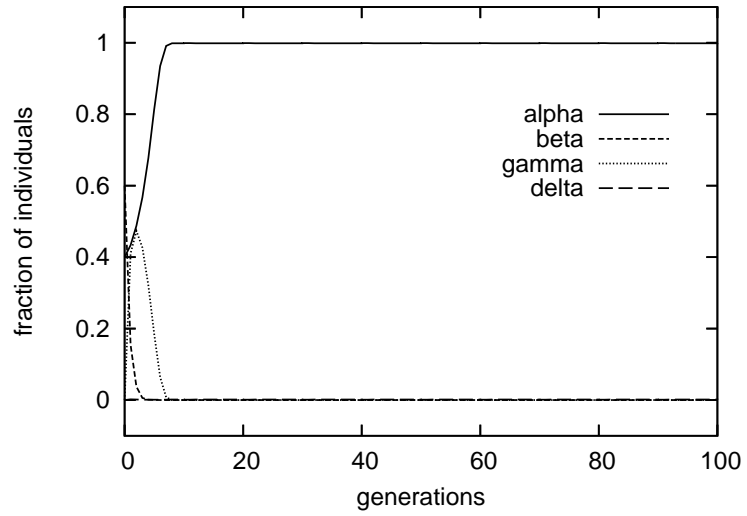


(a) model

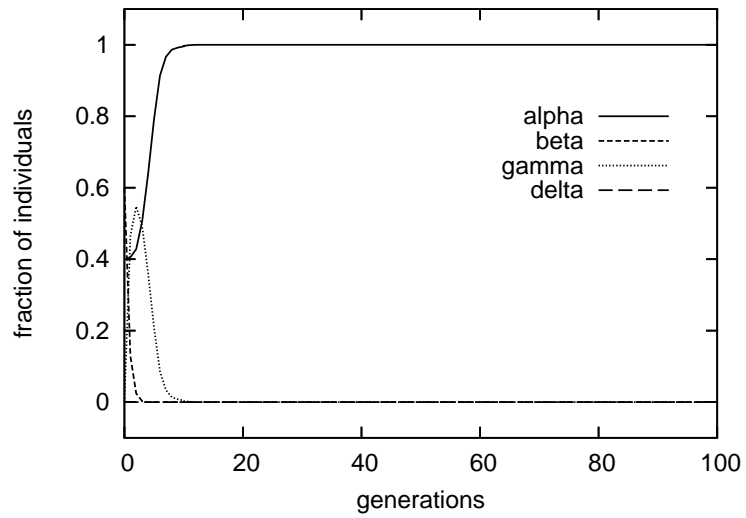


(b) experiments

Figure 6.19: *Delta* individuals dominate for  $\alpha = 0.2$ ,  $L = 10$ . Results from the model with  $\alpha=1^*$ ,  $\beta=1^*$ ,  $\gamma=**$ ,  $\delta=11$ .

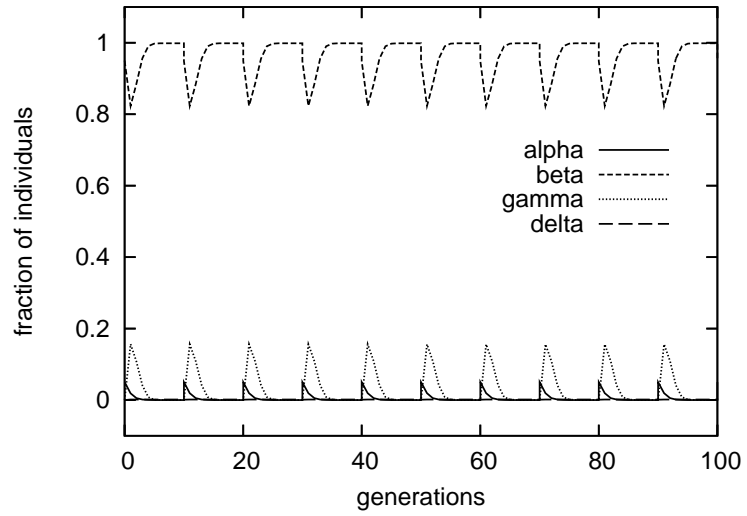


(a) model

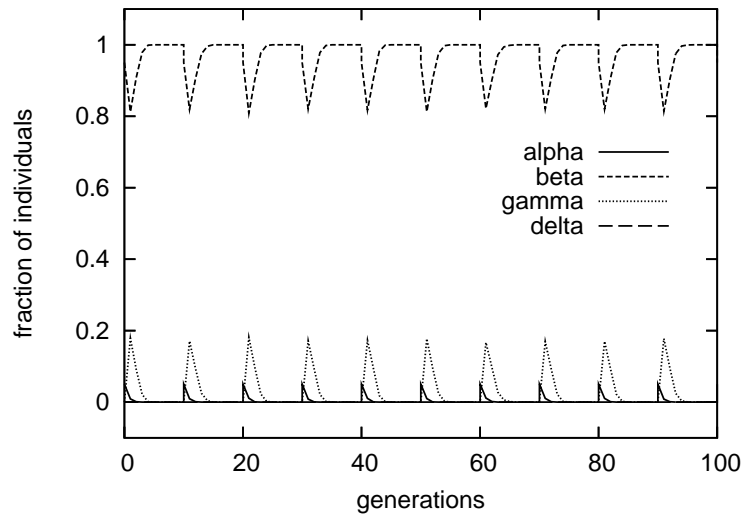


(b) experiments

Figure 6.20: *Alpha* individuals dominate for  $\alpha = 0.4$ ,  $L = 20$ . Results from the model with  $\alpha=1^*$ ,  $\beta=1^*$ ,  $\gamma=**$ ,  $\delta=11$ .



(a) model



(b) experiments

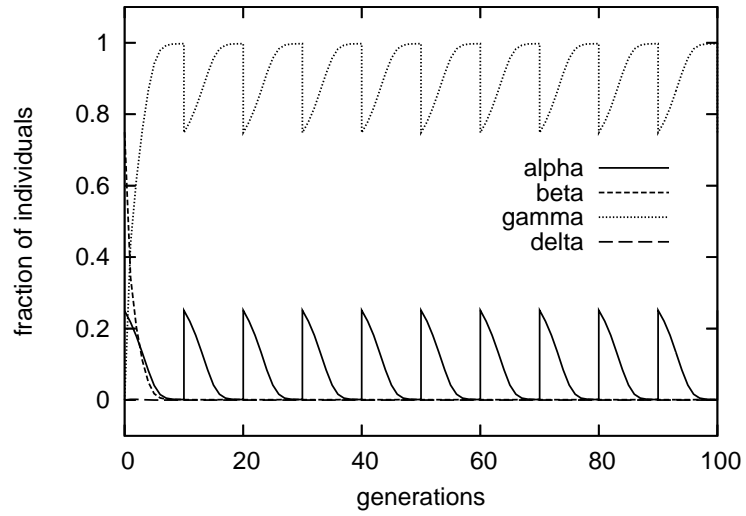
Figure 6.21: *Beta* individuals dominate for  $\alpha = 0.05$ ,  $L = 20$ . Results from the model with  $\alpha=1^*$ ,  $\beta=1^*$ ,  $\gamma=1^*$ ,  $\delta=11$ .

quite rarely. Instead, in reality we often observe a convergence to alpha (see Fig. 6.22). To some extent this is caused by repeated migrations of alpha, before the island has converged to all gammas after the previous migration, which consequently increases the alpha percentage in the population. However, the model should appropriately predict this growing alpha percentage, and it does not. Therefore, there must be something else causing the increase of alphas.

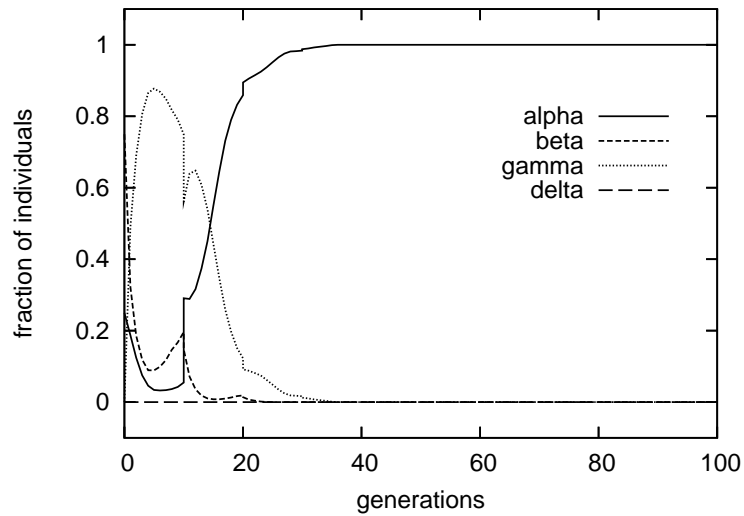
The difference is due to simplification used in the model. In the model I do not compute the actual distribution of alpha and beta genes in the individuals — and repeated recombinations of alphas and gammas cause the latter to contain more and more alpha genes. In effect, alpha individuals become more easily created by recombination. This effect probably significantly contributes to the faster and easier convergence to all alphas.

Furthermore, whereas in the model the survival of gammas may be due to an unrealistic balance between alpha and beta genes (I always assume the same probability of gamma's genes being in either state); in the actual system it is the fixation to a single gamma individual. Migrations repeatedly disturb fixations, however, making the gamma state ultimately unstable.

Let us then analyze the case of convergence to betas again. We have seen that the actual distribution of genes in the “random” parts of individuals is important. It turns out that when the system recovers to “all betas,” these betas are not the initial betas in the form of **01**. Rather, some alpha genes have been already transferred into the first half of the genome, while the second half remained untouched. Because the first half does not contribute to fitness, neutral evolution allows for accumulation of alpha genes.



(a) The model predicts convergence to *gamma*...



(b) ...but experiments show a fast domination of *alphas*.

Figure 6.22: Inconsistency between the model and a simulation for  $\alpha = 0.25$ ,  $L = 30$ . Results from the model with  $\alpha=1^*$ ,  $\beta=1$ ,  $\gamma=11$ ,  $\delta=1$ .

A more detailed description is the following. Gammas survive for some period of time and they get mixed with betas. Moreover, some of these gammas, due to previous recombinations between each other and more importantly with betas, can have more than 50% of beta genes. If a beta is recombined with such a gamma, it is much more likely that the beta BB will survive. On the other hand, some of the alpha genes contained in the gamma parent will survive as well. Because each beta has the same fitness (it depends only on the presence of the proper BB), “dirty” betas (with alpha genes in the first half) will survive equally with initial betas. In Fig. 6.23, we see how an average gene value in the first half of beta genomes grows. I compute the average using only beta individuals, so this growth is not caused by any increase in number of alpha individuals, or alpha genes in gammas (both of which are rare, anyway).

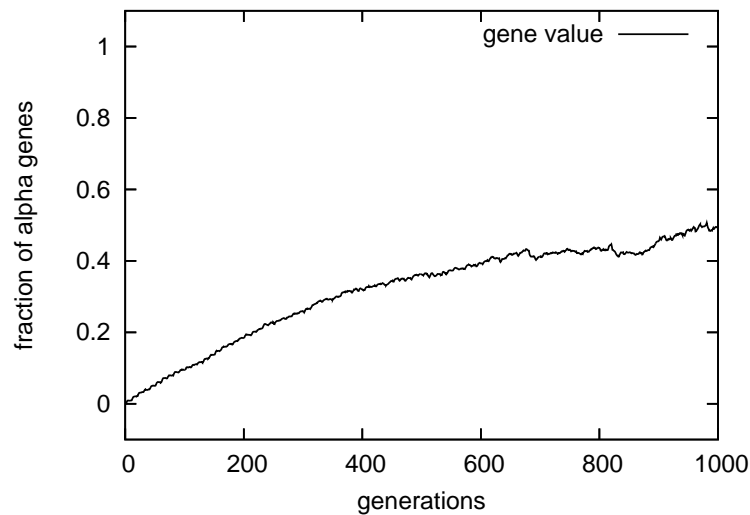


Figure 6.23: The growth of alpha BB genes, while maintaining beta BB,  $\alpha = 0.1$ ,  $L = 20$ .

The situation described so far could very well happen in a single population model

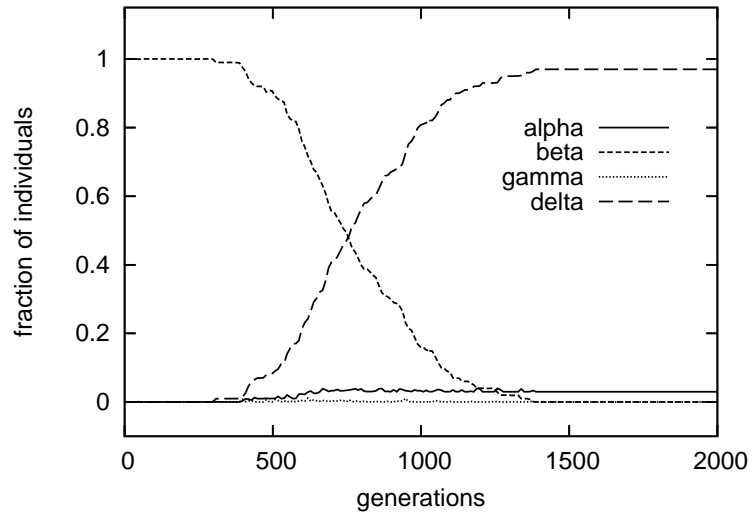
as well. A recombination of alpha and beta, could create a gamma; this gamma could mix with a beta, and so on. However, in a single population (and also in an island, before the next migration occurs), diversity drops fast and fixation occurs. This means that one of the betas, even if containing some alpha genes in its first half, would dominate the population (in case of small  $\alpha_0$ ). In IM, each time migration occurs and the island converges back to a single beta individual, more alpha genes stay in the population.

A constant growth of the alpha genes in beta individuals eventually make it very likely for those individuals to contain both BBs, and become deltas! The neutral evolution I just described is very interesting, because it allows for combining two building blocks gradually, as opposed to requiring recombination to combine them in a single try. Therefore, by keeping the migration low, we can keep the properties of the original individuals, and at the same time, slowly merge good properties of other solutions that evolved in different islands!

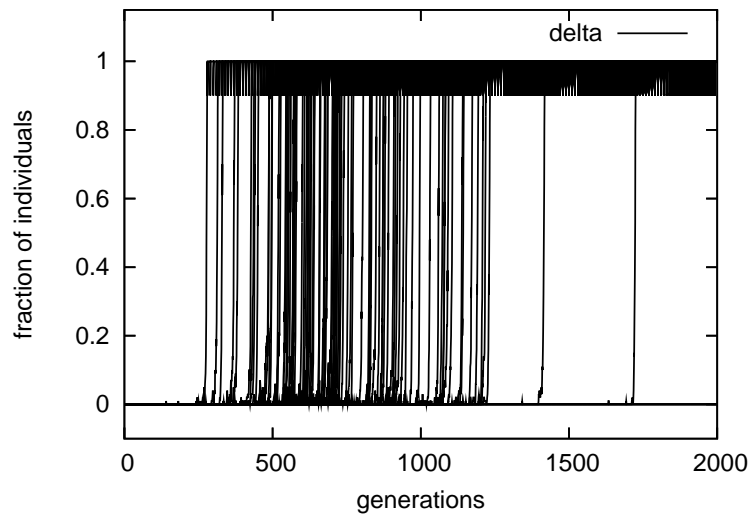
An increasing variance seen in the last figure for higher generations is due to averaging less and less individuals because of beta individuals disappearing from the population. Fig. 6.24a shows the emergence of delta individuals. Note, that although from the figure it seems delta individuals are gradually taking over the population, this is an average over multiple runs. In a single run, once a delta individual gets created, it dominates the population very fast (see Fig. 6.24b).

An interesting note is that when I analyzed the parents of the first delta individual created in each run, they are usually a very “dirty” alpha, and a very “dirty” beta. Whereas I have explained how beta individuals may contain many alpha genes, alpha individuals containing a lot of beta genes remains puzzling because we know that most





(a) average from multiple runs, local changes due to migrations are not shown



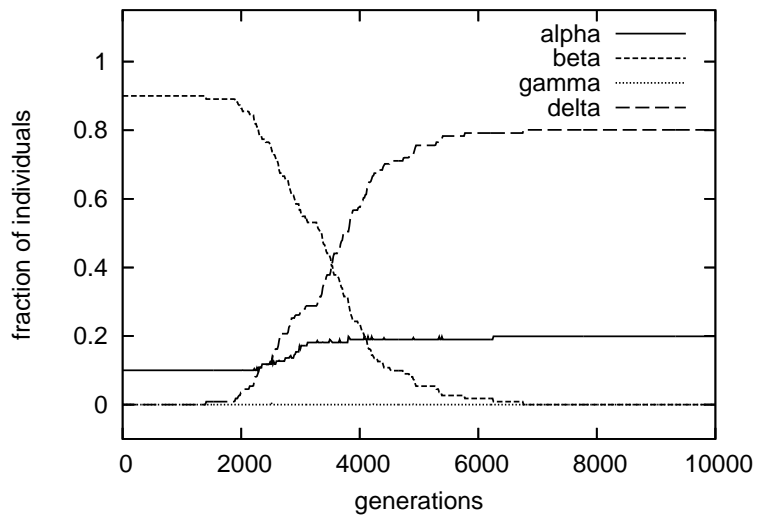
(b) 100 runs shown (only *deltas*)

Figure 6.24: Delta individuals appear,  $\alpha = 0.1$ ,  $L = 20$ ,  $i = 10$ .

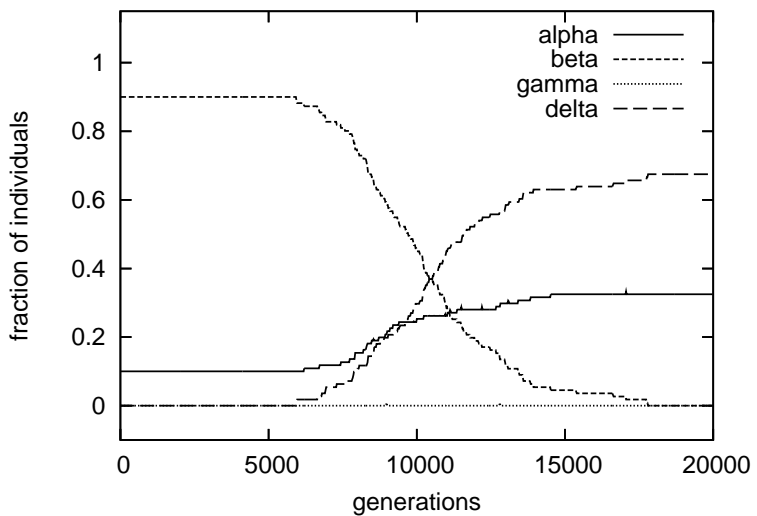
of the alphas were eliminated before the last migration, and the migration injects “clean” alphas of the form of **10**. New alphas must have been able to recombine with betas and gammas in a short time past migration. The explanation is probably that betas contain many alpha genes by the time deltas emerge, so a recombination of alpha and beta, preserving alpha BB is not so unlikely.

Since I use uniform recombination, the BBs need not be continuous. Neutral evolution should automatically preserve non-continuous BBs as well. For longer BBs, a larger migration interval seems appropriate as it allows for an extended period of mixing between gammas and betas. In these situations, it is highly important to be careful with injecting alphas prematurely, to avoid convergence to alphas, and generally use smaller  $\alpha_0$ , which prolongs neutral evolution. I confirmed the necessity of using longer migration intervals (e.g.  $L = 40$ ) experimentally and I show the results in Fig. 6.25.

To observe delta individuals being created for  $L = 40$  runs of 20,000 generations are not uncommon. However, the number of evaluations in long runs are orders of magnitude smaller than the number of trials required to create delta individuals at random from alpha and beta individuals. For  $L = 20$ , I have evolved the population of 100 individuals for up to 2000 generations, which gives  $2 \cdot 10^5$  recombinations/evaluations. Creating a delta individual randomly from alpha and beta would happen once in  $2^{20} \approx 10^6$  cases, so  $10^6$  is the (approximated) expected number of trials before the success occurs. For bigger  $L$ , the speedup is even bigger:  $L = 30$  required  $10^6$  evaluations, whereas randomly creating delta would make us wait  $2^{30} \approx 10^9$  evaluations; for  $L = 40$ , I had to make  $2 \cdot 10^6$  evaluations, but a random search would require around  $2^{40} \approx 10^{12}$  trials!



(a)  $\alpha = 0.1, L = 30, i = 15$



(b)  $\alpha = 0.1, L = 40, i = 20$

Figure 6.25: Delta individuals appear for  $L = 30$  and  $L = 40$ .

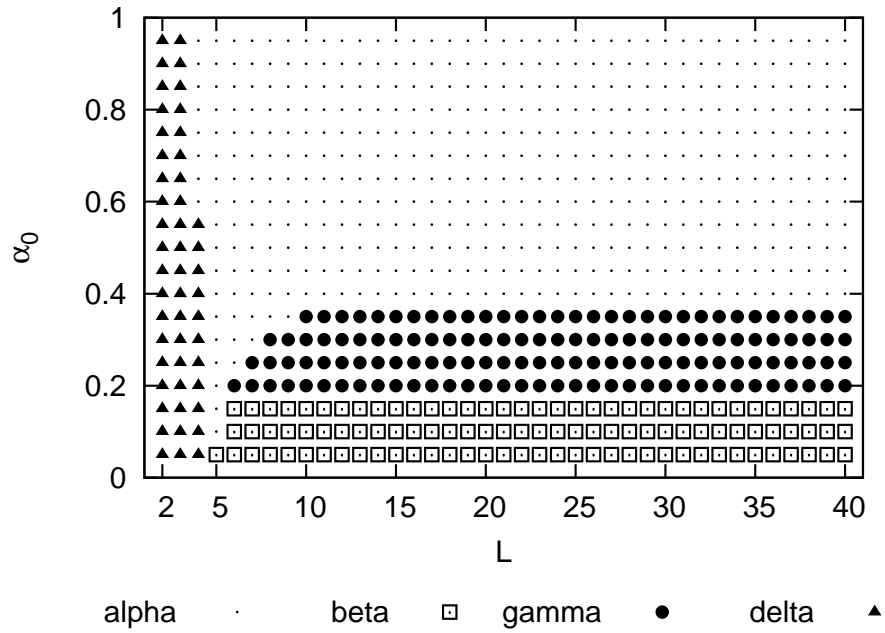
Perhaps just maintaining diversity in a single population could result in a similar mechanism. However, losing the diversity in islands between migrations is a way to multiply the number of copies of a given “dirty” beta individual and seems to be a quite important aspect of the mechanism.

As we have seen, longer migration intervals must be chosen for larger values of  $L$ . In our case, too long of intervals used with a smaller  $L$  only seem to proportionally prolong the time until delta appears, although generally it may have some other negative effects because the diversity of islands is lost completely before new migrants come.

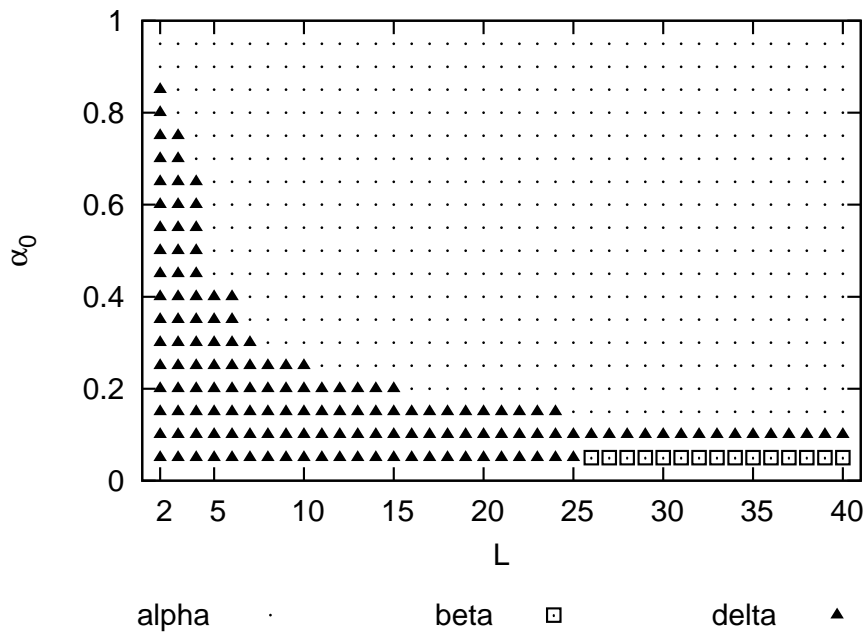
I have fixed the migration interval to 20 and in Fig. 6.26 I show the appropriate convergence maps. While the  $\alpha\beta\gamma\delta$  models correctly predicted dynamics after a single migration, they are not powerful enough to capture the effects of repeated migrations, which effects we clearly see in the simulation. Repeated injections of alpha individuals cause a dominance of alpha individuals for lower  $\alpha_0$  values, and no gamma individuals survive. More importantly, a carefully chosen  $\alpha_0$  may enable creating delta individuals even for large  $L$  (of course, there are limits for this). It seems that by carefully using IMs we can combine BBs for much longer genomes (and more difficult problems) than using a single-population EA.

## 6.4 Inter-island interpretation of scenarios

In this section I discuss similarities between an EA and inter-island processes and show that we can indeed talk about inter-island evolution. The possible scenarios occurring locally (mostly after migration) correspond to various possible interactions between islands, which in turn can be treated as individuals of a higher-level population. In



(a)  $\alpha\beta\gamma\delta$  models do not capture the effect of repeated migrations



(b) a simulation shows that migrations effect accumulate

Figure 6.26: Convergence maps with repeated migrations,  $i = 20$ .

fact, these interactions form all major EA components at the inter-island level, as explained in greater detail below.

Migrant rejection corresponds to little change at the global level. Some neutral evolution as we have seen is possible, but generally if migrants and their offspring die, the island population will contain only local offspring, as if no migration occurred.

Mixing migrant genes and converging to a new individual corresponds to recombination between islands. The new dominating hybrid will have genes representing both parent islands. The success of recombination at the global level will depend on how successful recombination is at the local level.

Accepting migrants without mixing them with the locals (migrants dominating the locals) corresponds to selection and resembles a *deterministic crowding* setup, whereby one of the parents is replaced with an offspring. After such a migration, the target island population is gone and there only exist two “clones” of the source island.

By introducing new alleles and initiating local changes, mutation pushes an EA into a different trajectory. Evolution inside islands is of a local nature and as such it resembles mutation. Therefore, one can assume that at the inter-island level, local evolution may stimulate global evolution.

Whereas mutation works randomly, local evolution is directional and can have positive and negative implications. Local evolution may get stuck in a local optimum. At the same time, it may be easier to adapt unfit solutions. For example, if a combination of building blocks turns out to be a bit inferior due to some non-linearity, intra-island evolution could search the fitness landscape neighborhood for a fix to the problem.

Table 6.1: Local dynamics creates inter-island evolution.

| <i>local dynamics</i> | <i>inter-island EA component</i> |
|-----------------------|----------------------------------|
| migrants rejected     | drift                            |
| migrants mixed        | “recombination”                  |
| migrants dominate     | “selection”                      |
| evolution             | “mutation”                       |
| randomness            | diversity                        |

“Random initialization” at the inter-island level is caused by divergence in evolutionary paths due to stochasticity. In Chapter 7, we will see how heterogeneity may be another source of diversity at the inter-island level.

I have summarized the correspondence between local dynamics and inter-island evolution in Table 6.1. I believe that both intra-island and inter-island evolution is important, and these two levels of evolution should be properly balanced.

## 6.5 Summary

In this chapter we have seen that IM dynamics result from interactions at the level of genes. We have also seen how the dynamics of an island are a series of restarts from unstable island configurations (being mixtures of migrants and locals) just after migrations. Such configurations are not achieved in standard EAs, and are the source of increased evolvability. Furthermore, the local dynamics have a profound impact on the dynamics between islands, creating another level of inter-island evolution. I expect that the interleaved evolution at a local and global scale should be particularly visible for setups with many small islands.

This chapter gave us additional insight into the reasons behind parameters' influence on IM behavior. In IMs, we need to properly match island EA parameters with migration parameters. The key to understand their interaction is to realize that all of these parameters apparently determine what the chance for migrants is to dominate the target island, be rejected, or get mixed in with the local individuals. All of these processes, in turn, determine the inter-island evolution.

The choice of *selection pressure* has important implications on other parameters. Generally, it seems that the selection should not be too weak or too strong. Selection is needed to identify good offspring. If it is too strong (too exploitative), however, it will have trouble combining BBs from different islands and creating fit hybrids (*deltas*). If it is too weak (too explorative), it may fail to select emerging good hybrid individuals.

It is not clear how the maximum  $L$ , for which BBs can still be successfully combined, changes with selection pressure, but one can assume it must become smaller with a stronger selection; or in other words the probability of the *recombination operator* being constructive must be higher if we want to use a stronger selection.

We have seen that even good individuals can be rejected after migrations not because of their fitness, but because of a non-constructive recombination unable to merge their genes into the target island population. This is an important conclusion, because it shows that sometimes islands may be functionally more isolated than it appears. When individuals from different islands cannot be successfully combined, recombination is not scaled to the inter-island level and islands compete rather than cooperate by combining results.

Even though, if migrants are rejected, mixing genes may still occur as a long-term,



accumulated effect of migration. Longer BBs can more easily survive (and get mixed) in IMs because the recombination process is repeated many more times compared to the standard EA, due to multiple similar migrants and repeated migrations.

The optimal *migration size* for a successful combination of solutions depends on the selection pressure as well. Assuming migrants are better than locals, small migrations must be used to prevent a premature takeover of the island (and effectively a quick loss of global diversity). The stronger selection pressure, the smaller  $\alpha_0$  (migration size) should be.

The *migration interval* also plays an important role, and is also dependent on the type of EA, and in particular on the strength of selection which determines the time after migration needed for convergence (a full domination of migrants, a return to the locals' domination or a new stable evolutionary point). If the migration interval is shorter than the convergence time, then migration effects will easily sum up. Whereas this may be beneficial for local diversity, it may be bad for global diversity, making the system very sensitive and probably leading to a domination of migrants. On the other hand, choosing migration intervals longer than the island convergence time does not change the situation much, and prolongs computations by unnecessary generations. Simplifying, a stronger selection allows for shorter intervals, whereas a weaker selection requires longer intervals. Too weak selection will, however, make it much longer for the island population to stabilize, and as we have seen in some models, the system may even converge to a state where both migrants and locals are lost and unfit hybrids survive.

With migration intervals close to the island convergence times, both choosing random and best individuals for migration results in very similar groups because the

source island is converged. Therefore, in the  $\alpha\beta\gamma\delta$  models I have assumed that all migrants are the same. On the other hand, a strong *migration policy* may achieve the same result — namely a nearly homogeneous group of migrants — even if the source population has not converged yet. Therefore, I suspect that a strong migration policy can be used if migration intervals must be shorter for an arbitrary reason (like limitations due to the maximum amount of computational resources).

The dynamics that I have studied is to some degree independent from particular *migration topology*. I have analyzed an interaction already happening between two islands and topology only describes which islands out of many may interact. The influence that migration topology has is probably to some degree orthogonal to the influence of earlier described parameters.

## Chapter 7: Heterogeneity in island models

In previous chapters of this dissertation we saw that islands may successfully combine partial solutions. To effectively create new solutions the partial solutions must differ from each other, which is normally achieved due to stochasticity. However, in standard IMs there is no force at a global scale that would actively increase the inter-island diversity. It seems that we could ensure inter-island diversity by *forcing* different evolutionary paths in islands. This may be achieved by varying islands' characteristics, e.g., making them heterogeneous. So, heterogeneity could serve as such “diverging” force resulting in different evolutionary paths.

Heterogeneous models should also enhance the ability of the EA to successfully go around obstacles that cause a stagnation, switch goals, try several options and adjust the behavior in run-time. Simply because of their non-homogeneity, heterogeneous models are expected to verify multiple “paths” leading to solutions. As we will see, by using migration it is sometimes possible to alternate between such paths.

As observed earlier, in our two-level perspective on IM evolution, migrations convert inter-island diversity into local diversity. Therefore, heterogeneity “supplying” the inter-island diversity should be an improvement for IMs. Another possibility to increase inter-island diversity, not studied in this dissertation, would be systems that create (or isolate) new islands, and systems that inject new (possibly random) individuals into the system.

Heterogeneity can occur in various forms. In this chapter I focus on representation-based heterogeneity. While fitness-based or island EA-based heterogeneity are possible, they are not included in this dissertation.

## 7.1 Representation heterogeneity

I study a model for representation heterogeneity in which individuals are transformed from one representation to another during migrations. As in a standard IM, migrations occur at the same time between all islands. This model allows for changing representations in either direction, as opposed to Injection Islands GA, where the flow of individuals is one-way, from coarser to more precise representations. In other approaches (e.g. shifting) the representation changes for all individuals at the same time. A heterogeneous IM is more flexible because it allows the IM to have different representations simultaneously and to have separate control of representation for each individual. The general model I just described does not make any assumptions about the type of representations used or the number of islands.

We have seen that in standard IMs islands may not only spread good solutions, but also combine them to create novel ones, unknown to any of the contributing islands. Similarly, a representation-heterogeneous IM benefits from the best representation, and in addition may benefit if representations cooperate with each other to reach solutions unreachable for any of them alone.

I will first introduce a method for achieving heterogeneity, after which I will show two related studies. In the first study I show that a heterogeneous IM shares good characteristics of both representations used, and in the second study I will show that heterogeneous IMs have the potential to achieve better results than an EA with any

of the used representations. I will discuss reasons why heterogeneous IMs in general should have an advantage over single-representation IMs.

### 7.1.1 Heterogeneity by using exclusive or

In this section I will create a universal way to introduce representation heterogeneity on binary genomes. This is done by repeatedly using exclusive or function on selected genes.

Let us define an operation  $\rho : \{0, 1\}^2 \rightarrow \{0, 1\}^2$ :

$$\rho(x, y) = (x, x \oplus y),$$

where  $\oplus$  denotes the exclusive or (xor) operation. Obviously,  $\rho^{-1} = \rho$ . In modular arithmetic (modulo 2) we can represent this operation as a matrix

$$\rho = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

Further, let us extend the  $\rho$  function to a set of such functions  $\{\rho_{i,j} : \{0, 1\}^n \rightarrow \{0, 1\}^n, i \neq j\}$ , which will operate on the  $(x_i, x_j)$  the way  $\rho$  does, and copy the remaining arguments:

$$\rho_{i,j}(x)_k = \begin{cases} x_k & \text{if } k \neq j; \\ x_i \oplus x_j & \text{if } k = j. \end{cases}$$

The appropriate matrix would have values of 1 on the diagonal, and an additional value of 1 at the  $(j, i)$  position:

$$\rho_{i,j} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & \ddots & & & \\ & & 1 & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}.$$

Again  $\rho_{i,j}^{-1} = \rho_{i,j}$ . Reversing the order of indices is equivalent to transposing the matrix ( $\rho_{j,i} = \rho_{i,j}^T$  and thus  $\rho_{i,j} \neq \rho_{j,i}$ ).

Let us use a notation  $(f \circ g)(x) = g(f(x))$ . When applying a sequence of  $\rho_{i,j}$  functions, the order is important, since  $\rho_{i,j} \circ \rho_{j,k} \neq \rho_{j,k} \circ \rho_{i,j}$ . We can represent symbolically the application of  $\rho_{i,j}$  as a directed edge  $i \rightarrow j$  in a graph of all indices. The use of the arrow is justified by the fact that both  $i$  and  $j$  are involved, but only  $j$  is affected. We see that if an argument to one function is a result of another function, then swapping their order most likely will give different results. If the application occurs in order, we can create longer symbolic representations by going through all nodes involved. For the just mentioned example, we can write  $i \rightarrow j \rightarrow k \neq j \rightarrow k, i \rightarrow j$ .

For  $i \neq j, j \neq k, i \neq k$  we have the following relations, all of which can be easily checked by their definitions (symbolic representations on the right):

$$\rho_{i,j} \circ \rho_{j,k} \circ \rho_{i,j} \circ \rho_{j,k} = \rho_{i,k} \quad (i \rightarrow j \rightarrow k = j \rightarrow k, i \rightarrow j, i \rightarrow k),$$

$$\rho_{i,k} \circ \rho_{j,k} \circ \rho_{i,k} \circ \rho_{j,k} = id \quad (i \rightarrow k \leftarrow j = j \rightarrow k \leftarrow i),$$

$$\rho_{i,j} \circ \rho_{i,k} \circ \rho_{i,j} \circ \rho_{i,k} = id \quad (j \leftarrow i \rightarrow k = k \leftarrow i \rightarrow j),$$

where  $id$  is the identity function.

Since  $x \oplus x = 0$  for all  $x$ , any composition of the functions results in a matrix consisting solely of ones and zeros. However, each  $\rho_{i,j}$  function is a bijection with the determinant of the corresponding matrix always equal to 1, and therefore, not all matrices built from zeros and ones correspond to some sequence of  $\rho_{i,j}$  functions.

Finally, certain sequences of  $\rho_{i,j}$  functions give us well-known transformations:

- Swapping arguments  $i$  and  $j$  (or genes in a genome) can be achieved by the following sequence:

$$\text{swap}_{i,j} = \rho_{i,j} \circ \rho_{j,i} \circ \rho_{i,j} \quad (i \rightarrow j \leftarrow i \rightarrow j).$$

- The transformation from the standard binary to the standard reflected Gray encoding is equivalent to a series of  $\rho$  functions applied “backwards”:

$$\text{binary}^{-1} \circ \text{gray} = \rho_{n-1,n} \circ \rho_{n-2,n-1} \circ \cdots \circ \rho_{1,2} \quad (n \leftarrow (n-1) \leftarrow \cdots \leftarrow 1).$$

- The transformation from the Gray to the binary encoding is equivalent to a series of  $\rho$  functions applied “forwards”:

$$\text{gray}^{-1} \circ \text{binary} = \rho_{1,2} \circ \rho_{2,3} \circ \cdots \circ \rho_{n-1,n} \quad (1 \rightarrow 2 \rightarrow \cdots \rightarrow n).$$

The  $\rho_{i,j}$  functions are used to gradually transform representations, by applying them to genomes either in sequence, or in random order.

Let us take a series of representations  $R_k$  obtained by applying only first  $k$  of all  $(L - 1)$  required  $\rho_{i,i+1}$  functions, which transform binary into Gray representation. We have  $R_0 = \text{binary}$ , and  $R_{L-1} = \text{gray}$ . Intermediate representations ( $R_i : 0 < i < L - 1$ ) will have more significant bits still in binary representation and less significant bits in Gray representation, which probably results in a global behavior similar to that of a binary representation, and local behavior characteristic for Gray representation. Further, we can transform genomes between these intermediate representations also by applying an appropriate number of  $\rho_{i,i+1}$  functions, as follows:

$$R_i^{-1} \circ R_j = \begin{cases} \rho_{j-1,j} \circ \rho_{j-2,j-1} \circ \cdots \circ \rho_{i,i+1} & (j \leftarrow (j-1) \leftarrow \cdots \leftarrow i) & \text{if } i < j; \\ \rho_{i,i+1} \circ \rho_{i+1,i+2} \circ \cdots \circ \rho_{j-1,j} & (i \rightarrow (i+1) \rightarrow \cdots \rightarrow j) & \text{if } i > j. \end{cases}$$

### 7.1.2 The F and F2 functions

In this section I will construct special functions useful for experiments with representation heterogeneity. Let us denote the solution search space by  $\Omega$  and by  $G_n$  denote a graph, in which nodes are binary strings of length  $n$  and the edges represent the neighborhood relation between solutions (for a more formal definition and discussion see (Rowe *et al.*, 2004)). An encoding  $f$  is a function  $f : \Omega \rightarrow G_n$ . I will denote the binary and Gray encodings by  $\text{binary} : \Omega \rightarrow G_n$  and  $\text{gray} : \Omega \rightarrow G_n$ .

To construct the function, I first define two functions from  $G_n$  to  $\mathbb{R}$ .



Let us define a function  $\text{easy} : G_n \rightarrow \mathbb{R}$  by

$$\text{easy}(a) = \text{hamming}(a, 0^n)/4,$$

where the function  $\text{hamming}$  returns the number of bits by which its two arguments differ and  $0^n$  denotes a string of length  $n$  consisting of 0's. It is easy to see that it is a simple unimodal function with one global optimum equal to  $0^n$ .

Let us also define a function  $\text{deceptive} : G_n \rightarrow \mathbb{R}$  by

$$\text{deceptive}(a) = \begin{cases} 3 * \text{hamming}(a, 0^n) & \text{if } \text{hamming}(a, 0^n) < \frac{1}{4}n; \\ n - \text{hamming}(a, 0^n) & \text{otherwise.} \end{cases}$$

The function has two optima,  $0^n$  and  $1^n$ . Although both have the same value,  $1^n$  is easier to reach due to a larger basin of attraction, but  $0^n$  will be preferred later in combination with other functions because it represents the same point in both binary and Gray encoding. Hence the name “deceptive” function.

I further define a function  $F_1 : \Omega \rightarrow \mathbb{R}$  by

$$F_1(x) = \text{deceptive}(\text{gray}(x)) + \text{easy}(\text{binary}(x)).$$

For the binary encoding we have  $(\text{binary}^{-1} \circ F_1) : G_n \rightarrow \mathbb{R}$  given by

$$(\text{binary}^{-1} \circ F_1)(a) = \text{deceptive}(\text{gray}(\text{binary}^{-1}(a))) + \text{easy}(a),$$

and for the Gray encoding we have  $(\text{gray}^{-1} \circ F_1) : G_n \rightarrow \mathbb{R}$  given by

$$(\text{gray}^{-1} \circ F_1)(a) = \text{deceptive}(a) + \text{easy}(\text{binary}(\text{gray}^{-1}(a))).$$

The easy component of the function is usually smaller than the deceptive

component and has a smaller impact when using the Gray encoding. Nevertheless, it makes  $0^n$  to be the global minimum. When using the binary encoding, the easy component should mostly affect the overall evolutionary path of the algorithm, with the deceptive component creating a somewhat rugged landscape. Therefore,  $F_1$  is difficult for Gray encoding and relatively easy for binary encoding.

Similarly, I define a function  $F_2 : \Omega \rightarrow \mathbb{R}$  by

$$F_2(x) = \text{deceptive}(\text{binary}(x)) + \text{easy}(\text{gray}(x)).$$

Analogously,  $F_2$  is difficult for binary and relatively easy for Gray encoding.

Finally I construct the function  $F$  to be

$$F(x) = \min(F_1(x), F_2(x)).$$

The function inherits both local minima from  $F_1$  and  $F_2$  and has one global optimum at  $0^n$  with value 0. It is a difficult function when using only binary or only Gray encoding, but turns out to be much easier when switching between the two representations is allowed.

Because the  $F$  function turned out to behave somehow differently than expected (in particular, deceptive parts were dominating more than it seems from the construction above), I also used a simpler function  $F_2$ , defined below:

$$F_2(x) = \text{deceptive}(\text{binary}(x)) + \text{deceptive}(\text{gray}(x)).$$

The idea here was similar, namely the function is difficult when using a single representation, but when using both, it is easier to solve simply because what is

deceptive in one representation need not be deceptive in the other. When using a given representation, the F2 function is a combination of the deceptive part for this representation and a composition of the deceptive function and a transformation into the other representation. This second component produces multiple local optima.

The definition of both F and F2 may be extended to any two representations  $R_i$  and  $R_j$ , instead of binary and gray, as shown below

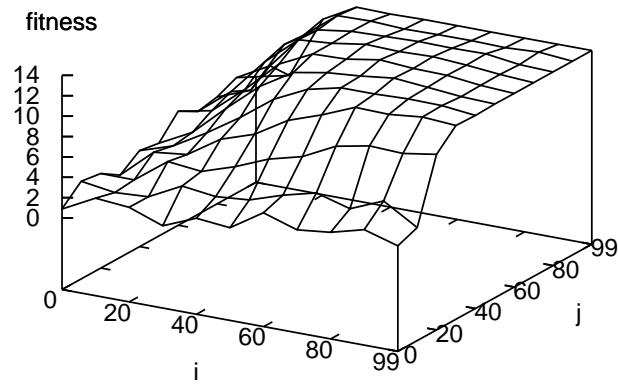
$$F_{R_i, R_j}(x) = \min(\text{deceptive}(R_i(x)) + \text{easy}(R_j(x)), \text{easy}(R_i(x)) + \text{deceptive}(R_j(x))),$$

$$F2_{R_i, R_j}(x) = \text{deceptive}(R_i(x)) + \text{deceptive}(R_j(x)).$$

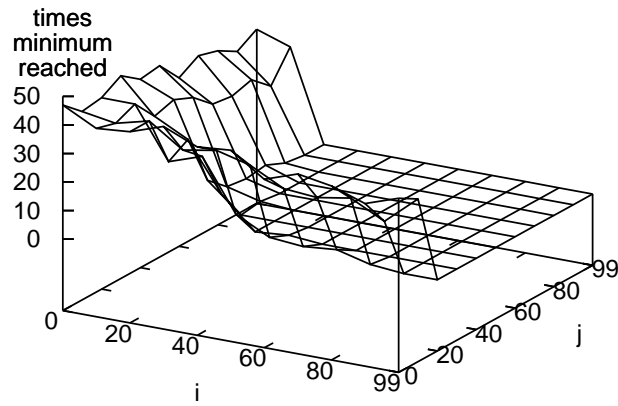
### 7.1.3 Better representation dominating

Using the heterogeneity model defined in Section 7.1.1 and functions defined in Section 7.1.2 I was able to run experiments with representation-heterogeneous IMs.

I used a 20x50 setup in which every second island used one representation  $R_i$  and the others used another representation  $R_j$ , and I varied both representations from a binary ( $R_0$ ) to Gray ( $R_{n-1}$ ), using  $\rho_{i,j}$  functions. This resulted in a combinatorial set of representation pairs. I performed experiments for both the F and F2 functions. The results for different migration sizes ( $\alpha = 0.02$  or  $\alpha = 0.1$ ) and different intervals ( $i = 10$  or  $i = 50$ ) were nearly identical, and hence I show charts for  $\alpha = 0.1$  and  $i = 10$  setups only. The remaining parameters were: L=100, binary parent selection, parameterized (0.2) uniform crossover at a rate 0.8 and binary tournament migration policy.

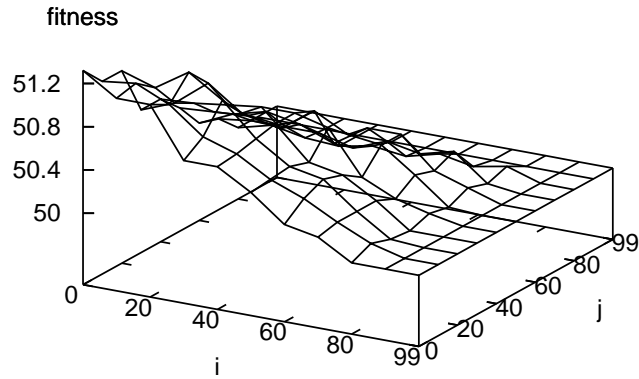


(a) average best value, function F

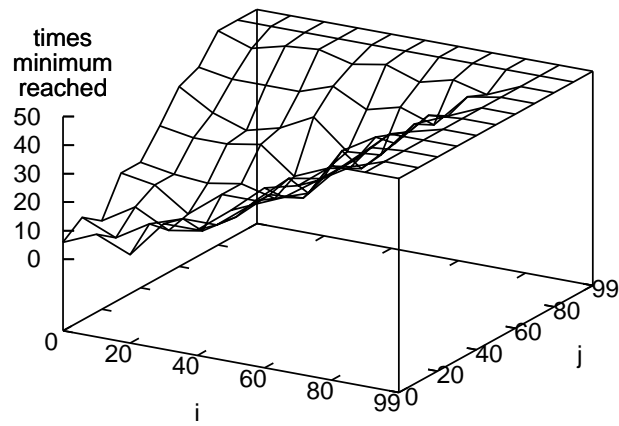


(b) How many times the global minimum (zero) was achieved, function F

Figure 7.1: Performance when gradually changing representations (for both axes 0=binary, 99=Gray).



(c) average best value, function F2



(d) How many times global minimum (zero) was achieved, function F2

Figure 7.1: Performance when gradually changing representations (for both axes 0=binary, 99=Gray).

In Fig. 7.1, we see results for 50 runs. As expected from 7.1.2, function F is much easier for binary representation and function F2 is much easier for Gray representation. Nevertheless, in both cases we can observe two things. First, the ability to solve the function changes gradually with gradually changing representation (see the change of minimal fitness on a diagonal, that is for the same representation in all islands). Second, it is enough if one island has an appropriate representation to successfully find the global optimum, because it is later shared with the other.

While I did not observe heterogeneous IMs outperforming standard EA models in these examples, the reason may be that in both functions one of the representations occurred good enough to solve the problem. In the next section we will see how islands cooperate, which in general may lead to an overall better performance.

#### **7.1.4 Representations cooperating**

To show the “cooperation of representations” we<sup>1</sup> used two islands, one with a standard binary encoding and one with a standard reflected Gray encoding. I start with initial single-run observations, and later I present the results of a more systematic set of experiments, in which we differed the parameters of the model and tested the model against other functions.

##### **Initial observations on single runs**

Not surprisingly, single populations using just one encoding usually failed to find the global optimum on the F function. A typical run for the binary encoding is shown in Figure 7.2a. and a typical run for Gray encoding is shown in Figure 7.2b. I plot the

---

<sup>1</sup>This subsection is based on a paper published earlier (Skolicki and De Jong, 2004).

island’s best fitness. The length of genomes was  $L = 100$ . In both cases the EA got trapped in local minima, which are neighborhoods of respectively  $\text{binary}^{-1}(1^L)$  and  $\text{gray}^{-1}(1^L)$ .

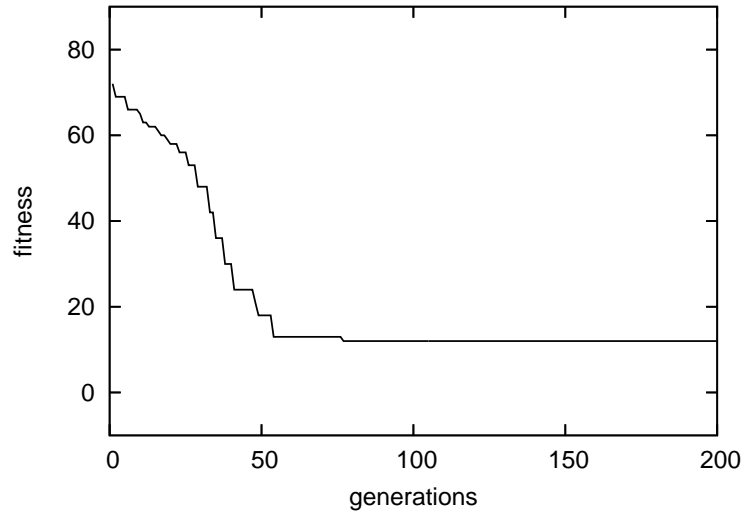
The situation changes with a heterogeneous IM. In Figures 7.3a and 7.3b, I show typical runs with two islands using two representations. Abrupt changes in best fitness correspond to migrations from the other island. We can see that the representations “help each other”, by not allowing each other to get stuck in local minima.

When using an interval of 5, one may notice that it is the binary representation that is leading in the beginning but it is the Gray representation that plays an important role towards the end. The influence of such migrations is very distinct in generations 20–35 for the Gray encoding and in generations 55–65 for the binary encoding.

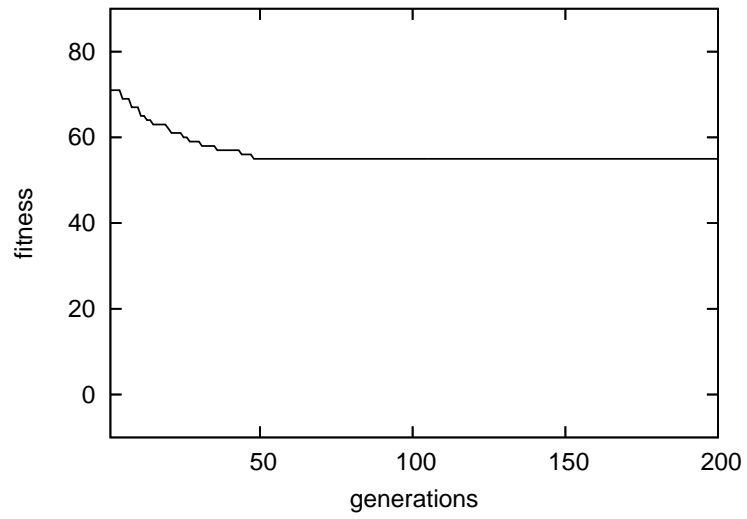
When using an interval of 50 one can see a very distinctive mutual influence of islands by means of migrations. The island with binary representation initially leads. By generation 50, the Gray encoding island is already stuck at its local minimum. A migration helps it escape from this minimum. Later, the roles of the islands switch.

## Experiments

I have compared a two-island, heterogeneous IM (each population of size 50) with single-population models of size 100, for both binary and Gray encoding. I have tested the model using two crossover types (two-point and uniform) and two selection strategies (ranked with elitism and binary tournament). The island model performed better — it was the only one to find the global optimum and it was statistically either better or comparable in terms of average island-best fitness. In particular



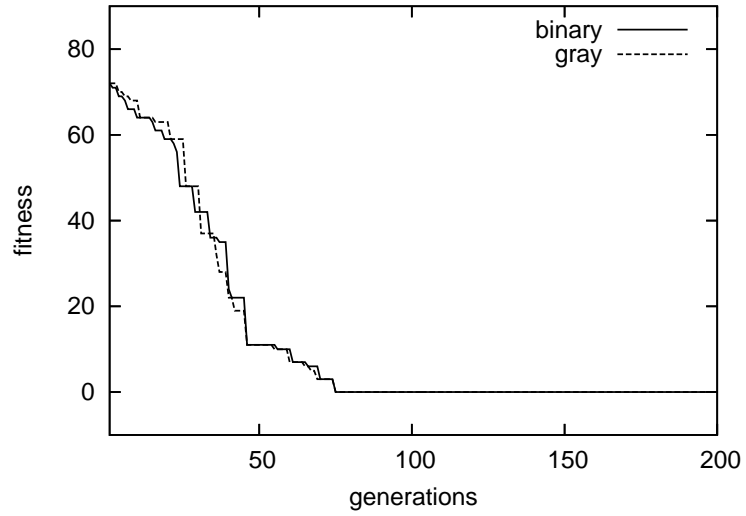
(a) Binary



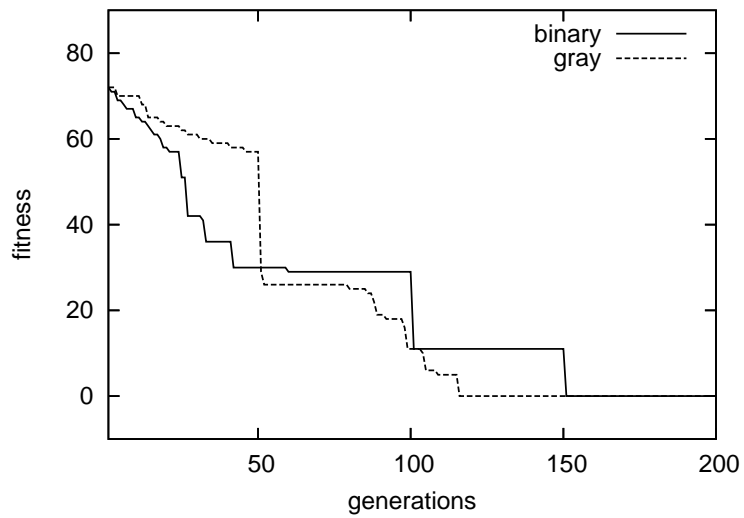
(b) Gray

Figure 7.2: A typical single run with single encodings on the function  $F$ . EA got trapped in local minima.





(a) interval=5



(b) interval=50

Figure 7.3: A single run with both encodings on the function  $F$ , with different migration intervals (each line corresponds to one island). EA finds the global optimum.

Table 7.1: Parameters used.

| <i>parameter</i>               | <i>value</i>  |
|--------------------------------|---|
| mutation rate                  | 1/L   |
| recombination type             | two-point, <i>uniform</i>   |
| recombination rate             | 1.0   |
| selection strategy             | ranked with elitism, <i>bin. tournament</i>                                 |
| island size                    | 100 (binary), 100 (Gray)<br>and 50+50 (binary+Gray)                         |
| migration policy               | copy best — replace random  |
| migration size                 | <i>0.02</i> , 0.1, <i>0.4</i>   |
| generations between migrations | 1, 5, <i>20</i>   |
| total number of generations    | 500   |
| number of runs                 | 50  |
| function                       | F, <i>Rastrigin</i> , <i>Rosenbrock</i> , <i>Schwefel</i> , <i>Griewank</i> |

for the uniform crossover and ranked selection, the island model found the global optimum every time.

Additional parameters used in the experiments are shown in Table 7.1. Parameter values listed in italics were additional ones tested to verify the robustness of the model (recombination operator + EA selection strategy, migration size + interval and fitness landscape).

The results of experiments are presented in Table 7.2. The confidence interval (for level 0.05) is given for reference only, as the calculations assume normal distribution and the distribution of local optima of the function is quite different. Heterogeneous IMs performed either better or comparable to single-population models.

A similar analysis was performed for different migration sizes and intervals, and the results are given in Table 7.3. The first three configurations maintained the same

Table 7.2: Changing EA parameters.

| <i>description</i>  | <i>binary</i>  |             | <i>Gray</i>    |             | <i>island model</i> |             |
|---------------------|----------------|-------------|----------------|-------------|---------------------|-------------|
|                     | <i>average</i> | <i>%opt</i> | <i>average</i> | <i>%opt</i> | <i>average</i>      | <i>%opt</i> |
| two-point, ranked   | 21.08±4.99     | 0%          | 55.00±0.00     | 0%          | 22.00±7.54          | 60%         |
| uniform, ranked     | 12.00±0.00     | 0%          | 55.00±0.00     | 0%          | 0.00±0.00           | 100%        |
| two-point, b.tourn. | 25.38±5.76     | 0%          | 55.00±0.00     | 0%          | 27.50±7.70          | 50%         |
| uniform, b.tourn.   | 12.00±0.00     | 0%          | 55.00±0.00     | 0%          | 3.48±2.16           | 38%         |

total number of individuals exchanged per generation, and configurations number 1, 2 and 5 maintained the same migration size. In all cases a heterogeneous IM was able to find the global optimum in a considerable percentage of runs, as opposed to the experiments with single-population models.

Again, because the EAs converge to several very different optima, the averages and confidence should be treated as an approximation only. In all experiments the global optimum was found relatively often. For experiments with  $\alpha = 0.1$  and  $i = 20$ , I observed a bigger variance of solutions: the global optimum is found in 56% cases, but the average optimum is high (equal to 30.8). This means that there are very different outcomes possible, either very good (and hence a large percentage of good individuals), or very bad (and hence the high average). This may suggest that with long intervals islands either help each other, or both get stalled.

The last experiments were aimed at verifying the applicability of the model for multi-modal functions used in the literature. Three multimodal functions were used, the Rosenbrock, Schwefel and Rastrigin functions.<sup>2</sup> Ten-dimensional versions were

<sup>2</sup>Results for a fourth function, reported as Griewangk in (Skolicki and De Jong, 2004) are omitted due to an implementation bug discovered later.

Table 7.3: Changing migration parameters.

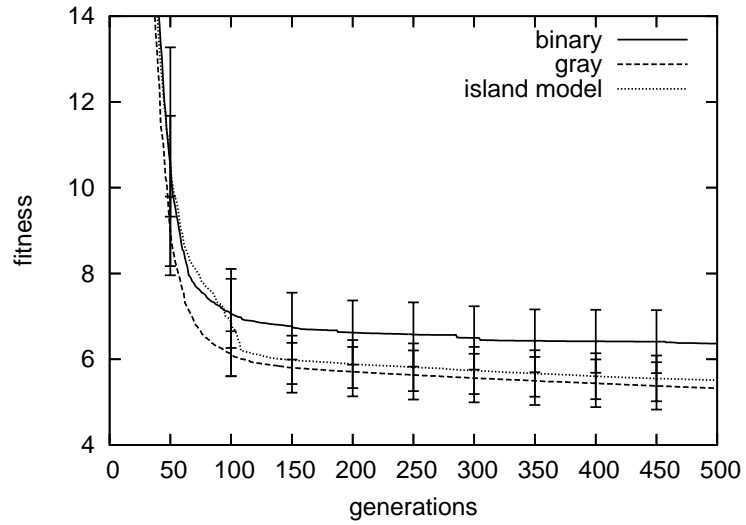
| <i>description</i> | <i>average</i>  | <i>%opt</i> |
|--------------------|-----------------|-------------|
| $\alpha=0.1, i=5$  | $22.0 \pm 7.54$ | 60%         |
| $\alpha=0.02, i=1$ | $20.9 \pm 7.48$ | 38%         |
| $\alpha=0.4, i=20$ | $18.7 \pm 7.3$  | 34%         |
| $\alpha=0.1, i=1$  | $17.6 \pm 7.19$ | 32%         |
| $\alpha=0.1, i=20$ | $30.8 \pm 7.65$ | 56%         |

used and each parameter was represented with 12 bits, resulting in  $L=120$ . The mutation rate was set to  $0.0083 \approx 1/L$ . Figure 7.4 shows the results (confidence level of 0.05), in which we see that heterogeneous IMs behave either better or similarly to single representation models.

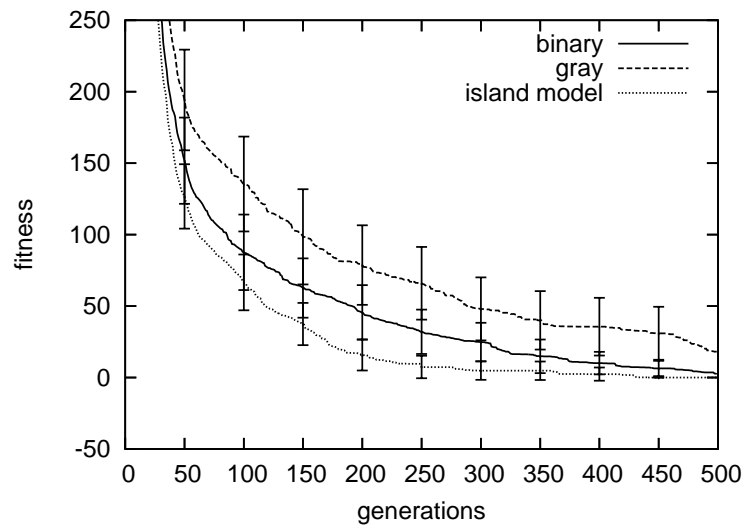
## 7.2 Discussion and conclusions

The results presented in this chapter show that representation-heterogeneous islands have an ability to cooperate. First, we saw that heterogeneous islands spread good solutions. Therefore, the representation which is better in terms of performance, dominates. Later we saw how islands help each other to move out of a local optimum, even when all of them separately converged to different local minima.

Whereas in homogeneous IMs one would need a novel recombination of migrants and locals to escape from a local minimum, in heterogeneous IMs this may happen just due to different locations of local minima in another representation. Solutions that cause evolution to stall in one island, may continue being evolved after migration in another island. Therefore, recombination of solutions from different islands may

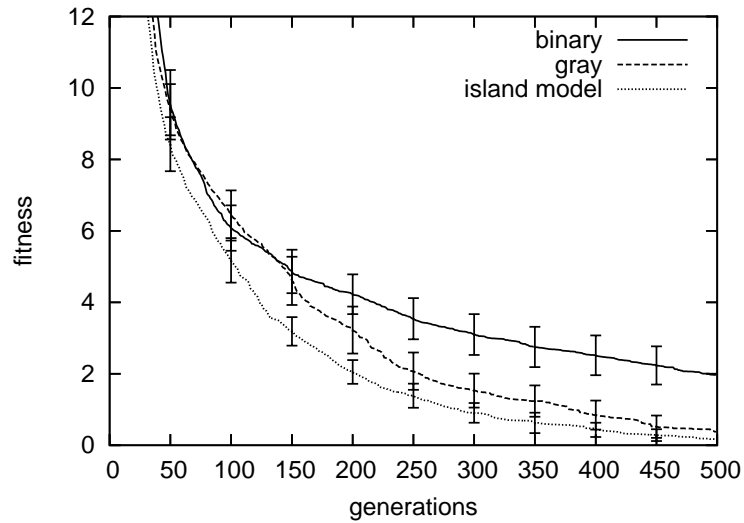


(a) Rosenbrock



(b) Schwefel

Figure 7.4: Average results from 50 runs on standard multimodal functions.



(c) Rastrigin

Figure 7.4: Average results from 50 runs on standard multimodal functions.

seem less important, and mutation-based gradual evolution may play a bigger role. This ability to increase evolvability without using a recombination may be important for domains in which designing a constructive recombination operator is difficult. In these situations the global-level evolution would resemble evolution processes studied in evolution strategies and heterogeneity would then serve to increase exploration at the global level.

The ability of heterogeneous islands to cooperate seems to enhance the natural ability of all IMs to solve difficult, deceptive and multi-modal problems. While the No Free Lunch theorem states that the results can only be valid for a subclass of all problems, I think that many real-life engineering domains fall into this subclass due to their modularity.

Although I showed very selected examples of representation-heterogeneous IMs, I believe that the idea of heterogeneity is worth exploring wider as a general concept.

It would be interesting to see how the model behaves with other representations (e.g. real-valued encodings), or when representations are very different and some solutions are representable only in one of the representations. This would require special transformation procedures.

There are other possible ways of introducing IM heterogeneity meanwhile keeping the same representation in all islands, including heterogeneous fitness landscapes or varied EA parameters. An example would be diversification of the selection pressure. We have seen that a weak selection and longer migration intervals allow for more chances for recombination to create good offspring. These settings, however, are worse for finding BBs initially. Using different selection pressure in islands could relieve the above problem.

A hypothetical cooperation of islands could be the following: Weaker selection islands at first would serve for the exploration the solution space. Stronger selection islands would then focus on good regions, tweak solutions' details and isolate good BBs. Then, weaker selection islands could be used again for successful mixing of individuals. In fact, one could even design a (possibly asymmetric) topology to force such behavior, similar to some existing heterogeneous setups.

Heterogeneous islands may reach a higher inter-island diversity because each island follows its own evolutionary path. For the same reason, the inter-island diversity can be regained, even after convergence. This ability is very important in the context of two-level evolution because the inter-island evolution needs some source of diversity. It would be interesting if this property could be studied in abstraction of the heterogeneity type. Quite possibly, however, each type of heterogeneity needs to be analyzed separately because they differ so much between each other.

To summarize, in this chapter we saw that heterogeneous IMs are powerful models, which are not a simple sum of their components (e.g., islands with various representations) but benefit from interaction of these components. The dynamics of heterogeneous IMs could be perceived as one step further toward more flexible evolutionary systems, compared to the homogeneous IM behavior. Such systems, even if using imperfect components, could achieve more than a carefully designed, single-component approach.



## Chapter 8: Exemplary problem analysis

So far we have seen experiments and analysis of IMs on a small set of functions. This was chosen deliberately for simplicity. The functions were carefully chosen, to illustrate various behaviors. However, what we are really interested in is how IMs behave on real functions.

In this chapter I will show a few experiments with more complex problems, including an engineering problem, and I will use a full-fledged IM. I will try to explain the behavior of IMs and suggest improvements, based on so-far analysis.

A short note about IM design is needed. While in this chapter I experiment with various problems, I still choose to focus more on the analysis of IMs, rather than on finding the very best solution for the given problems. For example, in real problem solving situation it would be advisable to avoid unnecessary drift (see guidelines in Appendix F). However, a typical implementation of an IM (which I want to study) is not optimal from a theoretical point of view. Therefore, even though I have shown that bidirectional migrations are beneficial (by avoiding drift), from now on I will use unidirectional migrations. Implementing bidirectional ones requires much more communication between two islands and if they reside on separate machines, exchanging migrants can be quite complex (one would need to keep track of each migrant origin). Similarly, I will use a one-child recombination, even though a two-children case maintains diversity longer. In certain cases recombination is biased to choose the “better” parts of each parent. In these cases, producing the second

children could be questionable. Also, traditionally a single child is produced by a recombination operator. Those decisions should not qualitatively change the behavior of IMs.

## 8.1 Problems well suited for island models

What are the “good” problems for IMs? In other words, for which problems should we use IMs? Based on the research in previous chapters, this should happen for two (related) types of problems, corresponding to two modes of evolution in an IM: incremental and compositional.

The first type of problems are irregular multimodal functions. For these type of functions, there is a high chance that a better optimum will be found by one of multiple islands. A good individual representing such optimum would then spread onto other islands and dominate worse solutions.

The second type of problems are separable or modular functions, for which different subcomponents can be discovered in different islands and then combined together after migration. Functions that cause a rejection of migrants are bad candidates for compositional evolution because they do not allow for effective interaction between islands. Therefore, it is important to understand under which conditions migrants are accepted.

In most of the experiments we’ve seen so far (see Chapter 4) recombination was always very constructive and successful. Experiments with constructive recombination were used on purpose, so that we could observe IM behavior (and compositional evolution) when it is not hindered by unnecessary obstacles. As we have seen in Chapters 4, 5 and 6, if recombination is successful at the local level,

it scales up to the global level. However, when recombination is not constructive, IMs behave differently and achieve good results by incremental evolution, even if the problem itself would seem modular. After all, not only must the problem be suitable for decomposition, but representation and operators also must match the structure of the problem.

## 8.2 Simple functions

First, I will experiment with some simple functions. They are all either modular, or can be made modular by switching to binary representation.

### 8.2.1 Asymmetric quadratic function

Let us start the analysis from a very simple quadratic function, as used by De Jong (De Jong, 2006). It's given by a formula  $f(x) = \sum_1^L x_i^2$ , and defined over an asymmetric domain of  $[-3, 4]^L$ . At the first sight, this quadratic function seems quite similar to the IM6 function used so far, so we could expect a similar behavior. As we will see however, the quadratic function turns out to be much simpler for standard EAs, and therefore, IMs have little advantage if any. Nevertheless, experiments with the quadratic function serve as an interesting comparison and allows to identify typical IM behaviors.

In Fig. 8.1, we see the results of running EA-1 and EA-2 on the quadratic function. The parameters used were  $\alpha=0.1$ ,  $i=10$ , random-random (group) policy, full (dynamic) topology, uniform crossover, 1x100 panmictic population or 10x10 IM/isolated islands,  $L=20$  (to make the problem more difficult for IMs), mutation at

a rate of  $1/L$  and with  $\sigma = 0.1$  of domain range. The panmictic case and the IM case are comparable and better than the isolated case.

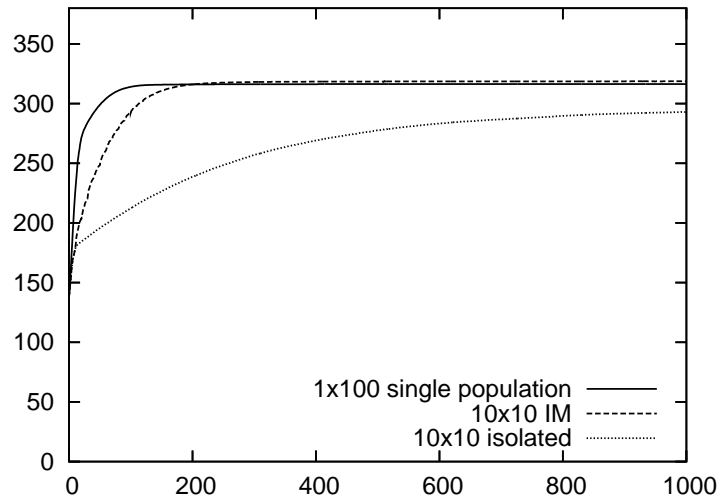
I repeat the same experiments with a binary representation, for which each gene was represented by 17 bits, to ensure a  $10^{-4}$  precision. This makes  $L = 340$ , and I set mutation to  $0.003 \approx 1/L$ . The results are shown in Fig. 8.2 and again we see that the panmictic and IM cases are comparable.

A good performance of the panmictic case means several things. There must be no real benefit in postponing recombination and selection, as in IMs. Early recombination plus fast selection of good results are apparently enough to obtain good results. Mutation must be able to effectively improve solutions.

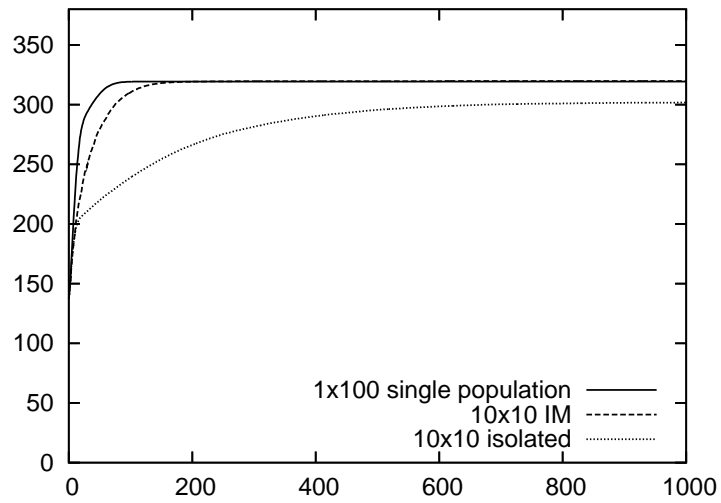
In Fig. 8.3, I show fitness increase and diversity drop using EA-1. When we look at Fig. 8.3a we see that even after global diversity drops in the panmictic case, the fitness is still improving considerably. In fact, for real-valued genes we see a subtle change in the slope of fitness. This may correspond to the end of a phase in which recombination is able to mix good solutions.

For the quadratic function, early recombination is beneficial because it makes optimization of each gene more independent. There is no need to prevent a selection that is occurring too quickly, or to find and recombine BBs in IMs because there is no linkage between genes. Preserving certain combinations by fixating them in islands therefore is unneeded. A good gene value will always positively contribute to an individual's fitness, regardless of the other gene values. Another reason for a good performance on the quadratic function is that higher peaks have somewhat bigger basins of attraction, which may also make it easier to find them.

There is a difference between the real-valued and binary representations when we

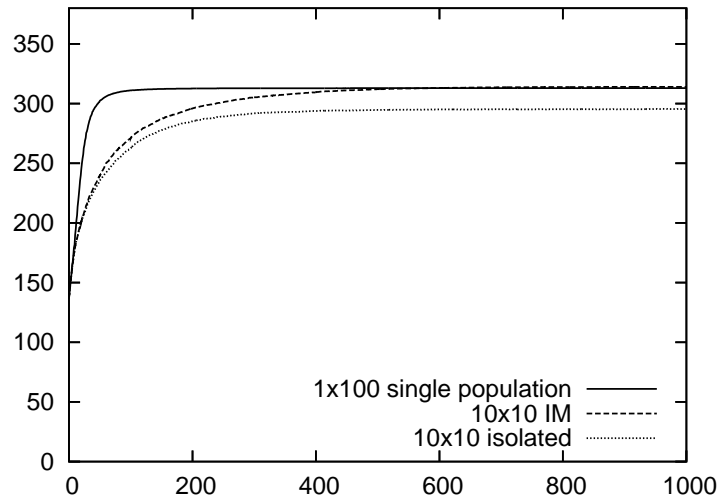


(a) EA-1

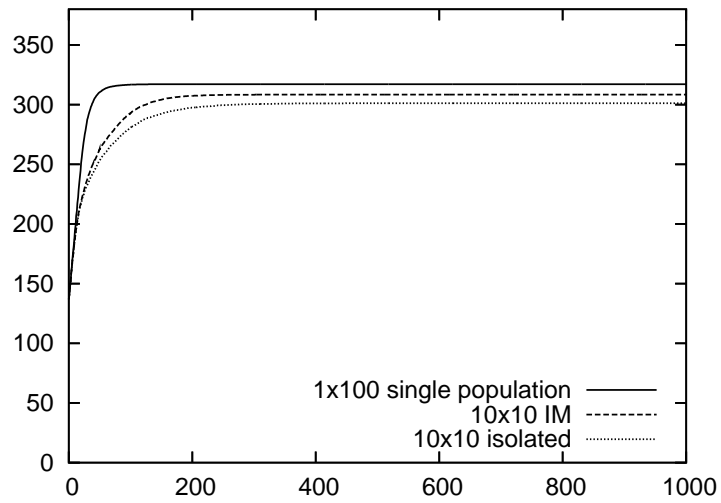


(b) EA-2

Figure 8.1: Asymmetrical quadratic function. Best fitness, real-valued representation.

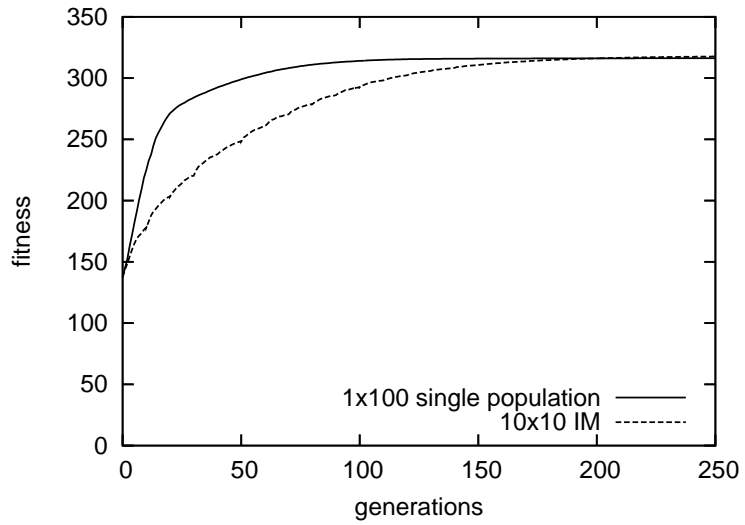


(a) EA-1

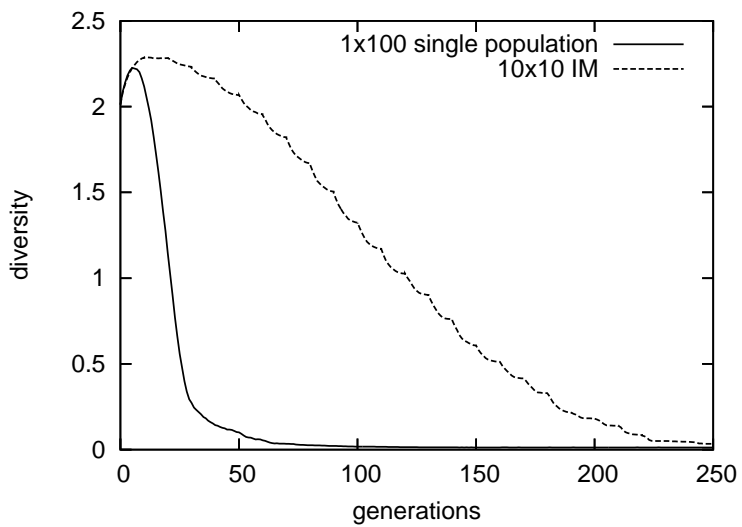


(b) EA-2

Figure 8.2: Asymmetrical quadratic function. Best fitness, binary representation.

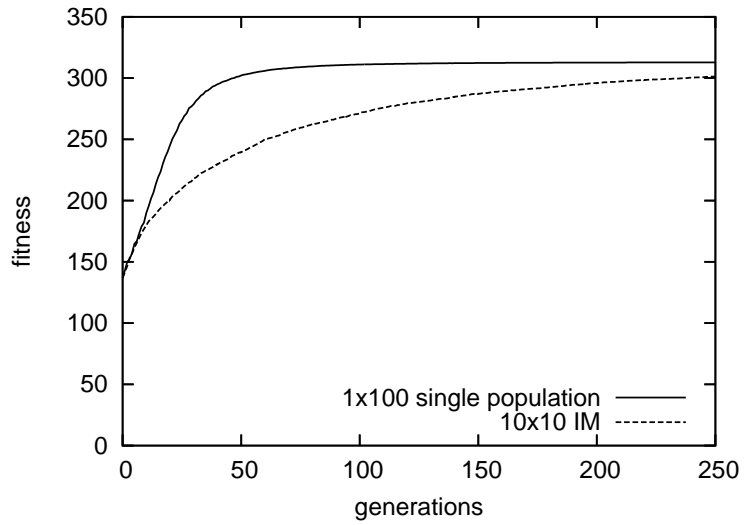


(a) fitness, real-valued representation

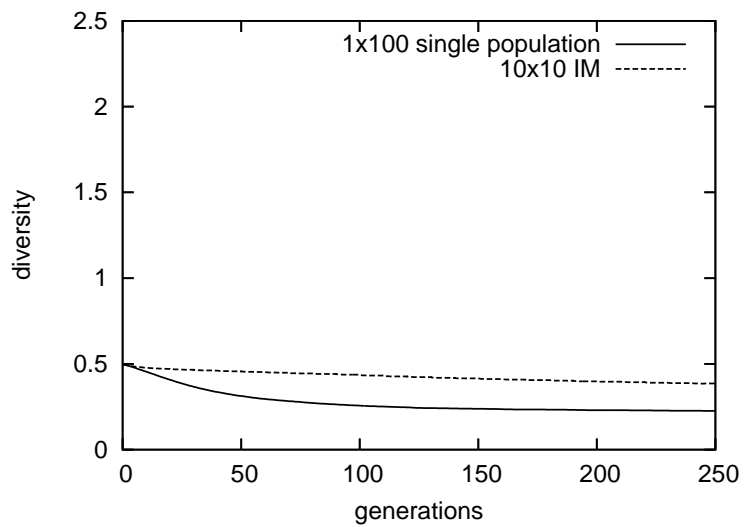


(b) diversity, real-valued representation

Figure 8.3: Asymmetrical quadratic function. Fitness grows even after global diversity drops significantly. EA-1.



(c) fitness, binary representation



(d) diversity, binary representation

Figure 8.3: Asymmetrical quadratic function. In the panmictic case, fitness grows even after global diversity drops significantly. EA-1.



look at global diversity. In the first case, it ultimately falls to zero, and in the latter it remains higher (and although not shown, diversity does not drop to zero until 1000 generations). This suggests a difference in internal IM behavior for real and binary encodings. In the first case solutions apparently exchange faster, whereas in the latter one, we observe less mixing between islands resulting in high global diversity.

A possible explanation for this behavior is that in the first case compositional evolution dominates, and in the second case, incremental evolution is more important. Charts in Fig. 8.4 and 8.5 show the survivability of migrants' genes in target populations, and confirm this hypothesis. In the case of real-valued representation, migrants are accepted, while for binary representation, migrants are rejected.

In Fig. 8.5, I show selection intensity. In each chart one line represents the changes due to evolution in islands (between migrations), and the second line represents changes due to migration events. Again, for real-valued representation, both migration- and evolution-based selection intensity are positive, which confirms that fitness grows as a result of migrations. For binary representation, we observe sharp negative peaks in evolution-based selection intensity following migrations. This means that fitness drops as a result of mixing migrants and locals, and migrants are eliminated.

In one case islands cooperate, and in the other they compete, acting on their own. Nevertheless, as already mentioned, in both cases good fitness is finally evolved. Note that the cooperative behavior occurs when recombining is relatively easy (each gene in real-valued representation is a module), and mutation is difficult (gaussian mutation with relatively small  $\sigma$ ). The competitive behavior occurs when recombining is harder (multiple genes per module in binary representation), but mutation is easier

(possible “jumps” across the domain when high-order bits are mutated).

### **8.2.2 The IM6 function**

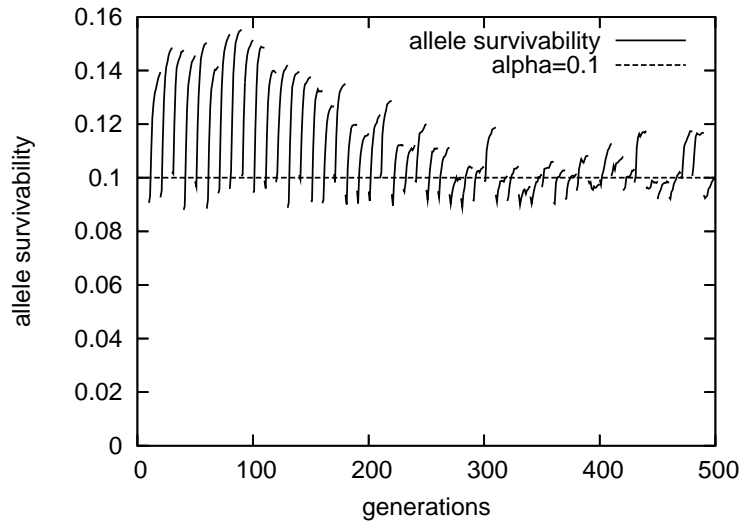
The IM6 function used throughout the dissertation so far is an example of a separable function, meaning that each variable can be optimized independently (although, of course, it is not linearly separable). Using a binary encoding, in a way analogous to the quadratic function, however, makes this function modular with one module being a block of bits representing one variable.

It turns out that IM performance on IM6 function is more sensitive to changes in representation than it was in the case using the quadratic function. The performance drops when I switch to the binary encoding (and incremental evolution). I discuss the differences between the functions below and argue why incremental evolution does not work well for IM6. We will then see how compositional evolution may be achieved, and results improved, by changing the recombination operator. I used four different setups for these experiments, and the difference between them will be illustrated with various analyses of internal dynamics.

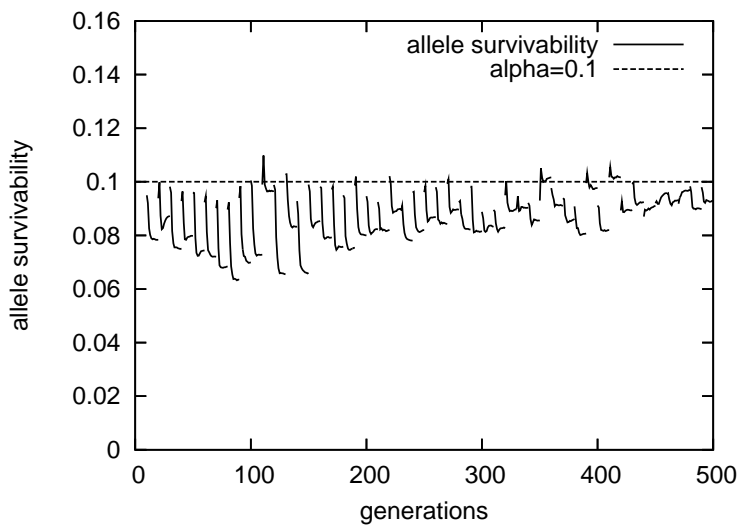
#### **Comparison to the quadratic function**

Although the quadratic and IM6 functions may look similar, there are a few differences between them that make incremental evolution (also in a standard, panmictic EAs) perform poorly on the IM6 function. I discuss the differences below.

Basins of attraction of high peaks in IM6 are much smaller than for the quadratic function. This makes it more difficult to randomly start from a high peak. Similarly, a recombination of random individuals has a smaller chance to “hit” a higher peak.

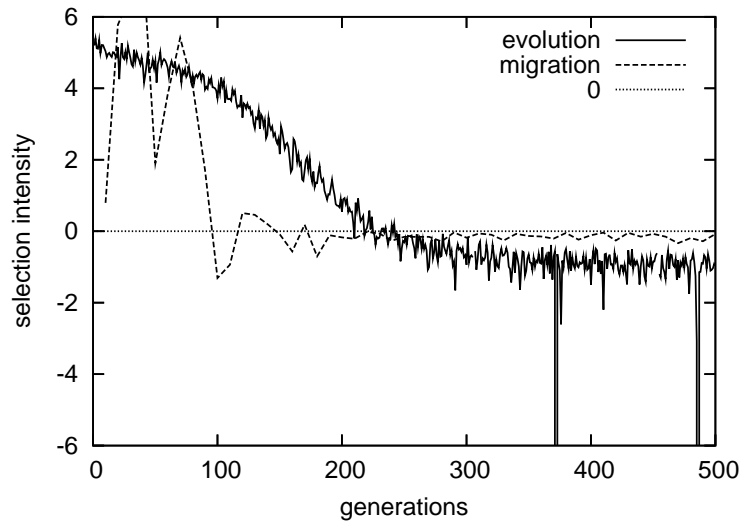


(a) real-valued representation

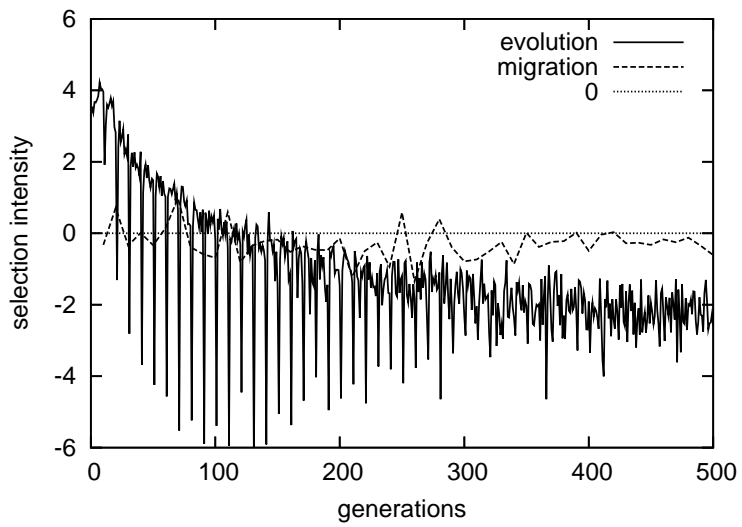


(b) binary representation

Figure 8.4: Asymmetrical quadratic function. Migrant alleles survivability differs between representations. EA-1.



(a) real-valued representation



(b) binary representation

Figure 8.5: Asymmetrical quadratic function. Selection intensity confirms rejection of migrants for binary representation. EA-1.

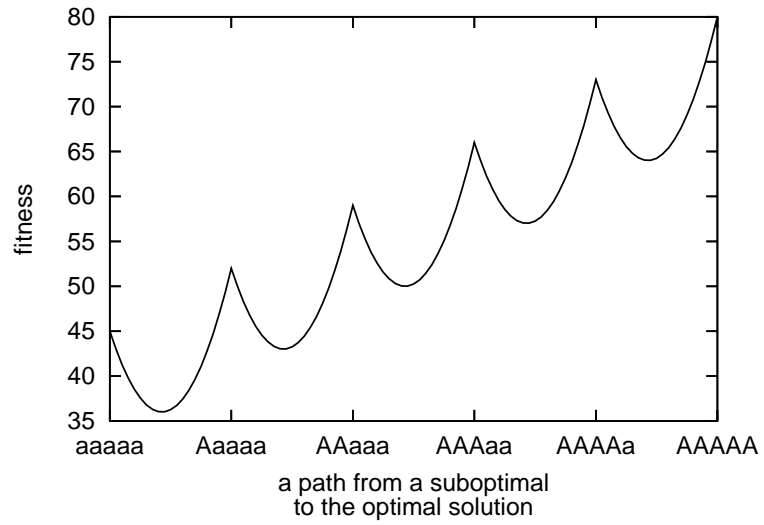
Another difference is the decrease in fitness when moving gradually between peaks. This difference is not important when optimizing a single variable and using rank-based selection, but it becomes important when optimizing multiple variables simultaneously. Both functions have 2 peaks for each argument — one lower (let's denote its value by  $a$ ) and one higher (let's denote its value by  $A$ ). We have  $a = -3$ ,  $A = 4$  for the quadratic function, and  $a = 0$ ,  $A = 1$  for the IM6 function. The lower peak is deceptive and the higher one belongs to the optimal solution. Imagine a path starting at  $(a, \dots, a)$ , and switching one module after another toward  $A$  (in arbitrary order). In the case of the quadratic function, each switch requires passing a relatively shallow “valley” in fitness landscape. In the case of IM6 the switch requires going through a point of zero fitness. This is shown in Fig. 8.6.

If recombination can successfully combine sub-optimal solutions for both functions, evolution can easily proceed. However, if recombination is destructive, mutation must be able to generate good offspring. For the quadratic function, it is much easier to escape from deceptive peaks toward the global optimum using mutation, but in the case of IM6, mutation — and thus incremental evolution — is less successful. Therefore, to solve IM6, compositional evolution must occur.

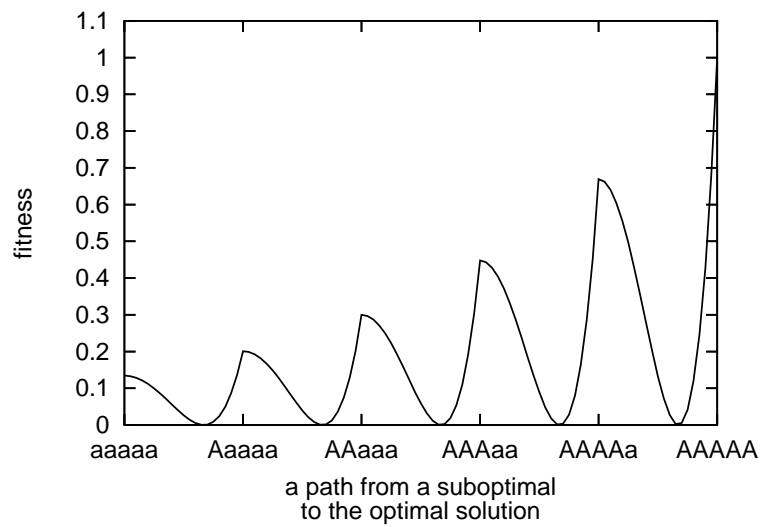
## Experiments

I ran a set of four experiments, in which individuals may be represented and recombined in different ways, as listed below:

- a) real values + uniform recombination.
- b) 10 bits per gene + uniform recombination.



(a) Asymmetric quadratic



(b) IM6

Figure 8.6: The IM6 function is more difficult for mutation than the asymmetric quadratic function, here shown for five dimensions.

- c) 10 bits per gene + uniform recombination preserving gene/BB boundaries.
- d) 10 bits per gene + two-point recombination.

The reason behind using these setups is the following. In the setup a) we observe a good increase of fitness, in periods after migration. However, in the setup b) such an increase is not observed. Since compositional evolution is important for the IM6 function, the lack of improvement must be due to the fact that uniform recombination in binary representation does not preserve genes, so migrants have little chance to mix with locals. To prove that this is the reason, in the setup c) I change only recombination, so that it now preserves gene boundaries (it copies blocks of 10 bits). As we will see, the behavior will resemble the one from the setup a) again. In fact, the only difference between the setup a) and the setup c) is that mutation may change each gene more randomly, as opposed to a gaussian perturbation of a certain radius. In the setup d) I use a more commonly used two-point recombination, which is much more linkage-preserving than a uniform recombination.

I use settings similar to EA-1, with binary parent selection and a non-overlapping model, but the rate of recombination is 0.7 (so, 0.3 fraction of offspring is created by cloning and mutation) and elitism is used. I analyzed a case of five islands, each of 20 individuals, connected by a full topology. The migration size is 0.01, migration interval is 20 and migrants are chosen randomly. I analyze a case with 10 genes and the mutation is set to  $1/L$  (being 0.1 and 0.01 for real-valued and binary representation respectively). For real-valued representation the  $\sigma$  parameter (for gaussian mutation) is 0.1.

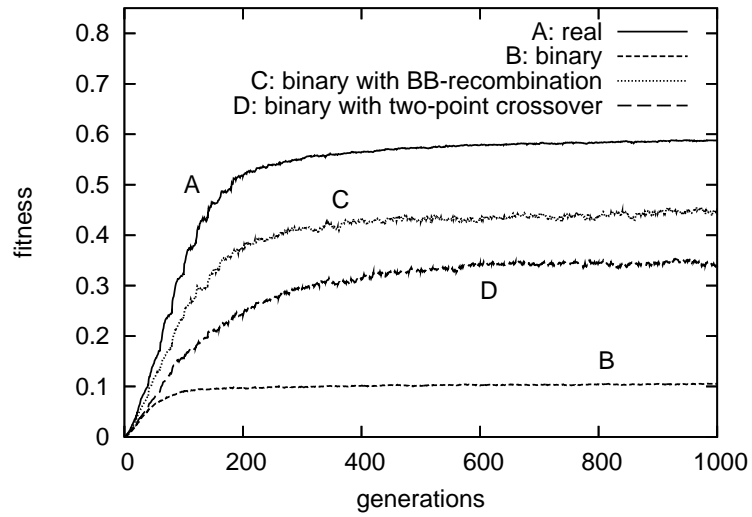
Let us analyze the differences step by step. In Fig. 8.7a, fitness increase for all four cases are shown. As mentioned, while the setup a) produces good results, we see that switching to binary encoding in the setup b) yields bad results. The setup c) produces good results again, and the setup d) is also relatively good in terms of performance.

In Fig. 8.7b, analogical plots for local diversity are shown. Diversity measured on binary genomes is different than that measured on real value genomes, so the setup a) is not directly comparable with the setups b), c) and d). Nevertheless, we see that the migration impact on diversity is similar in the cases a), c) and d) — diversity drops relatively fast to a point of stabilization. In the setup b), which achieved the worst results, diversity stays high for the longest time.

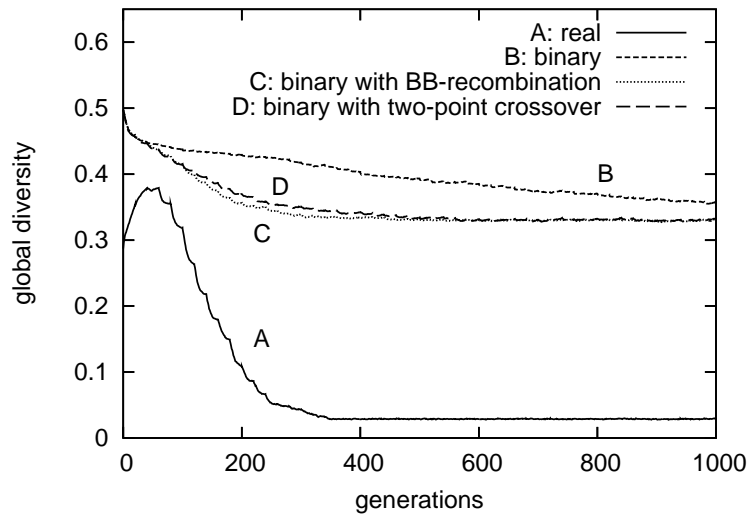
Because each variable for the IM6 function has a local optimum in 0 and a global optimum in 1, I can plot the average value of genes as an indicator of convergence toward either local or global optimum (the closer they are to 1, better it is). The same applies to the binary representation because 1 is represented with all genes set to 1, and 0 is represented with all bits set to 0 (although some play a more important role than others). Not surprisingly, for the setups a), c) and d), the average gene value is growing, which means that evolution on average moves to peaks closer to the optimum, and recombination is effectively exchanging sub-solutions. Such evolution practically does not happen in the case b). This is shown in Fig. 8.8.

The good performance of the setup c) and d) must be due to recombination because this was the only thing I changed from the setup b). Let us measure how much individuals/islands are mixed with each other. In Fig. 8.9a, we see that recombination is successfully mixing individuals, possibly coming from the same island. In case of



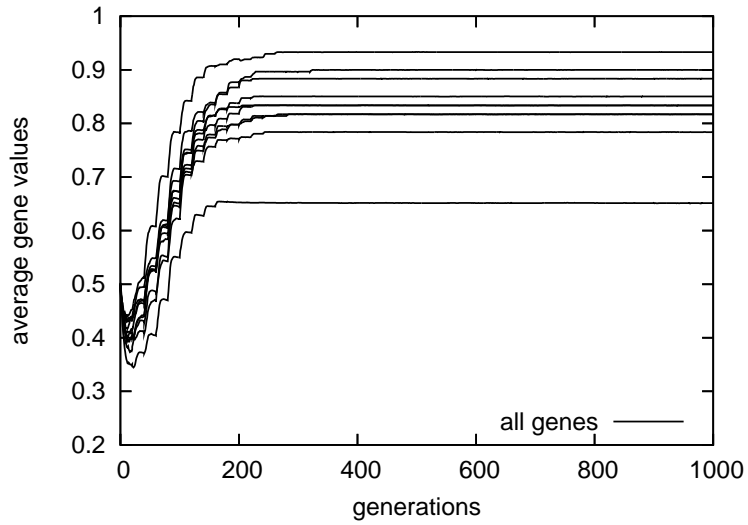


(a) Difference in performance

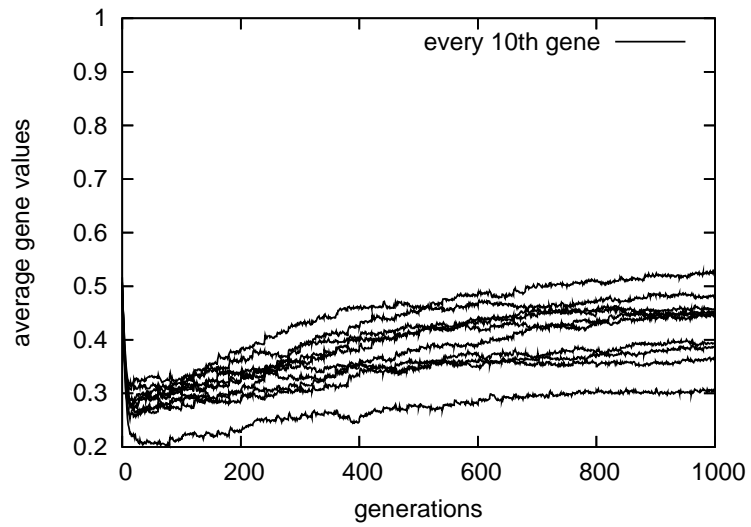


(b) Difference in convergence

Figure 8.7: IMs behavior depends on representation and recombination, the IM6 function.

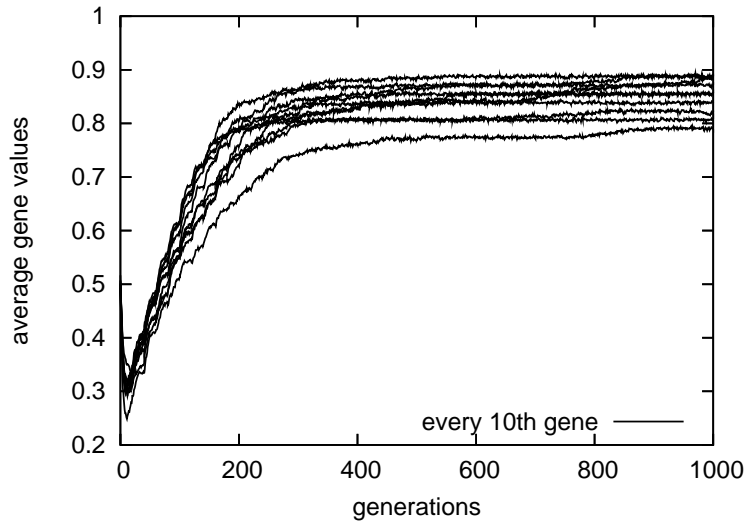


(a) real value genomes

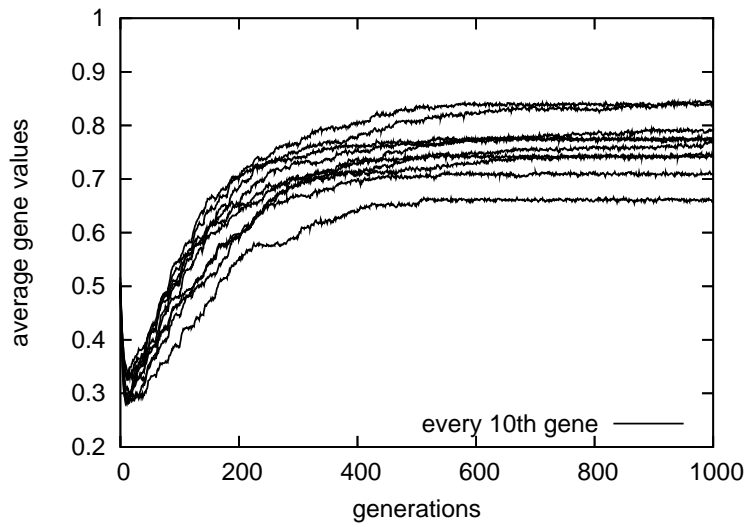


(b) binary genomes

Figure 8.8: Average gene value, the IM6 function.



(c) binary genomes + block-preserving crossover



(d) binary genomes + two-point crossover

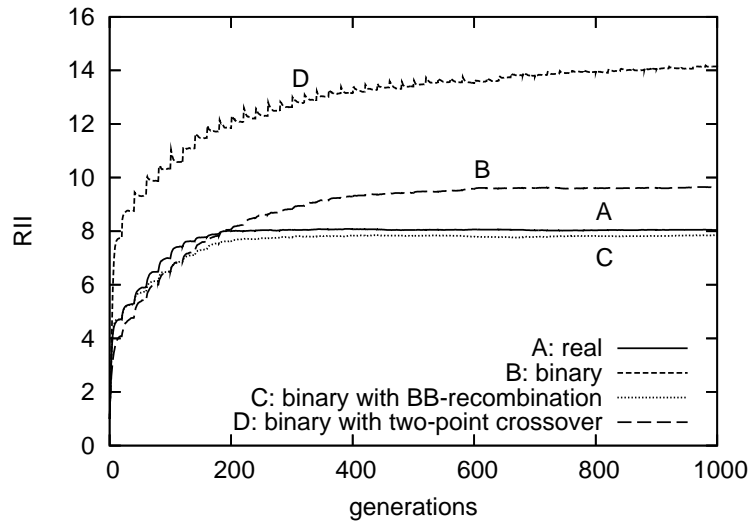
Figure 8.8: Average gene value, the IM6 function.

the setup b), the values are higher, because there are 100 loci to store alleles from other individuals. In case of the setup c), because of moving whole blocks, we have in fact 10 regions for storing gene values from other individuals. The setup d) shows a slow but steady increase in mixing. However, Fig. 8.9b shows an opposite effect for mixing islands. The b) setup keeps islands separated, compared to other setups. What this means in terms of the fitness function, is that local optima are maintained longer, and better solutions are not composed from sub-solutions. Comparing the four cases, we can see that a good recombination operator scales up and makes it possible to “recombine” islands, whereas a bad recombination operator disables compositional evolution both locally and globally.

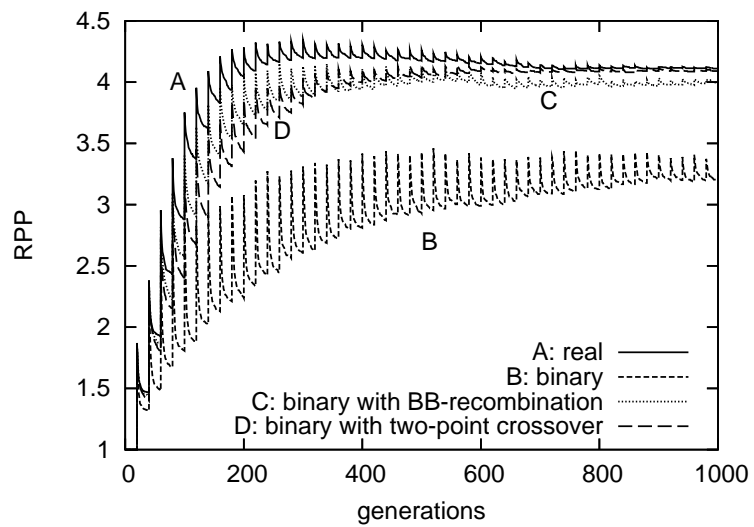
Studying selection intensity plots gives us yet another perspective. As we see in Fig. 8.10, the three “successful” cases a), c) and d) are comparable. For the b) case there are negative impulses for evolution-based selection intensity following migration, meaning that even though migrations themselves carry potentially good individuals, their genes get rejected.

These observations about survivability of migrants can be confirmed directly by measuring this survivability. I show results of two measurements. The first one measures the percentage of migrants alleles in the target population. Again, we clearly see that it is growing for the a), c) and d) cases and quickly dropping for the b) case in Fig. 8.11. Allele survivability is measure independently for each migration interval, and thus its value repeatedly moves back to the percentage resulting from the migration size, but the percentage of alleles being measured would keep evolving.

The second measurement may be a little difficult to understand. For each migration, I waited until the end of the migration interval and verified whether an

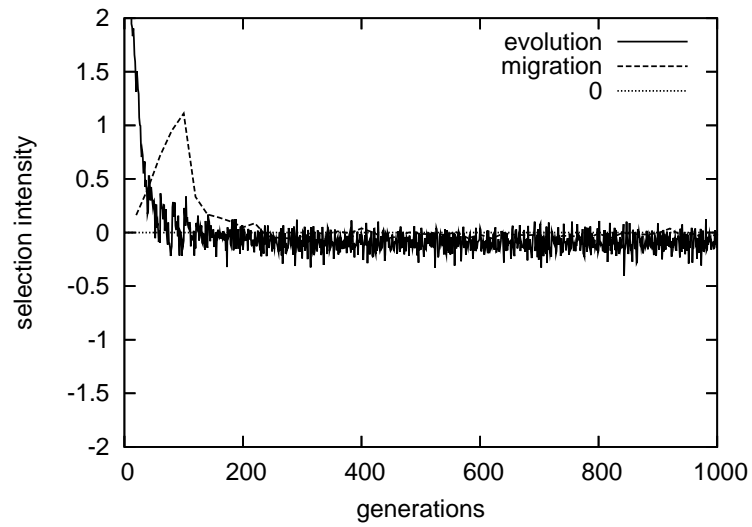


(a) RII

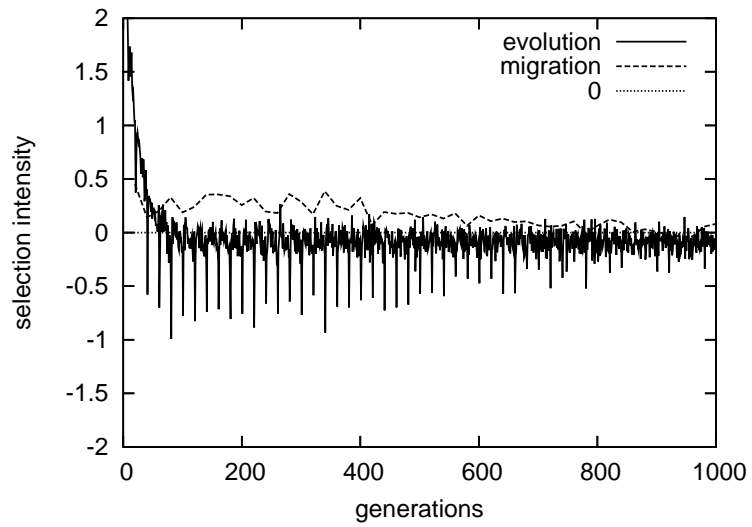


(b) RPP

Figure 8.9: Extent to which individuals are the effect of mixing alleles between individuals and between islands; the IM6 function.

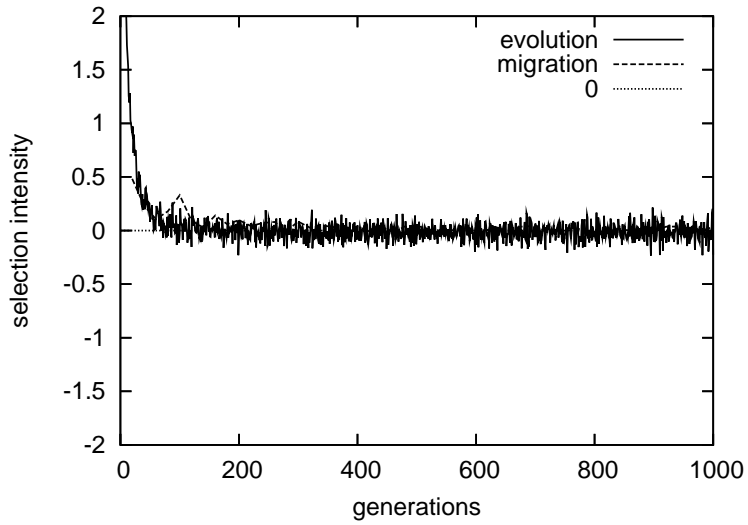


(a) real value genomes

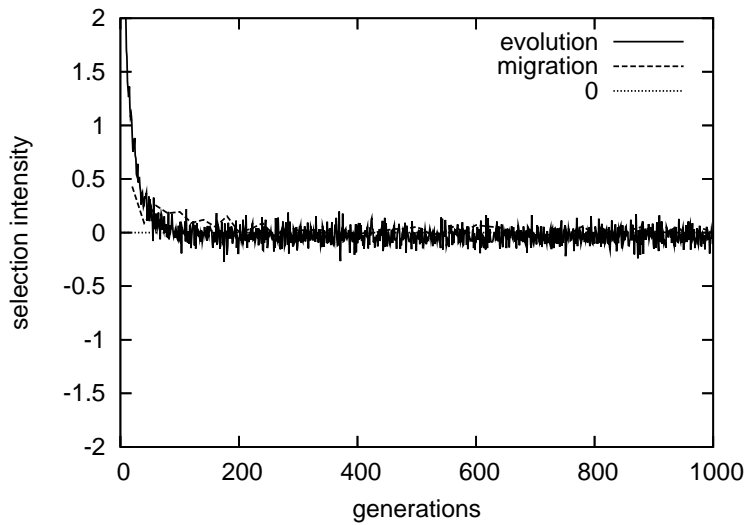


(b) binary genomes

Figure 8.10: Selection intensity, the IM6 function.

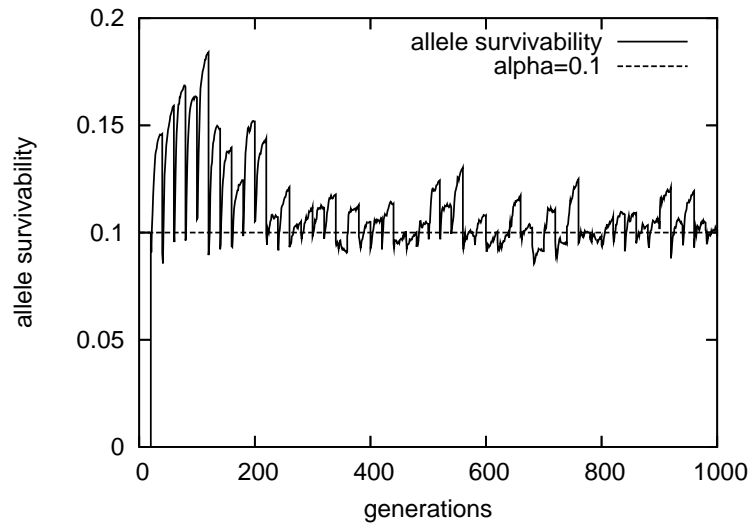


(c) binary genomes + block preserving crossover

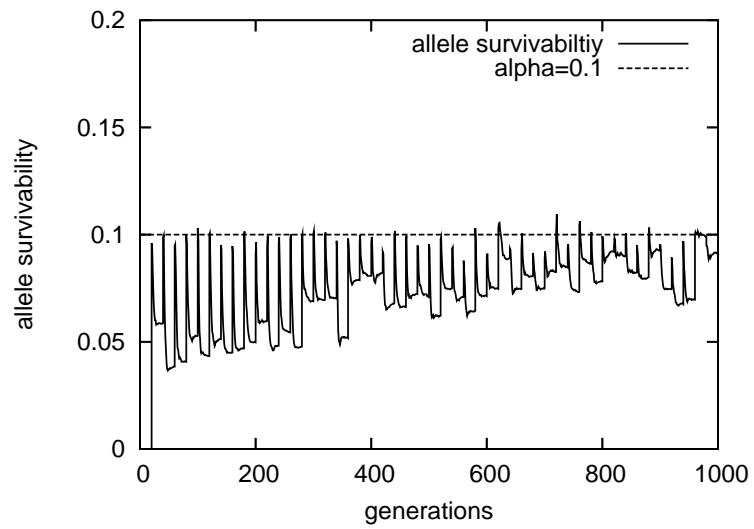


(d) binary genomes + two-point crossover

Figure 8.10: Selection intensity, the IM6 function.



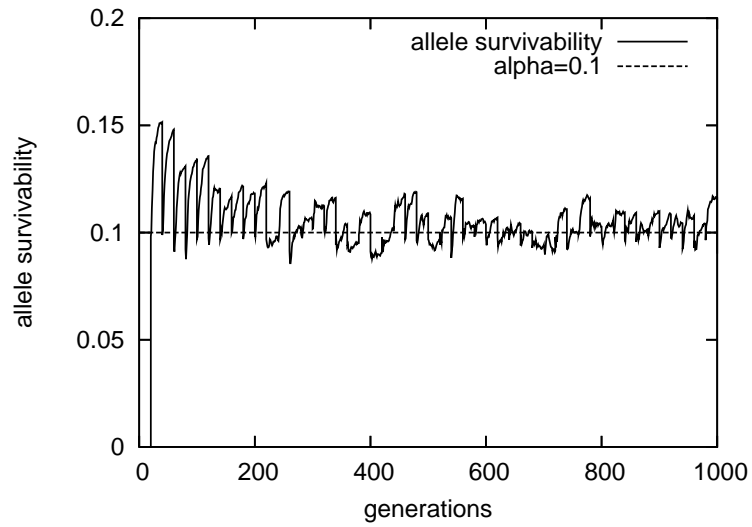
(a) real value genomes



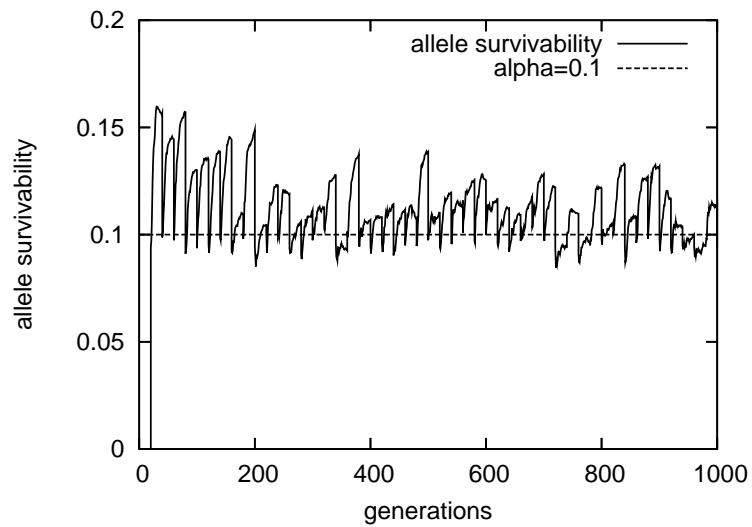
(b) binary genomes

Figure 8.11: Allele survivability, the IM6 function.





(c) binary genomes + block preserving crossover



(d) binary genomes + two-point crossover

Figure 8.11: Allele survivability, the IM6 function.

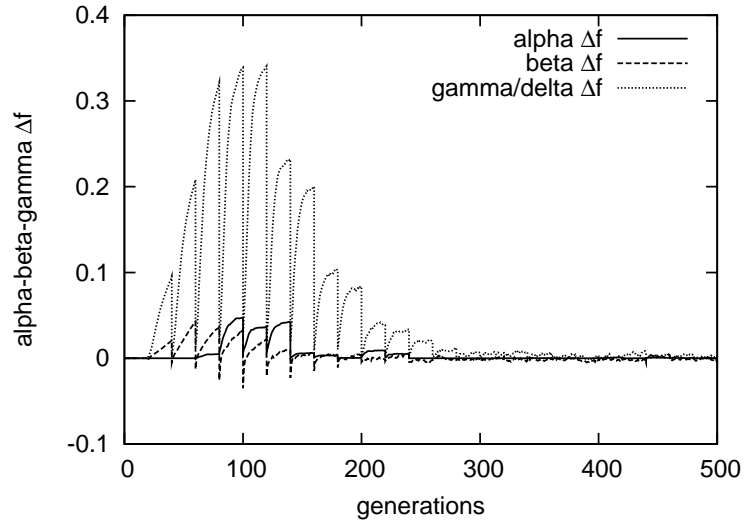
island converged back to locals (*betas*, using Chapter 6 terminology), converged to migrants (*alphas*), or converged to some hybrids (*gammas* or *deltas*). Then, the change in fitness in the whole interval was credited to one of those types, and results were averaged over multiple runs. In each migration interval, the change in fitness is measured with regard to its value at the beginning of the interval, so even when fitness grows continually under this measurement, this is reflected as repeated increase from 0.

What this tells us is what type of internal behavior is responsible for fitness increase. If we see high values for gamma/delta, as in Fig. 8.12a, then we know that mixing locals and migrants (or inter-island “recombination”) is the source of fitness increase. If we see a high value for beta, as in Fig. 8.12b, then it means that rejecting migrants is the best behavior (and this occurs only as long as locals are able to maintain evolvability without the help of migration). For the cases c) and d) we again see that inter-island compositional evolution increases overall fitness.

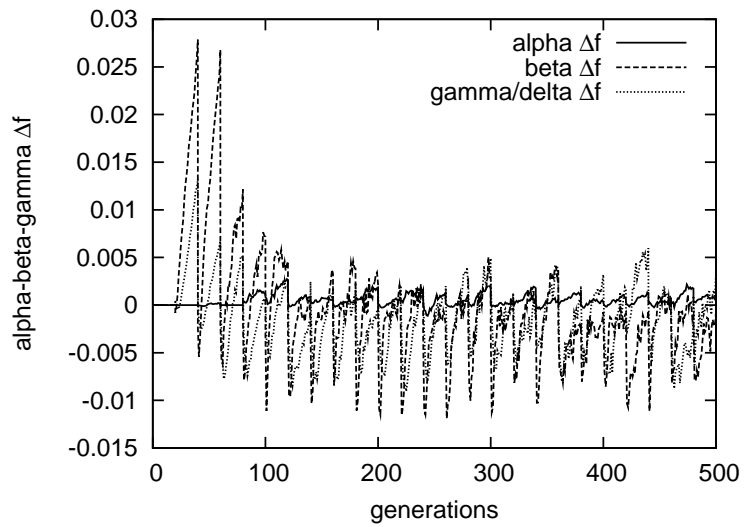
All these observations confirm that for modular problems which are difficult to solve gradually, effective recombination occurring after migrations is critical because it scales up to the inter-island level. It also shows how we could improve the algorithm by adjusting the recombination operator based on observation of internal dynamics of an IM.

### 8.2.3 The H-IFF function

After we have seen experiments with the quadratic and IM6 function, I will now compare the IM behavior on two modular functions. This section describes

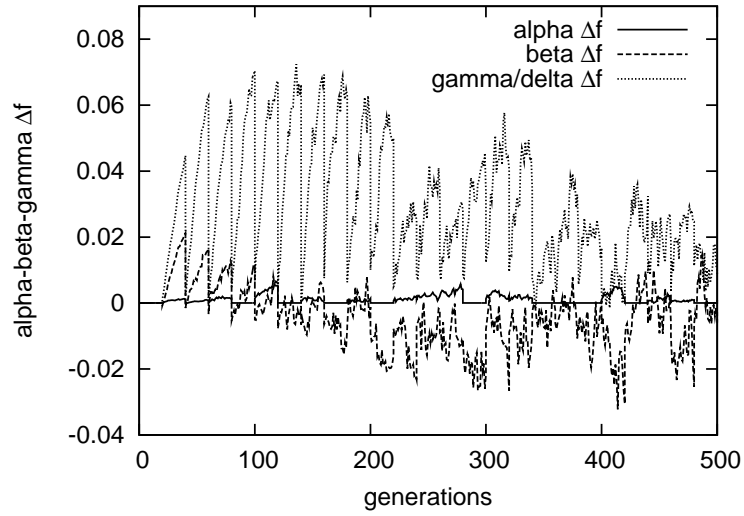


(a) real value genomes

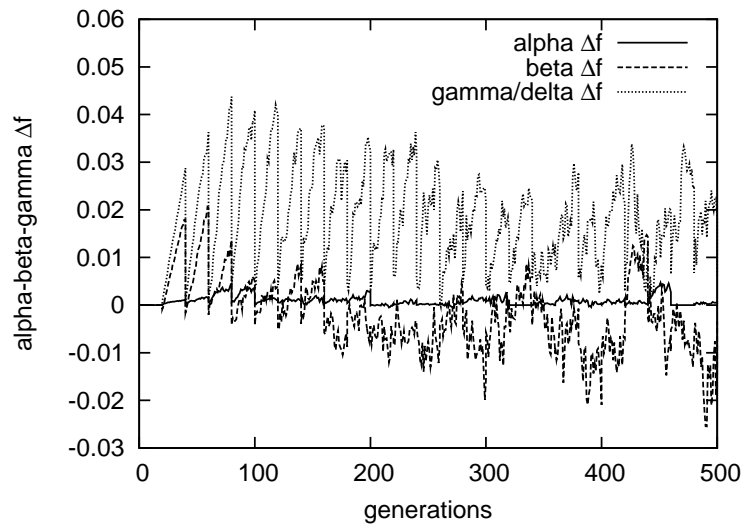


(b) binary genomes

Figure 8.12: A change in average fitness when alpha, beta or gamma/delta dominate (note a different scale on the y-axis), the IM6 function.



(c) binary genomes + block preserving crossover



(d) binary genomes + two-point crossover

Figure 8.12: A change in average fitness when alpha, beta or gamma/delta dominate (note a different scale on the y-axis), the IM6 function.

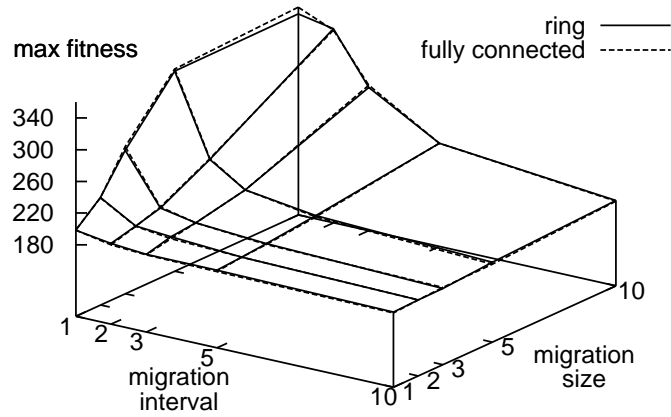
experiments with the H-iFF function and the next section describes experiments with the so-called Deceptive function.

Because the H-IFF function is hierarchical, we may expect good results using IMs on this function, due to compositional evolution, if we properly choose the recombination operator. I have chosen a two-point recombination operator, because genes that are linked in H-IFF are located close to each other. Similar to experiments in chapter 4, I have performed an extensive combinatorial set of experiments using five parameters: migration size, interval, policy, topology and islands selection pressure. I used a 10x50 IM setup. In this section I report only selected, representative results.

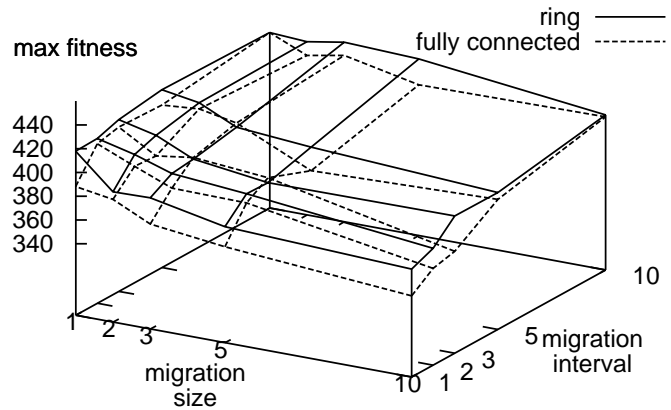
First, notice how migration, in the absence of normal selection pressure in islands, may serve as an additional selection pressure. Therefore, for EA uniform selection we observe the best results with a high level of selective migration. In Fig. 8.13a, we see a mean best-in-run fitness for this situation and for different sizes and intervals of migration. This behavior fits into incremental evolution description, although frequent and strong migration effectively creates one large population made up of individuals from all islands, rather than making islands compete to evolve the best solution. It is hard to imagine reasons for a compositional evolution in such setup, without local selection creating proper BBs.

With a normal EA selection on the islands, it is more beneficial to use migration to occasionally exchange genetic material than to send often migrations, which carry the best individuals. In Fig. 8.13b we see that the best best-in-run value is achieved for sporadic migrations.

I will show that this higher fitness comes from compositional evolution by comparing two opposite IM setups, using the H-IFF function, and a strong selection



(a) “Additional selection” role of migration. Uniform selection, 5-tournament migration policy.



(b) “Genetic material exchange” role of migration. Binary tournament selection, binary tournament migration policy.

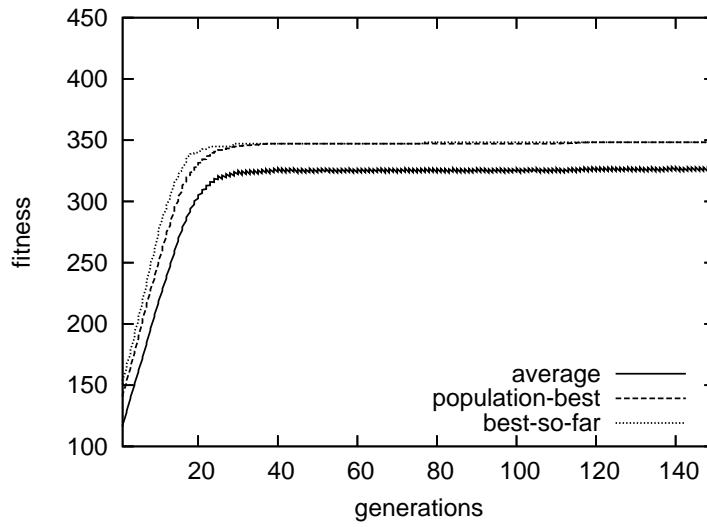
Figure 8.13: Two roles of migrations — additional selection pressure, or exchanging genetic material, the H-IFF function (note a different axis layout).

pressure (5-tournament). The first setup is a migration of size 10 and occurs in every generation, and the second setup is a migration of size one, and occurs every 10 generations. These setups differ in the level of interaction between islands, which, as we will see, is not equal with the level of cooperation between islands. Seemingly little interaction may result in a successful cooperation.

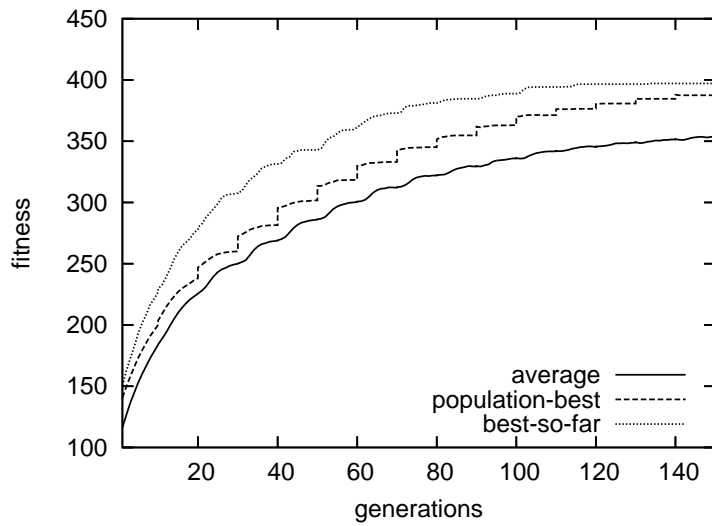
In Fig. 8.14, we see how fitness changes on average in both cases throughout a run. Note that for frequent migration, system-best and island-best fitness merge quickly, which means that from then on all islands resemble each other and follow the same evolutionary path. For sporadic migration there remains a difference between system-best and island-best, which confirms that islands follow different evolutionary paths. Migrations increase average fitness, and there is further growth observable *after* migrations, which must be explained by an increased evolvability. Note that system-best also shows this periodicity, and its increase cannot result directly from migration because it just moves genomes between islands. We can assume that the increased evolvability is a result of successful mixing of locals and migrants.

On charts with selection intensity (Fig. 8.15), we see that for big and frequent migrations, the migration directly improves fitness, and evolution improves fitness only in the very beginning (when local diversities are still high). For small and infrequent migrations, we observe that fitness increases due to the evolution between migrations, and the migration itself has very little effect directly, which matches the compositional evolution model again.

A corresponding plot for global diversity is shown in Fig. 8.16. Again, we see that for the first setup, diversity falls quite fast and then stabilizes, which means that mostly incremental evolution occurs from that point forward. For small migrations we



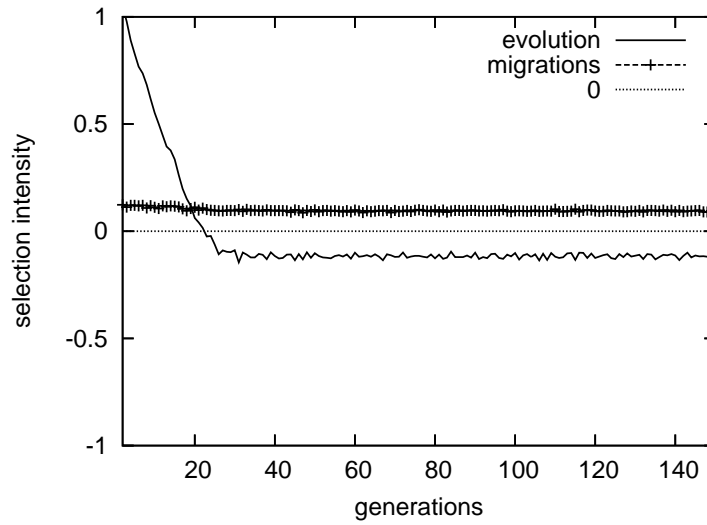
(a) selection=5, size=10, interval=1



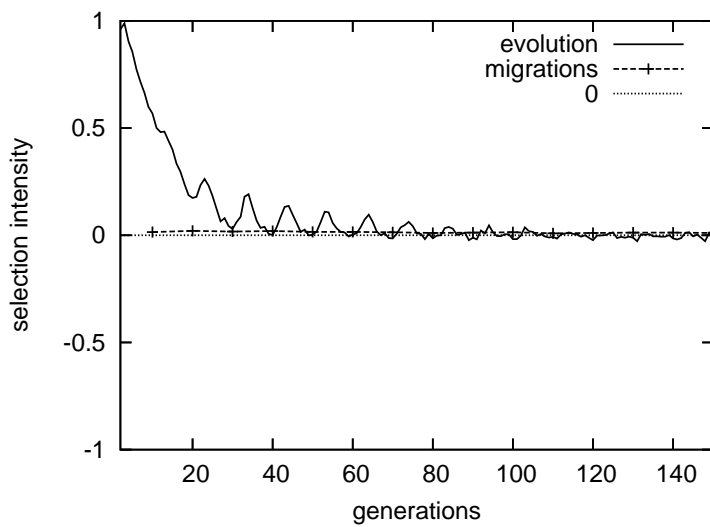
(b) selection=5, size=1, interval=10

Figure 8.14: Fitness in runs for different migration size and interval. Policy=2, topology=2, the H-IFF function.





(a) selection=5, size=10, interval=1



(b) selection=5, size=1, interval=10

Figure 8.15: Selection intensity in runs for different migration size and interval. Policy=2, topology=2, the H-IFF function.

observe a boost in diversity after migration, which suggests compositional evolution.

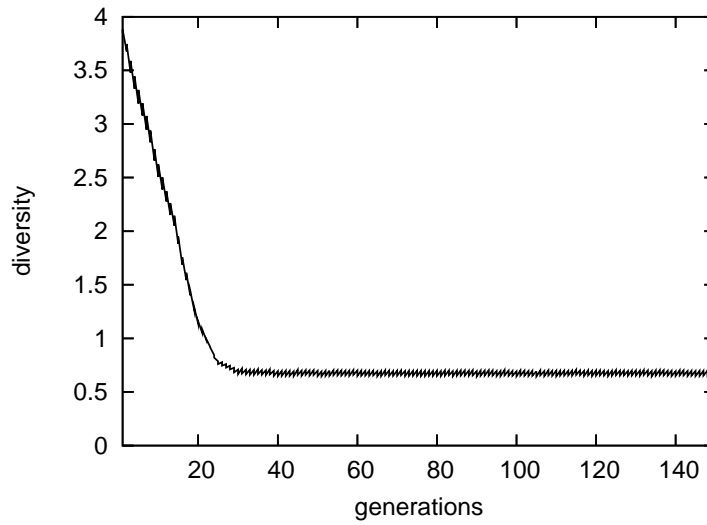
This is confirmed yet again in Fig. 8.17 where I measure how much time the migrants' alleles stay in the population. If we stretch the first chart to match the migration interval of the second chart, we might observe seemingly comparable plots. What is however worth noting, however, is that in the second case, migrations cause a 10 times smaller direct impact. This impact is amplified very quickly to values comparable with the ones for big migrations. This happens due to evolution following migrations, as is the case in compositional evolution.

Summarizing, the results for function H-IFF are similar to the results for the IM6 function. We saw that depending on parameters, either incremental or compositional evolution takes place, and the latter gave us better results. The better results are hierarchically built from lower-level solutions, thanks to an appropriate recombination applied.

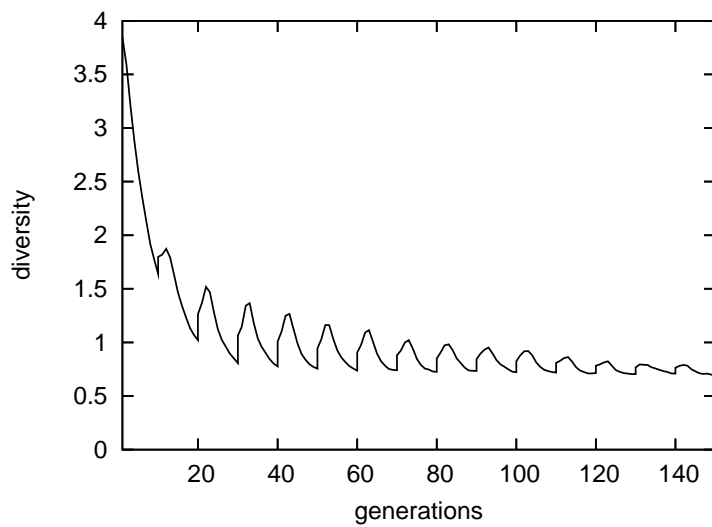
#### **8.2.4 The Deceptive function**

I have performed a similar set of experiments for another function, the Deceptive function, composed of 10, interleaved, 4-bit trap problems (see Appendix C). I used the same setup as for the H-IFF function (including two-point crossover). In Fig. 8.18, we see again that the two roles of migration are possible. However, when we look deeper, we see differences between results for H-IFF and Deceptive functions.

Although the Deceptive function is also modular, its building blocks are scattered through the genome — and the two-point crossover operator is incompatible with this linkage. Therefore, this recombination operator is unable to effectively insert alleles coming from migration.

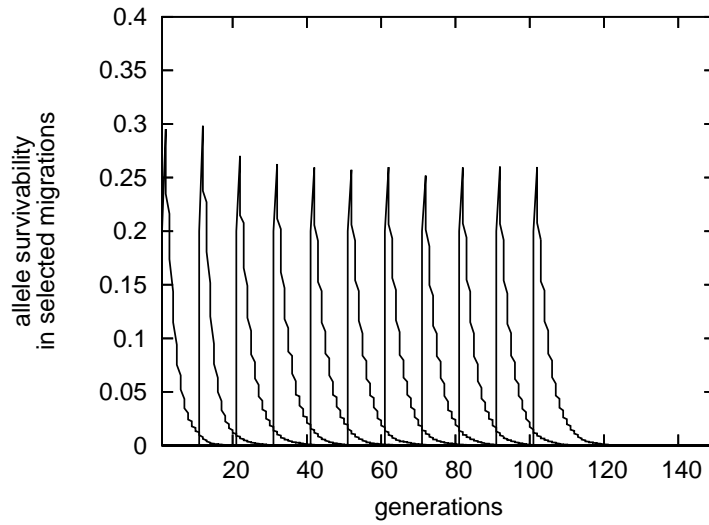


(a) selection=5, size=10, interval=1

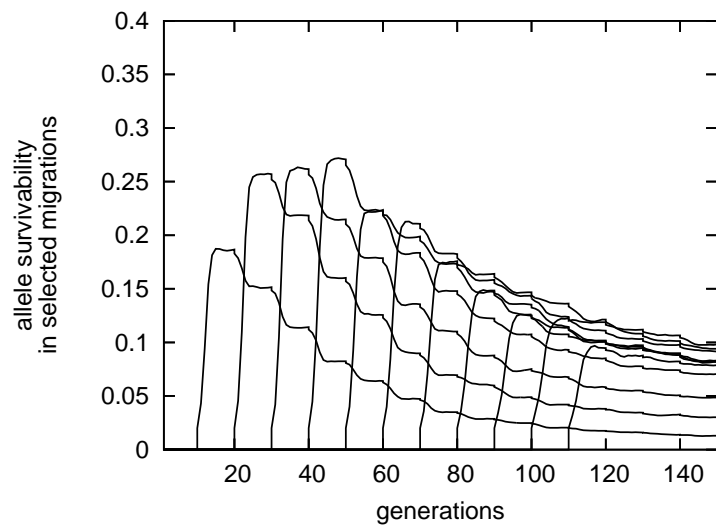


(b) selection=5, size=1, interval=10

Figure 8.16: Global diversity in runs for different migration size and intervals. Policy=2, topology=2, the H-IFF function.

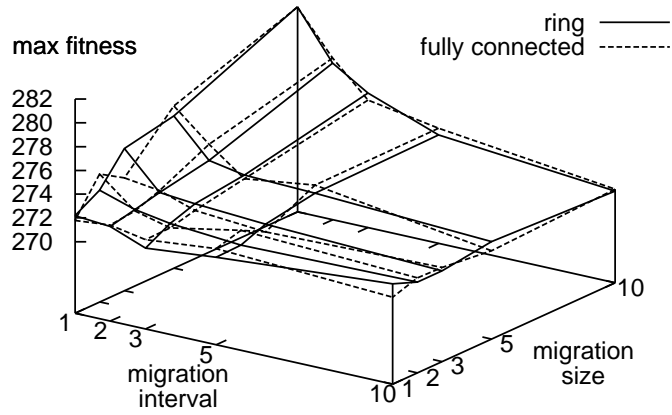


(a) selection=5, size=10, interval=1

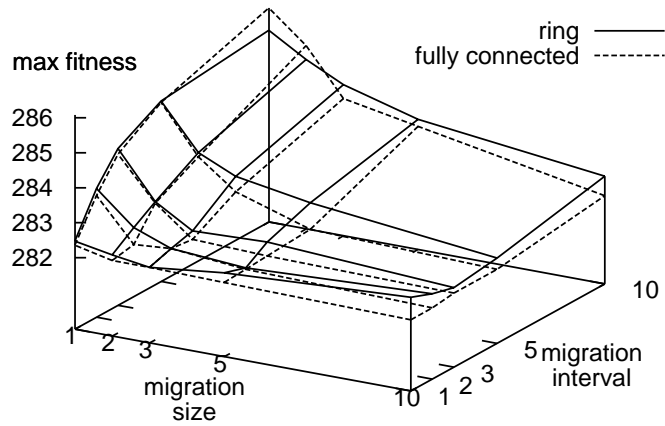


(b) selection=5, size=1, interval=10

Figure 8.17: Alleles survivability in runs for different migration size and interval. Policy=2, topology=2, the H-IFF function.



(a) “Additional selection” role of migration. Uniform selection, 5-tournament migration policy.



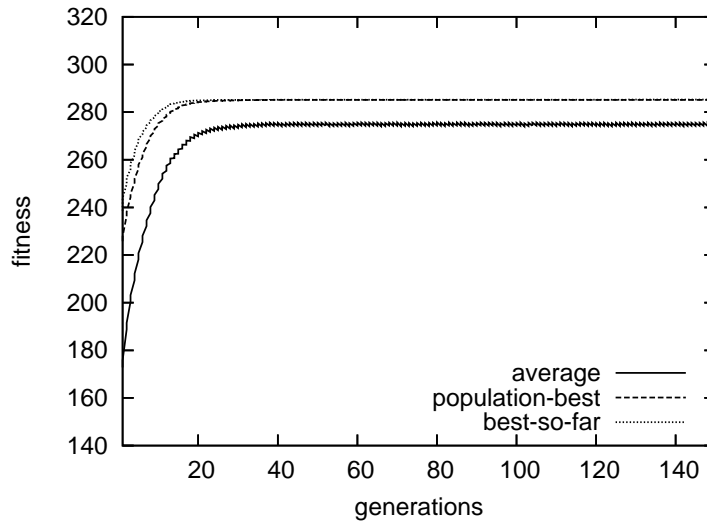
(b) “Genetic material exchange” role of migration. Binary tournament selection, binary tournament migration policy.

Figure 8.18: Two roles of migrations for Deceptive function — additional selection pressure, or exchanging genetic material, the Deceptive function (note a different axis layout).

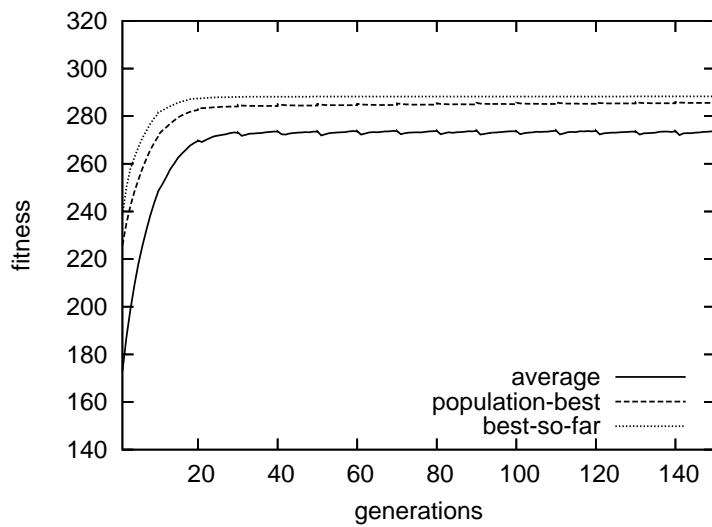
Nevertheless, I observed that strong selection pressure and occasional migration results in slightly better performance (versus frequent migrations). I think that this is simply due to an increased inter-island diversity for a longer time and spreading good solutions, rather than due to composing better solutions from partial solutions, as in compositional evolution. Additionally, a high volume of rejected migrations may have a destructive effect because they may overwrite good solutions in target islands, and, therefore, occasional migrations turn out better.

Similar to H-IFF experiments, I analyze two opposite setups — one with a large and frequent migration and the other with a small and rare migration. In Fig. 8.19, we see that for the big and frequent migration the system-best and island-best curves quickly overlap, as all islands start to share the same contents. For the case with the small and infrequent migrations, we see a separation of the two curves, although the gap is not as big as it was in the case of the H-IFF function. In this case, islands evolve more independently and the resulting diversity is the reason for a difference between the maximal and average island-best value. In the same time, islands quickly stall at some (sub)optimum. We see that fitness does not increase as a result of migration in these experiments. This function does not allow for the potential of inter-island diversity to be used to increase evolvability.

Migrants are quickly rejected for the analyzed case, and this is clearly seen in selection intensity plots (see Fig. 8.20). Frequent migrations seem to have a positive immediate effect, but evolution does not accept migrant genes in target islands. For infrequent migrations, the rejection effect is even more visible, as we observe negative spikes *following* migration. This is of course in agreement with no fitness improvements, which we already noticed.

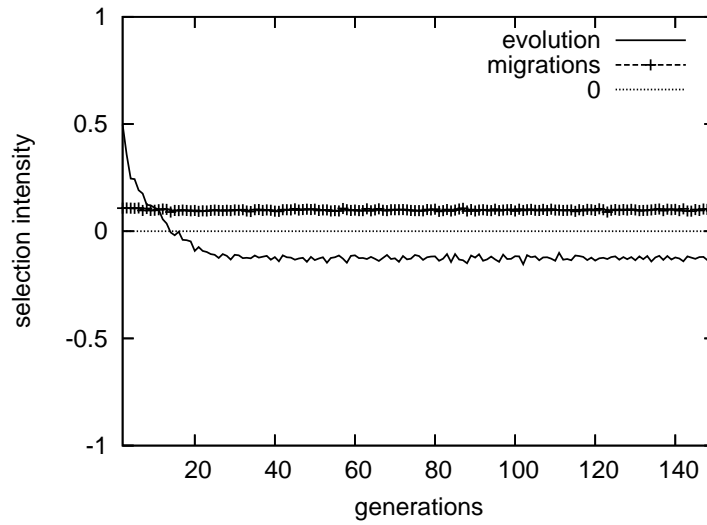


(a) selection=5, size=10, interval=1

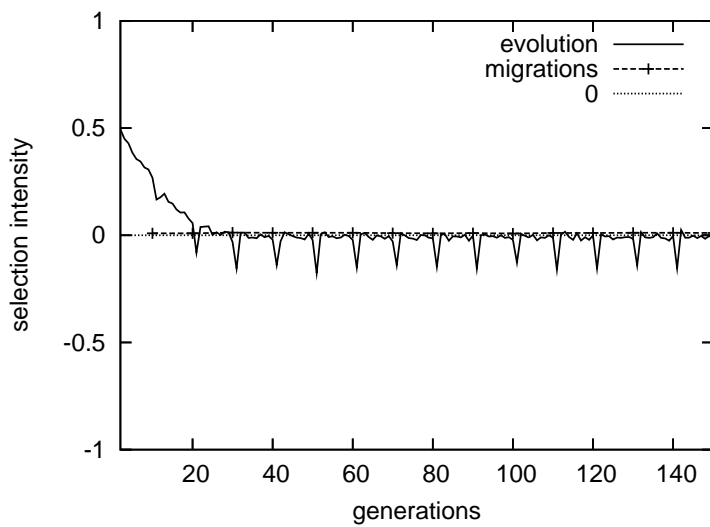


(b) selection=5, size=1, interval=10

Figure 8.19: Fitness in runs for different migration size and interval. Policy=2, topology=2, the Deceptive function.



(a) selection=5, size=10, interval=1



(b) selection=5, size=1, interval=10

Figure 8.20: Selection intensity in a run for different migration size and interval. Policy=2, topology=2, the Deceptive function.



Diversity plots shown in Fig. 8.21 correspond to the general picture by showing that even though migrations inject new alleles, in the following generations diversity quickly drops, without any period of increased diversity and evolvability.

Finally, Fig. 8.22 very distinctively shows that, for the Deceptive function, migrant genes simply do not survive in the target island. For rare migrations we see this trend particularly well compared to the case with the H-IFF function.

Changing recombination to match the problem better would probably enable compositional evolution, as we have seen earlier for the IM6 function. Also turning recombination off might actually help, as it seems that it is not able to effectively mix alleles, and is only creating unfit hybrids. These would be valid future experiments.

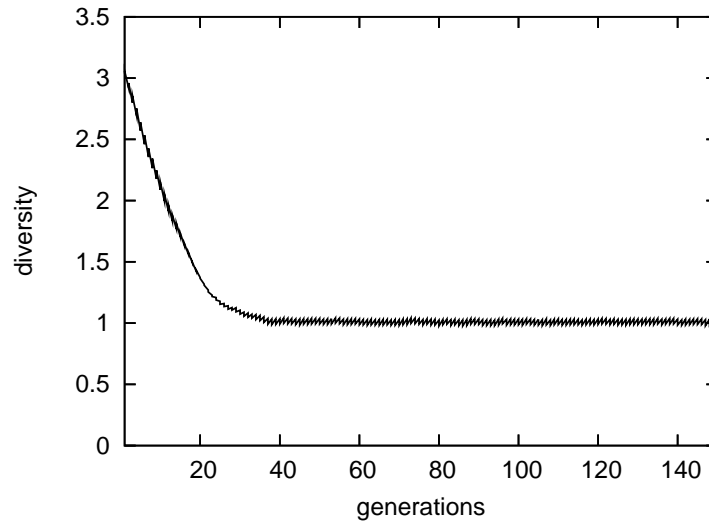
Summarizing, when recombination does not work and compositional evolution does not occur, islands “switch back” into traditional incremental evolution. It is then beneficial to keep the migration level low to support diverse search.

## 8.3 Complex domains

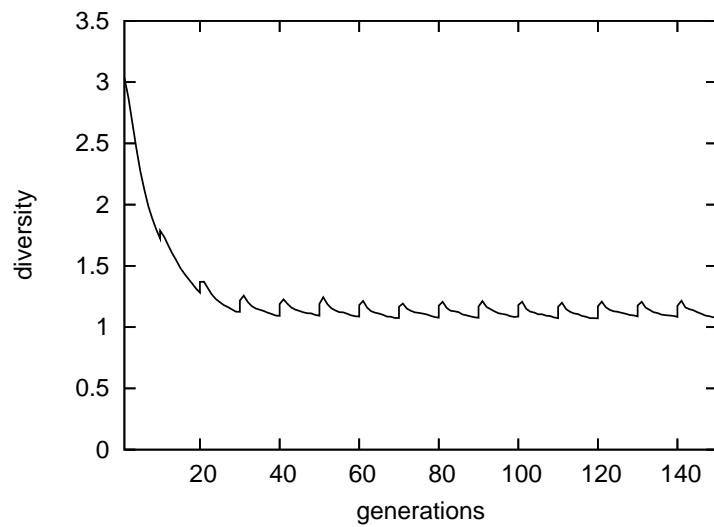
In this last section I present the results for two complex problem domains. The first problem is to optimize a macroscopic behavior of certain rule-based rewriting systems (cellular automata). The second is more engineering oriented, and is to optimize a traffic network in order to minimize the average travel time through a city.

### 8.3.1 Custom recombination operators

Scaling evolution to the inter-island level, and in particular achieving compositional evolution at this level, requires creating proper recombination operators. Uniform crossover does not produce satisfactory results for the two afore mentioned problems,

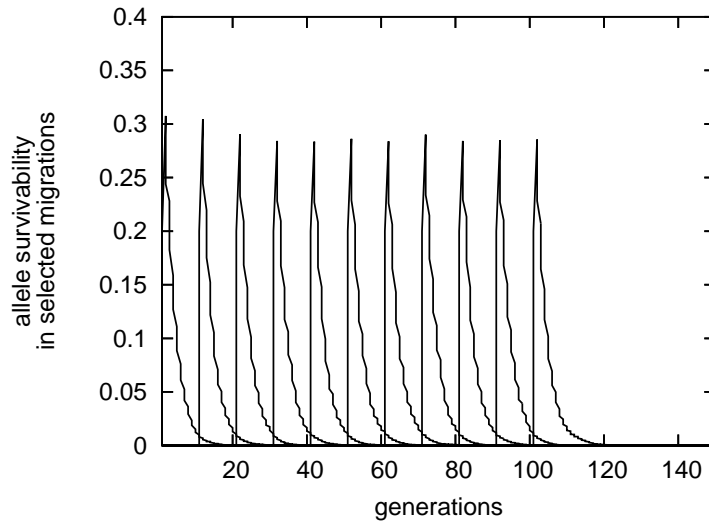


(a) selection=5, size=10, interval=1

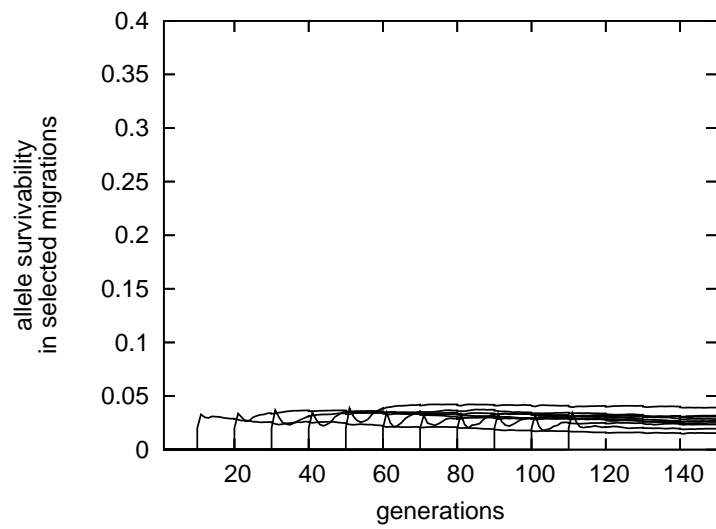


(b) selection=5, size=1, interval=10

Figure 8.21: Global diversity in runs for different migration size and intervals. Policy=2, topology=2, the Deceptive function.



(a) selection=5, size=10, interval=1



(b) selection=5, size=1, interval=10

Figure 8.22: Alleles survivability in runs for different migration size and interval. Policy=2, topology=2, the Deceptive function.

and I tried to design more complex recombination operators. It was not difficult to predict the need for a redesigned recombination operator and in fact both problems were chosen because of how they can be represented, as described below.

### **Neighborhood-based recombination for networks**

Each network can be represented as a graph. Nodes in this graph form a natural neighborhood relation which can be utilized for effective crossover. For both problems this graph has nodes with relatively low degrees. I believe that such networks should have a modular structure (or at least swapping groups of neighboring genes should be both relatively easy and meaningful).

I designed and tested two customized recombination operators. It is important to note that all individuals in a given domain use the same network structure (it does not change from individual to individual) and only the node values change. Recombining different graphs would be a much more difficult task. The two recombination operators are described below.

A simple approach to design a recombination operator could be to pick one link as a seed, and iteratively keep adding some neighboring links, forming a cluster, until half of the total number of links are gathered. Values for the selected links are copied from the first parent, and the remaining values are copied from the other parent. Such an approach would create asymmetry, however, because it would favor the first parent (the unselected links, corresponding to the second parent, could be spread at the borders of the network, and hardly connected with each other). While this approach could better preserve linkage between genes of the first parent, it would probably have

difficulties with transferring good building blocks from the other parent.<sup>1</sup>

I have tried to make the chances of both parents equal, and therefore, I decided to alternate the growth of the two clusters starting with randomly chosen links from the network. The clusters grow until they take all the possible unassigned neighbors. Thanks to alternating, both clusters should have relatively small radiuses. However, because one cluster may surround and block the growth of the other, the procedure may lead to two unequally-sized clusters. Therefore, I extended this method further, to use multiple seeds (clusters) per parent. This had an additional aim of copying long paths of links as a result of merging adjacent clusters for the same parent.

### **Dynamically determined recombination**

A good recombination operator, in order to be constructive, has to preserve a proper epistatic linkage from the parents. Depending on a particular solution, genes may be functionally linked differently than as predicted by the offline static relation between loci. Therefore, for the second recombination operator, I used utilized run-time, dynamic information.

For each parent certain genes may be marked as “important.” Such genes would be copied together from the appropriate parent. This time I resolved conflicts in favor of one of the parents. Alleles for genes that were not marked important for any parent were copied at random. Methods for identifying important genes depend on a particular problem and are described in appropriate sections later.

---

<sup>1</sup>Which is not necessarily a bad idea: Richard Watson’s implementation of SEAM resolves conflicts in favor of just one parent, which apparently works quite well. Also parameterized uniform crossover favors one of the parents.

### 8.3.2 Cellular automata and the majority problem

Cellular automata (CA) are systems consisting of a large number of small, simple elements (cells) which repeatedly interact with neighboring ones to update their state (Wolfram, 2002). Each cell can be represented by a variable that takes a value from a pre-defined set of symbols,  $\Sigma$ . Traditionally cells are put on some regular grid, like a one- or two-dimensional array (or ring and torus correspondingly) and all are updated synchronously. A standard CA is memory-less, which means that the next step depends only on the previous one.

Cellular automata are interesting from the computational point of view. It was shown that they are computationally equivalent to Turing machines provided that we appropriately encode the input problem and read the output.

A one-dimensional CA uses an array of cells. We can denote the cells by  $c_1 \dots c_n$ , where  $n$  is the length of a CA. A state of the whole cellular automata (that is, a concatenation of cells' values) is called *initial condition* (IC).  $IC \in \Sigma^n$ . One refers to these states as ICs, even though they are not necessarily initial — but they might be.

Cells are updated based on rules that read a neighborhood of a cell as input and generate the next step cell value as output. The neighborhood consists of  $k$  cells. Each cell uses the same set of rules to update its state. One of the approaches to encode the set of rules is to explicitly specify the output for each possible neighborhood configuration. There are  $|\Sigma|^k$  configurations, which have a natural order obtained by treating them as numbers in  $|\Sigma|$ -symbols alphabet. Therefore, to encode all the rules, just the outputs for all the rules can be concatenated, thereby, creating a string of length  $|\Sigma|^k$ . Since each rule can output any of  $|\Sigma|$  possible states, there is  $|\Sigma|^{(|\Sigma|^k)}$  of

such encodings representing all possible CAs (of the certain type that I analyze).

The simplest cellular automata have just two possible states for each cell ( $\Sigma = \{0, 1\}$ ) and the neighborhood consists just of the adjacent cells. The neighborhood may consist also of multiple cells on each side plus the cell being updated, e.g. for  $c_i$  it would be  $c_{i-r} \dots c_{i-1} c_i c_{i+1} \dots c_{i+r}$ . The number of cells on each side in the neighborhood,  $r$ , is called its radius and the neighborhood size is  $k = 2r + 1$ . There are other, non-continuous neighborhoods possible. ICs are often “wrapped,” which means that for cells close to the border of an IC, their neighborhoods extend to the other side of the IC. Alternatively, one could specify a constant value for all neighborhood cells extending beyond the border of an IC.

Since neighborhoods of adjacent cells overlap, the states of the cells in the next IC are correlated. For a given configuration of a neighborhood (and a rule corresponding to this configuration), there are two possible neighborhoods (rules) that may overlap to the left (created by concatenating either 0 or 1 to the left, and dropping the last symbol on the right), and similarly two neighborhoods that may overlap to the right. If we order rules as binary numbers, the “left neighbors” for rule  $i$  will have numbers  $\lfloor i/2 \rfloor$  and  $\lfloor i/2 \rfloor + 2^{k-1}$ , and the “right neighbors”  $(2i) \bmod 2^k$  and  $(2i + 1) \bmod 2^k$ . A few rules will have less than four neighbors, and it is easy to show that those are  $0^k$ ,  $1^k$ ,  $(01)^{k/2}0$  and  $(10)^{k/2}0$ , which have 3 neighbors. We can represent these relationships on a graph (see Fig. 8.23, with rule left-hand sides being the nodes, and edges connecting neighboring rules. A more formal theory of the above observations is used for counting *preimages*, that is possible preceding ICs of a given one, and drawing *De Bruijn diagrams* (Wikibooks, 2007).

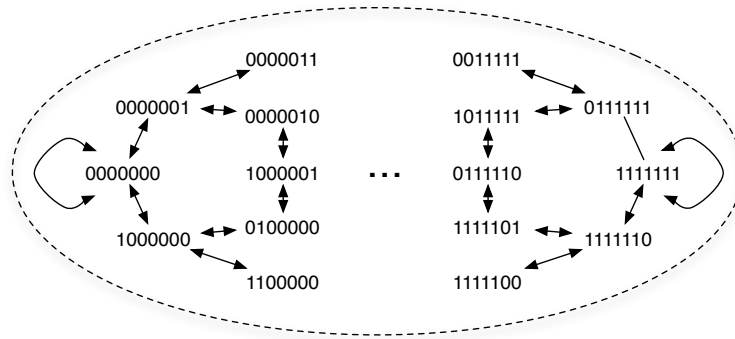


Figure 8.23: Part of a graph of rules neighboring each other (let-hand sides for only selected rules shown, close to  $0^k$  and  $1^k$ ).

### Problem description

Elements (cells) of a CA are usually very simple (have simple update rules). Therefore, the overall behavior of a system depends on the interaction between cells, which is of a local nature. Even though such an interaction may lead to an emergent behavior of a much higher complexity (Wolfram, 2002), this behavior is extremely difficult to predict and program because of its implicit nature. This characteristics makes this problem suitable for evolutionary computation.

An example of a global problem can be the *density classification task*, also called the *majority problem* (Pagie and Mitchell, 2002). The problem uses two symbols, 0 and 1. An automaton should converge to all ones, if there were more ones in the initial condition array, and to all zeros otherwise. The majority problem is hard to solve. The simplest, greedy algorithms fail to give good results and it turns out that the task requires passing the information across the array. It was also shown theoretically that there does not exist a perfect CA properly classifying all possible initial conditions (Land and Belew, 1995).



The fitness of a CA is the percentage of ICs which are correctly classified. It is, however, impossible to test a given CA on all possible ICs because of their large number. Therefore, one usually tests a CA on a subset of ICs that is large enough to give a good estimation of a CAs fitness. There are a few methods possible to generate ICs for this type of subset. One method involves first choosing the number of “1”s in an IC, and then randomly placing them in the IC and setting the remaining cells to zero. This method, however, results in a uniform distribution with regard to the percentage of ones in ICs, whereas it is known that for all possible ICs this distribution is Gaussian. Moreover, ICs with a large disproportion of “1”s versus “0”s are easier for CAs to classify correctly. Therefore, another method is to simply set each cell randomly to either 0 or 1, which results in a Gaussian distribution of “1”s’ percentages, although only ICs with percentage of “1”s close to 0.5 will practically be generated.

Experiments with evolving solutions for the majority problem show that we can identify three classes of relatively good CAs, and evolution usually finds them in order (Pagie and Mitchell, 2002). The first is a class of CAs that either always converge to all zeros, or always converge to all ones. Obviously, because there are 50% of ICs that should converge to ones and 50% that should converge to zeros, such a CA will have a fitness of 0.5. The second type of CAs are so-called “block-expanding” CAs. Their behavior is different for ICs converging to zeros and different for ICs converging to ones. These CAs attempt to quickly (within a few generations) converge to one of the symbols (say zeros), and succeed if there are many zeros in the starting IC. If there are continuous blocks with many ones, however, these type of blocks consolidate and start to expand, merge, and ultimately cause the whole CA to converge to ones.

Such CAs have fitness values usually around 0.60–0.65. The third type of CAs are so-called “particle-based” CAs. They consolidate blocks for both zeros and ones, and propagate both of them from iteration to iteration by utilizing coordinated rules. In fact certain patterns may also be formed, rather than a solid block, consisting of various combinations of 0s and 1s. The patterns may expand or shrink in width, and they can also shift to the left or to the right. Depending on a particular solution, when two patterns meet they form another pattern that may propagate in a different way (resembling two particles colliding and producing a third particle, hence the name). The aim of patterns is to pass a signal across the CA. The final outcome of interaction of those signals is the proper convergence of the CA. Most particle-based CAs simply use solid blocks of zeros and ones in the beginning and then form a single pattern that moves in one direction to denote the majority of zeros and in the other direction for ones. Particle-based CAs have fitness values usually above 0.7.

The majority problem was used as an example in multiple studies and thus the best solution was improved multiple times over the years. Table 8.1 shows a sample of the best CAs discovered until today; the last one by Juillé and Pollack being the current best known solution (Verel *et al.*, 2006). Recently, CAs with fitnesses above 0.8 have been routinely found by various researchers, but none have broken the record.

## **Implementation**

I implemented a CA to solve the majority problem in the standard way. The CA model uses two symbols,  $\Sigma = \{0, 1\}$ . The neighborhood is of size 7 (the radius of 3), so there are  $2^7 = 128$  possible configurations. This means that CAs are encoded by strings of length 128,

Table 8.1: Selected best CAs found for the majority classification problem.

| <i>author(s), method, year</i>                   | <i>hex representation</i>        | <i>fitness</i> |
|--|----------------------------------|----------------|
| Gacs, Kurdymov and Levin (GKL),<br>by hand, 1978 | 005F005F005F005F005FFF5F005FFF5F | 0.815          |
| Das,<br>by hand, 1996                            | 009F038F001FBF1F002FFB5F001FFF1F | 0.823          |
| Andre, Bennet and Koza (ABK),<br>GP, 1996        | 050055050500550555FF55FF55FF55FF | 0.824          |
| Juillé and Pollack,<br>coevolution, 1997         | 1451305C0050CE5F1711FF5F0F53CF5F | 0.863          |

e.g. 0001010001010001001100000101110000000000101000011001110010111110001-0111000100011111111010111100001111010100111100111101011111, which means that it transforms  $0000000 \rightarrow 0$ ,  $0000001 \rightarrow 0$ ,  $0000010 \rightarrow 0$ ,  $0000011 \rightarrow 1$ , etc. I operated on ICs of length 149, which were wrapped for neighborhoods. The number of all possible CAs is  $2^{(2^7)} = 2^{128} \approx 3.4 \cdot 10^{38}$ .

One genome corresponded to one CA. Two rules must always be present in any CA converging to either ones or zeros, and need not be encoded. They are  $0000000 \rightarrow 0$  and  $1111111 \rightarrow 1$ . Therefore, the genome length was 126. I used a flip-bit mutation. I operated on strings of length 149. Each CA was allowed to iterate through a maximum of 320 steps in evaluation, although a convergence could be detected earlier, which sped up computations.

The two customized recombination operators were implemented. The static recombination operator used a graph of rules mentioned earlier in this section for finding genes related to each other. The maximum degree of a node is defined by the possible number of neighboring rules bounded by four. The dynamical recombination

operator identified important genes measuring the frequency with which rules were executed. The frequencies were averaged throughout runs starting from multiple ICs, and weighted by the number of iterations for a given IC. If a rule was executed more often than the average for all the rules, it was considered important, which meant selecting it in recombination. Because certain rules were usually executed many times more than others, the distribution was skewed (the average was much higher than the median) and this procedure resulted in a relatively small subset of rules chosen.

## Experiments

I have performed a set of varied experiments to measure IM performance on evolving CAs for the majority problem. Due to a long computation time, the set of configurations is not exhaustive and configurations were chosen one after another manually when trying to understand specifics of this domain. Nevertheless, the set represents a spectrum of settings.

In all of these experiments I used a (dynamic) full topology, only parent selection (no survival selection) and non-overlapping generations. Mutation was either set to 0.008, or 0.01, both close to  $1/L$ .

Fig. 8.24 shows the best-so-far fitness curves. Parameter settings are given as descriptions on the chart. The best results were achieved by

- 20x50 setup — only IMs setups (i.e. not panmictic) with 1000 individuals (20x50) achieved results significantly better than 0.5.
- very small migration size ( $\alpha = 0.02$ ) — increasing the migration size resulted in considerably worse results.

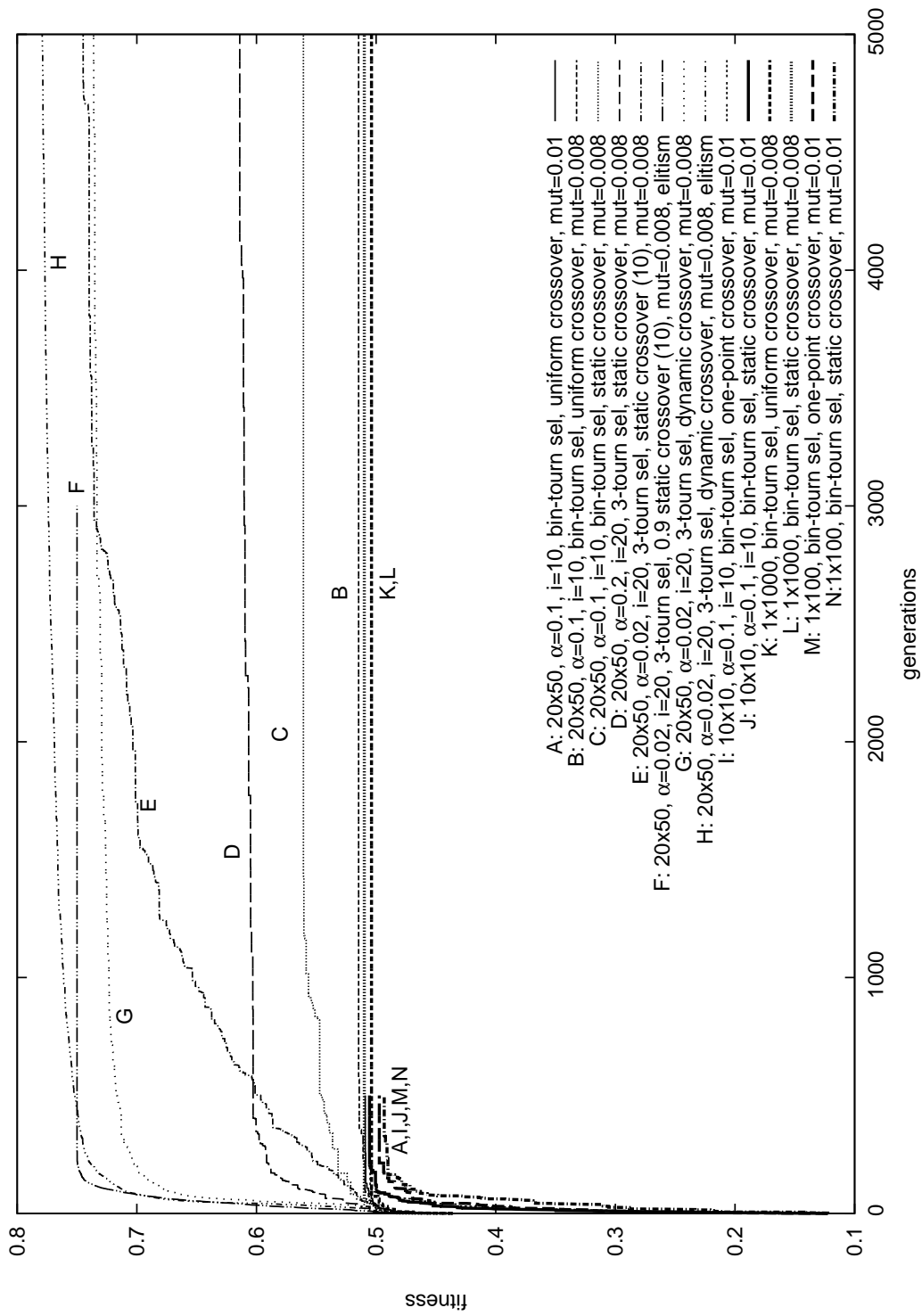


Figure 8.24: Fitness curves for CA domain results.

- relatively strong parent selection pressure (3-tournament) — elitism increased the performance.
- dynamic recombination operator was better than static recombination.

These observations suggest that keeping islands separate (with very small migration size) is a method for increasing the performance. Such a small migration size allows for safely using a strong selection pressure with elitism. A choice of dynamic recombination, which results in a faster convergence to active alleles matches a preference for stronger selection pressure. A high level of allele exchange between islands would lead to a premature convergence and loss of diversity.

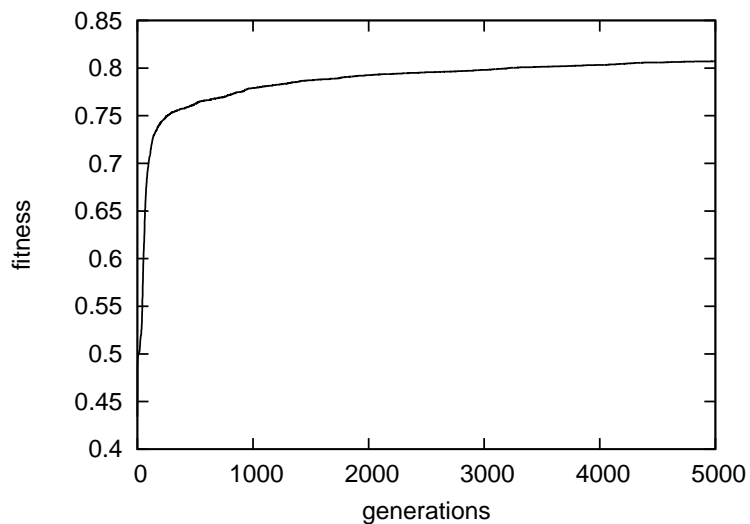
Since setups with more (smaller) islands and very low migration size obtain the best results, I ran an experiment using a 50x20 setting. The results are shown in Fig. 8.25. Looking at contributions of different type of individuals,<sup>2</sup> we see that it is in fact hybrid individuals that overall create the biggest increase in fitness. This means that keeping migration size very low may be appropriate to exchange genetic material, and not dominate the target islands. Recombination must be able to enhance the effect of small migrations, and the evolution must at least partially proceed in a compositional way.

Even though I used special recombination operators, I was not able to produce solutions beating the set of already known good examples. Therefore, I believe that it is extremely difficult to create an effective recombination operator for this problem.

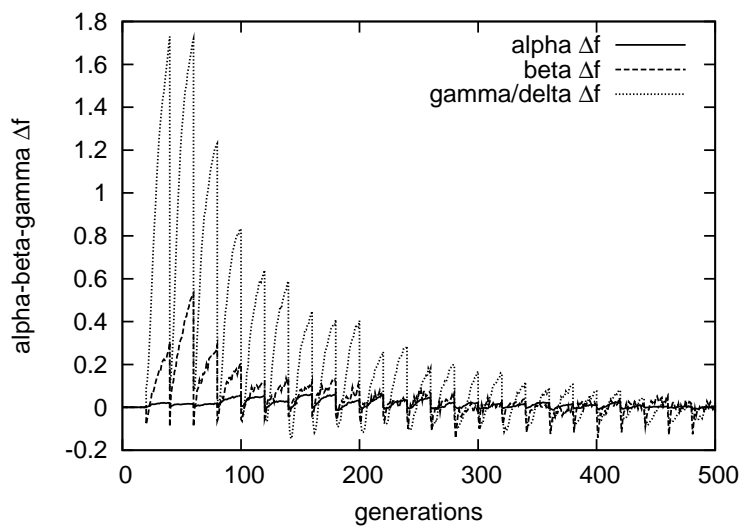
Quite possibly the nature of the problem is not as modular as I hoped, and most of

---

<sup>2</sup>See section 8.2.2. Individual type contribution is computed as an average fitness increase in periods between migrations for which a given individual type dominated.



(a) fitness



(b) fitness contribution from migrants (alpha), locals (beta) and hybrids (gamma)

Figure 8.25: Fitness changes in a configuration suggested by previous experiments for the CA domain (50x20 IM,  $\alpha=0.02$ ,  $i=20$ , 2-neighbor topology, 3-tournament selection, dynamic crossover, mutation=0.008, elitism).

the genes are strongly related to each other. Many rules are cooperating if they are executed together, even if they are not in a direct neighborhood.

However, I still managed to produce a high percentage of so-called particle CAs, which were better than the ones created by hand. This high efficacy is promising. Further changes to the settings of IMs or enhancing the model with some domain specific improvements, seems a plausible option to improve the performance. A deeper study of allele flow between islands could shed more light on the ability of obtaining better results.

### **8.3.3 Traffic congestion and optimizing street lanes**

The second problem I studied in the context of IMs, was related to a flow of traffic in a city. Such problems are significant in engineering because a proper design not only simply minimizes vehicle travel time, but has many other implications, including economical, ecological and safety concerns.

A standard way of modeling traffic in a city is the following. A city network is modeled by a graph, with nodes denoting crossings, and edges denoting roads. A number of so-called zones are identified. Zones either represent neighborhoods (internal zones), or they are placed where roads end on the border of the simulated area (external zones). All traffic inside the simulated area may be described in terms of zone-to-zone demands (original-destination, or O-D, trip matrix), that is for each pair of zones we can estimate the number of cars traveling per time unit. In real-world applications those demands would be determined by a field study. The zones are connected to the network (usually in just one place, but possibly in more than one place) through virtual “zone connector” roads.



As more drivers choose the same path (link), congestion causes the time of travel to increase. This can be described by the following function, recommended by The Bureau of Public Road (BPR)

$$t = t_f \left( 1 + \alpha \left( \frac{v}{c} \right)^\beta \right),$$

where

- $t$  = Congested link travel time
- $t_f$  = Free Flow link travel time  
(Length of the link divided by a fixed Free Flow Speed)
- $v$  = Link Volume
- $c$  = Link Capacity
- $\alpha, \beta$  = Calibration parameters (usually taken as 4.0 and 0.15 respectively)

An *all-or-nothing* procedure would be to assign all the traffic to the fastest paths (e.g. using a standard Dijkstra algorithm) between origin and destination and then compute the travel times. A better heuristics, recommended by the Federal Highway Administration (FHWA) is to compute traffic congestion iteratively (and this is the approach I used). The procedure is the following :

1. Load all traffic using all-or-nothing procedure and compute travel times
2. For some number of iterations:
  - a. Compute new traffic assignment based on current travel times
  - b. Update traffic load using 0.25 of the latest iteration  
and 0.75 of the previous assignment
  - c. Compute new travel times

More complex procedures can be used for this computation, in particular calculations that compute the equilibrium to which traffic converges. However, for my purposes more sophisticated models were unnecessary.

## **Problem description**

Imagine a network of streets in a city with each street having several lanes. I have assumed that it is relatively easy to switch permanently how many lanes are directed each way, and I sought a configuration that minimizes the average travel time between various points (given a fixed demand matrix). This problem was created only as an illustration for a class of problems, but I put effort into making it plausible engineering-wise.

Changing the direction of some lanes would obviously impact the capacity of the street in both directions and should be adjusted together with other streets at least in the neighborhood. We see that the problem has a lot of local interaction. This situation creates a good problem for evolutionary computation since the number of possible combinations grow exponentially with the number of streets, and it is not obvious how the choices are related to the final outcome. To compute the fitness of a given solution, route assignment and travel times must be computed.

## **Implementation**

In my research I used routes inside Washington DC only, from the National Highway Planning Network database. It consisted of 264 two-directional links (about 210 miles total length), 186 nodes and 90 zones (both external and internal). Each route represented in the network had at least two lanes (or a few did not, so I modified them to simplify the data), and most had four. Trip matrices were synthetic, generated using normal distribution and were fixed throughout experiments. The free flow speeds were set as equal to speed limits on given links. The network with zones is shown in Fig. 8.26.

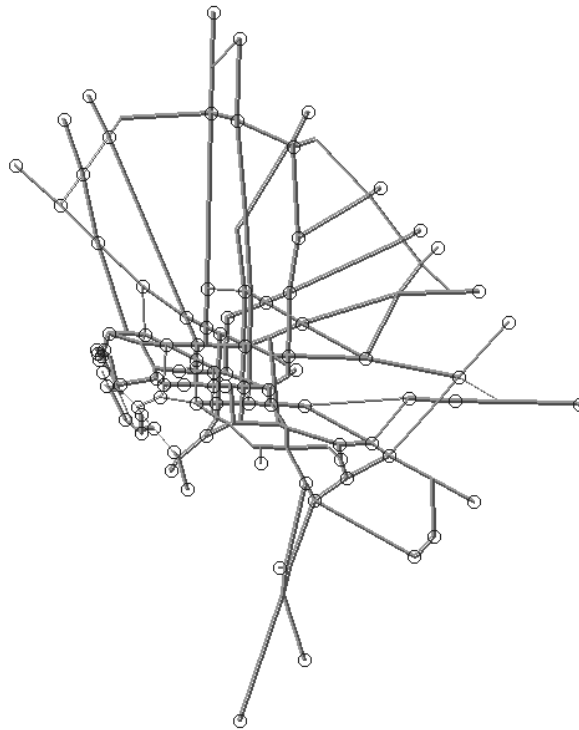


Figure 8.26: Washington DC traffic network used in experiments. Circles represent zones, and lines represent streets.

As already mentioned, the focus was not on solving a particular problem, but rather on comparing the behavior of algorithms utilizing a representative from a class of problems. Therefore, I was less concerned about the simplifications I have made, or the feasibility of enforcing the solution in reality. I believe these assumptions do not change the nature and complexity of the problem.

For the route assignment method described earlier I used the recommended  $\alpha = 4.0$  and  $\beta = 0.15$  constants.<sup>3</sup> The number of iterations when solving each solution was four.

Each link was represented by one gene, with an additional 90 genes representing zone connectors. This gave genomes a length of 354. Each gene could take an integer value representing the number of lanes in one direction. Knowing the total number of lanes, we also know how many lanes go in the other direction. I have additionally assumed that at least one lane must go in each direction, to avoid problems with unreachable nodes in the network. This decision obviously makes genes representing two-lane streets have only one acceptable allele (i.e. of value one). There were 46 such lanes, so the number of actual modifiable genes in a genome was 308, and the remaining genes created only some bias for the EA, which has been left unchanged for implementation simplicity. All the 308 lanes had four lanes, and thus the allowed alleles were 1, 2, or 3.

Similar to the CA domain, also for the traffic domain I implemented the two graph-based, customized recombination operators. For the static recombination operator, it was natural to copy together, from parents to children, genes representing streets that share a common intersection. Therefore, I represented a traffic network by a graph

---

<sup>3</sup>Do not confuse those numbers with  $\alpha$  and  $\beta$  used to denote respectively migrants and locals percentage in the target island after migration.

where streets are represented by nodes and there are links between streets sharing a common intersection. The degree of a node in such a graph is equal to the number of streets having a common intersection with a given one. This graph was then used to determine neighbors in the recombination operator. The dynamical recombination operator identified important genes by selecting genes with more traffic than average, which was done using the traffic assignment procedure for each parent. I believed that if these links get much traffic, there must be set up correctly, and, therefore, were copied to offspring.

## Experiments

I have performed a series of experiments, both with single-population models and with IMs. The traffic domain is computationally expensive, and, similar to the CA domain, I did not run experiments for a full combination of arguments, but rather chose a few selected configurations based on an incrementally deep understanding of the domain.

A limit of 100 individuals in the system was assumed (due to expensive computations). The default setup for the experiments was a random migration policy (group), a (dynamic) full topology (one exception),  $L=354$ ,  $0.003 \approx 1/L$  mutation rate, only parent selection (no survival selection) and non-overlapping generations. I used different migration size, interval, topology (in one case), selection, recombination, 1x100, 5x20, 10x10 or 20x5 IM setup, 1000 or 3000 generations, and elitism or no elitism. The fitness curves for all the experiments are shown in Fig. 8.27. Note that optimizing traffic flow is a minimization problem.

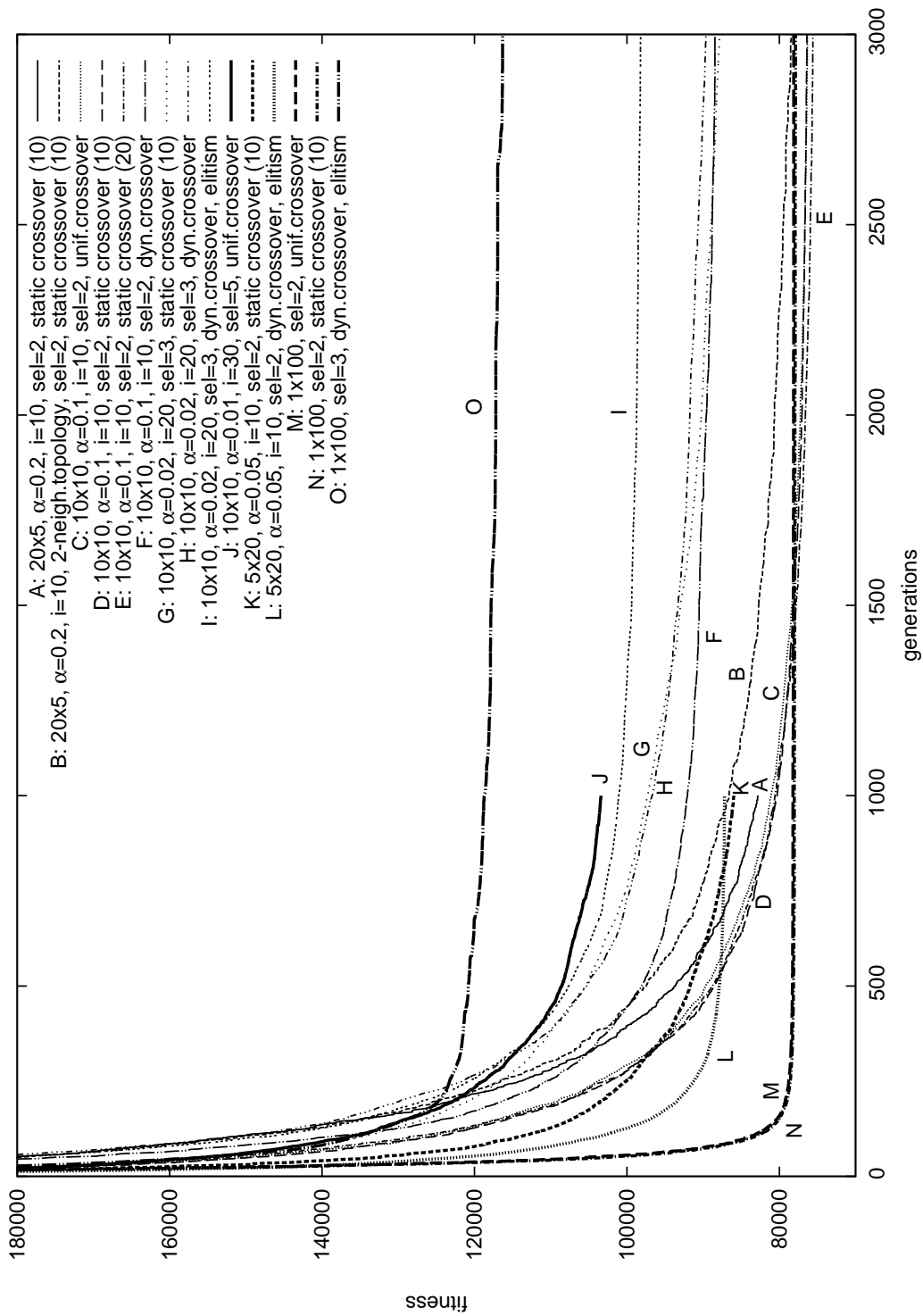


Figure 8.27: Fitness curves for traffic domain results.

Although the single-population model produced very good results, in longer runs IMs turned out slightly better. Both methods produced better results than the original assignment of lanes (when each four-lane street was divided in two lanes each direction). Unfortunately, although promising in general, due to the random values in the trip matrix, this comparison is not directly applicable to the DC street network.

The best setups differ from the CA experiments, as follows:

- 10x10 setup — 20x5 setups seemed to produce better results than 10x10, but suffered probably from a too high migration interval (which was fixed in the final experiment, see below).
- medium migration size ( $\alpha = 0.1$ ) — very small migration size ( $\alpha = 0.02$ ) resulted in suboptimal solutions.
- high exploration (binary tournament, static or uniform recombination) — stronger selection and elitism, as well as dynamic recombination resulted in worse solutions.

It is worth noting that panmictic setups with relatively weak (binary tournament) parent selection and static or uniform crossover produced good results.

I ran additional experiments, enhancing parameter values that created the best results. I used a higher number of islands (e.g. 20x5 setups), together with the best observed migration size, which was 0.1. I set the selection pressure low, and did not use elitism. Results are shown in Fig. 8.28. This time we see that hybrids of migrants and locals are quickly rejected (peaks *increasing* the fitness, which is bad in this case!), and a domination of either migrants or locals must lead to the good results.

This behavior would rather correspond to an incremental evolution and competition between islands.

While the IMs outperform a standard EA, they were only slightly better and only in really long evolutionary runs. It might be the case that this domain is even less modular. Possibly, roads must form optimal paths extending through a whole network, thus making the analysis of only local interactions not sufficient to explain the global behavior. However, even in the absence of effective recombination, the separation of islands turned out to be beneficial and in agreement with observations derived from the Deceptive function. We saw that when IMs were unable to effectively mix migrants into target populations, configurations with more exploration, and more interaction between islands turned out better. This may be surprising for a setup in which there is actually less cooperation between islands.

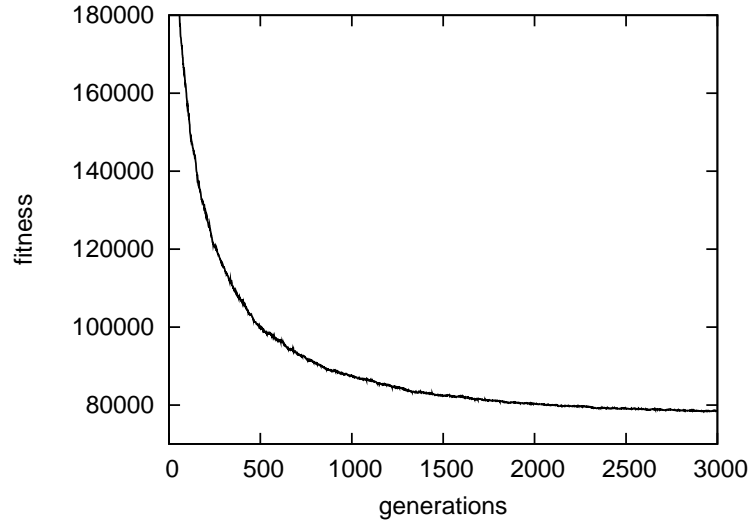
While the traffic domain is interesting in itself from engineering point of view, in this dissertation I used it purely as an example of an interesting computational problem for IMs. Nevertheless, we can clearly see that evolutionary computation is a methodology that could be successfully used for optimizing similar engineering problems, or at least as a support tool in planning. This approach could easily beat schematic human-designed solutions.

## 8.4 Summary

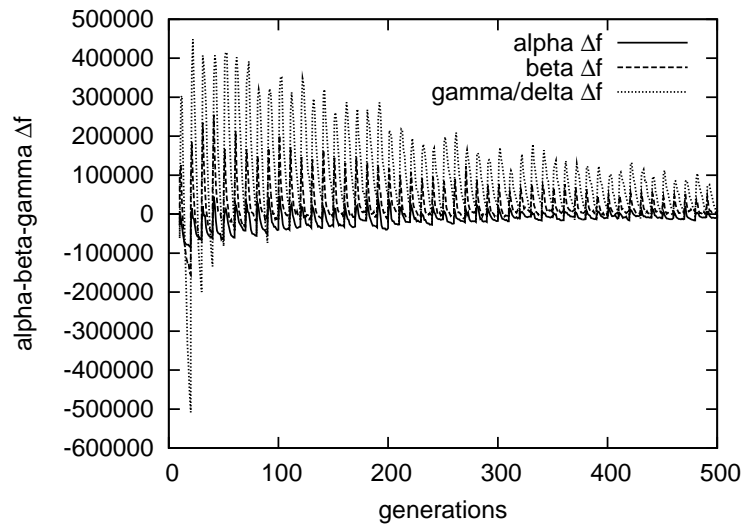
Thanks to the understanding of IM behavior gained in previous chapters, in this chapter I was able to explain IM behavior on a variety of functions. Several main points can be identified.

I confirmed that





(a) fitness



(b) alpha, beta and gamma contribution

Figure 8.28: Fitness changes in a configuration suggested by previous experiments for the traffic domain (20x5 IM,  $\alpha=0.1$ , interval=10, selection=2, static crossover(20)).

- Large number of small islands produced best results.
- Migrations may serve to exchange genetic material, which is later combined into the target island genetic pool, and successful hybrids are formed. Fitness improvement was maximal for a high selection pressure and very low migration level (see, e.g., see the CA domain).
- Migration may serve for spreading good solutions at the cost of decreasing global diversity. This happens when using badly designed recombination operators, because effective recombination of migrants is unlikely. We have observed this behavior for moderate selection pressure and average migration level (e.g., see the traffic domain).
- For very weak selection, migration level should be high and serve as an additional selection pressure (but the results will be worse than with a stronger selection).

In this case study, the recombination operator is very important. With the same migration settings, the choice of recombination operator may result either in accepting or rejecting migrations, which is crucial for inter-island evolution. If no constructive recombination operator is designed, keeping islands separated for a long time (to keep searching various search space regions) is probably a good approach. However, a considerable amount of migrants must be exchanged (moderate migration size), because unsuccessful recombination does not enhance the effect of migration.

It is difficult to relate the results in this chapter directly to the exact predictions of the theory developed in earlier chapters. This comes as no surprise, because the models used earlier were simplifications. Nevertheless, the repeating theme is a

distinction between an incremental evolution and compositional evolution at the inter-island level. Each of these modes requires different setups of all the other parameters.

## Chapter 9: Conclusions

In this dissertation I have studied island models (IMs) of evolutionary algorithms (EAs). Contrary to a standard EA with a single population, an IM has multiple sub-populations (called islands), which interact with each other by means of migrations. This interaction dramatically changes the dynamics of each island when compared to a case study of a set of isolated populations. Yet, due to a separate evolutionary algorithm in each island, the whole system is still very different from a single-population model. Therefore, IMs are a distinct category of algorithms.

There are several reasons why IMs are worth studying. As shown by numerous researchers, on certain problems with a non-trivial fitness landscape, IMs can reach better results than standard, single-population EAs. IMs are also good candidates to run on multiple machines because the distributed implementation is easy and requires little communication between machines, which in turn is important for computationally expensive problems.

Unfortunately, there is no one comprehensive theory describing IMs. There exist many “rules of thumb,” but obviously they are not appropriate in every situation. An understanding of internal mechanisms of IMs helps to create more appropriate guidelines that should ultimately improve performance.

## 9.1 Contributions

This dissertation made a few important steps toward understanding IMs. This deeper understanding changes the way of thinking about IMs. I showed that we may conceptually separate two levels of evolution in IMs, a local intra-island evolution, and a global inter-island evolution. Neither of those is similar to evolution in a standard EA, because they interact and profoundly influence each other.

Evolution at the global level may proceed either in competitive mode or cooperative mode. In the competitive mode, islands evolve solutions mostly on their own, usually by incremental evolution gradually improving solutions, and migration serves to spread good solutions. This mode corresponds to traditional understanding of IMs. In the cooperative mode, islands exchange parts of solutions by recombining migrants and local individuals, which results in a compositional evolution at the inter-island level. In both cases, sporadic migrations between islands turn out beneficial, because they ensure independent evolution and inter-island diversity.

Local evolution dynamics are greatly influenced by migrations. By injecting migrants, migrations create unstable island configurations, which afterwards, by evolution, converge to either migrants, locals or some type of hybrids, until the next migration occurs. Local evolution may be perceived as a series of such epochs between migrations. The outcome of local evolution determines the type of interaction between islands, and defines how the global, inter-island evolution proceeds.

The two-level evolution perspective produces different guidelines for setting up IMs. I argued that to effectively use the different characteristics of IMs, more resources should be given to the inter-island level. Instead of using a few big islands exchanging

best solutions, I proposed a configuration with many small islands exchanging random individuals. I showed that such configuration performs better for problems that may be solved by compositional evolution, in particular for modular, hierarchical problems.

The results in this dissertation were achieved by a range of methods, from linear algebraic models to fully-blown exhaustive experiments with real-life problems.

### **9.1.1 Interaction between the local and global evolution**

Below I list the conclusions regarding the two-level evolution perspective in more detail:

- The two-level IM dynamics is important.
  - A balance between the two levels should be maintained.
  - For global evolution, there must be enough islands to maintain the evolutionary process. The importance of the global evolution should not be underestimated and setups with the number of islands bigger than island sizes are recommended.
  - A two-level dynamic is beneficial for compositional evolution because local evolution may create good building blocks, and global evolution may mix them.
  - Effective recombination is crucial for scaling compositional evolution to the inter-island level.
  - To take advantage of both levels, inter-island diversity should be maintained by setting migration size low, interval long, topology sparse and and policy weak.

- Global evolution impacts local evolution.
  - Local and global evolution interact with each other with an initial phase of “accumulation” of global diversity, and a secondary phase in which diversity may be “released” to help maintain evolvability locally.
  - A slower pace for global evolution allows for using the local evolution to test and adjust various solutions.
  - Migration triggers new EA dynamics and increases evolvability (rather than just exchanging pre-evolved solutions).
  - While migrations may serve as an additional selection pressure, they usually serve much better as a sporadic exchange of genetic material.
  - Recombination may increase the local effects of migrants injection, allowing for longer active evolution.
  - Migrant acceptance may be low, but repeated migrations may help to enforce it.
  
- Local dynamics create global evolution.
  - The periods following migrations are critical for an IM’s problem solving ability. It is during those periods that most of the new solutions are constructed, especially for modular or hierarchical problems.
  - Migrant genes can be accepted or rejected or combined into target islands and the level of acceptance is crucial for the type of interaction between islands.

- The behavior depends on both problem characteristic and proper representation/recombination. For the same IM parameter settings, we may observe different behavior (either mixing, rejection or acceptance) for different problems.
- Migration level (size and interval), EA selection and recombination are tightly related in IMs.
  - \* If migrants acceptance is high due to utilizing constructive recombination, islands may be set with a higher level of migration because recombination enhances local diversity. Selection pressure should be adjusted not too high, to support recombination, and not too low, to detect new, good solutions. In such situations, we should observe an inter-island compositional evolution.
  - \* When rejection or migrants is high due to an ineffective recombination, it is critical to avoid a fast domination of any of the partial solutions and by repeatedly exposing partial solutions to each other increase the chances of composing a better solution, and also selecting the best solution. Therefore, islands should be kept separate by a low level of migration. In these cases we should usually observe an incremental evolution within islands.
- While a weaker selection pressure allows more time for recombination to mix migrants and locals, a stronger selection better preserves and spreads rare good solutions.
- Heterogeneity may increase the performance of IMs. First, it may increase



the inter-island distance (and diversity), which is later converted into local diversity by migration. Second, in certain cases heterogeneity may keep evolvability without a need for successful recombination of different solutions, but rather by opening new paths for mutation.

### **9.1.2 Tools for island models**

In addition to the above conclusions, two more contributions result from this work. A set of measures developed in this dissertation helps to analyze IMs, and is summarized in Appendix B. Furthermore, a set of test functions applicable for experiments with IMs was developed and is summarized in Appendix C.

## **9.2 Future work**

While this dissertation clarified some aspects of IMs, many questions remain unanswered, meanwhile new questions appeared. Below I list a subset of possible future directions. Some are just extensions to existing research, while others are more ambitious and less strictly defined.

I have only touched upon the influence of topology. As expected, I have observed that a sparse topology increases diversity, but a more detailed study could further certify these results. I assume that studies with fine-grained distributed EAs should be helpful in this matter.

Also research on IMs using hundreds or thousands of islands could open new possibilities because such IMs may share the characteristics of both fine-grained and coarse-grained models.

We have a better understanding of IM dynamics, but it is still a challenge to set results into concrete guidelines. For example, while I have stressed the relation between the migration level, recombination and selection, additional studies may further understanding of this relation; in particular, the influence of selection on IM behavior.

As we have seen, using IMs has a deep impact on how linked genes are processed, but this area also requires much deeper study. A repeated fixation of genes in islands makes them focus on particular subspaces of the search space. Such fixations might serve as a linkage learning process, fixing some building blocks and focusing on evolving the remaining ones. It would be important to understand precisely whether, due to the system-wide selection, such fixations indeed isolate closely linked structures.

Heterogeneity in IMs is probably such a broad topic that various aspects of it would be enough for separate dissertations. A more thorough study of its influence on the relation between other parameters would be interesting. In particular it would be interesting how heterogeneity could be quantified and if any theoretical boundaries (even if qualitative) could be identified to mark “sweet spots” for heterogeneous IMs.

A good direction for additional research would be on inter-island evolution. It is important to isolate possible behaviors at the global level, depending on the problem studied. Possibly, more than two levels of evolution may be beneficial in some situations — but the challenge would be how to keep a balance between them all. Inter-island evolution naturally has more profound impact on the general shape of solutions, compared to local evolution. It would be desired to see if we can build a system that is able to evolve solutions at multiple levels of generality, from tiny

details to general structure, in the same time. Such system could hopefully address problems not solvable today due to their scale.

Understanding the properties and the dynamics of IM behavior, it should be easier to construct a mechanism for adaptation in IM. Adaptation would allow for run-time adjustments, so that the outcome does not depend so heavily on the starting conditions (compare crossover self-adaptation (Spears, 1995), diversity guided search (Ursem, 2002)). A self-adapting system could be created to observe the balance between intra-island and inter-island evolution or to maximize the number of fit hybrids between migrants and locals.

There is a growing interest in evolutionary systems built as a network of interacting autonomous components or agents (Barabási, 2002; Watts, 2003; Wolfram, 2002; Dezinger and Kidney, 2003). With a few exceptions of complex adaptive systems like Holland's Echo (Forrest and Jones, 1994) and Patchwork models (Krink *et al.*, 1999) not much has been done in this area. Passing the control over evolution, from population level down to single individuals may improve the performance<sup>1</sup> of the system although (as is the case with cellular automata) it may be more difficult to design.

This dissertation aimed at improving our understanding of the IM internal dynamics, which is crucial to predict, and direct their behavior. I hope that the results of my work will contribute to further studies in this area and creation of future (r)evolutionary systems!

---

<sup>1</sup>Local decisions may also be easier for hardware implementations, if such are created.

## **Appendix A: Island models as a remedy for evolutionary algorithm problems**

The failure of evolutionary algorithms on complex problems can have various internal reasons. Here I list some of them and try to see which one of them and how may be alleviated by using IMs. The set of problems is an extended set of the problems identified in the Introduction of “Evolutionary Design by Computers” edited by Peter Bentley (Bentley, 1999). Some problems may overlap, as many features of EAs are interrelated or have several aspects. While I mention many issues here for completeness, I do not address all of them in this dissertation.

First I present a summary in a Table A.1 and then I explain each position. For each one, I present standard solutions, as well as reasons why an IM could help. Note that sometimes IMs create an additional dimension in the parameter space enabling for decoupling related parameters.

### **Vast search space**

The general problem of search algorithms is a huge search space. Being unable to test all the combinations of values, the algorithms use some heuristics to look only at “promising” parts of the search space.

If we cannot diminish the size of the search space, we can at least make sure that the same solutions are not examined over and over again. Also a big search space

Table A.1: EA problems.

| <i>problem</i>                | <i>existing solution</i>   | <i>island models</i>   |
|-------------------------------|--|--|
| vast search space             | long runs<br>weak selection  | searching multiple regions<br>decomposition?   |
| selection too strong          | weaker selection   | maintaining inter-island diversity<br>selection less deleterious<br>heterogeneous selection? |
| selection too weak            | stronger selection   | selection from migration policy<br>genetic drift limited?<br>heterogeneous selection?        |
| crossover<br>non-constructive | increase rate, redesign  | similar individuals in islands<br>heterogeneous crossover?<br>heterogeneous representation?  |
| crossover disruptive          | decrease rate, redesign  | <i>as above</i>  |
| mutation<br>non-constructive  | increase rate<br>change representation   | heterogeneous representation?  |
| mutation disruptive           | decrease rate<br>change representation   | <i>as above</i>  |
| multiple objectives           | increase population size<br>combined fitness                                       | heterogeneous fitness?   |
| constraints                   | prevent, filtrate  | focus on small regions in islands  |
| incorporating<br>knowledge    | use initial parents<br>representation design<br>operators design<br>fitness design | heterogeneous roles?   |
| inaccurate<br>representation  | redesign<br>increase precision   | heterogeneous representation?  |
| bloat                         | limit length<br>use parsimony pressure   | smaller bloat observed?<br>specialized island models   |
| epistasis                     | change representation<br>learn representation<br>redundant representation          | inter-island level tests<br>fixations in local level?<br>heterogeneous representation?       |
| interdependent<br>elements    | restrict search space<br>restarts  | independent evolution  |
| structure and detail          | structured EAs<br>morphogenesis  | inter-island level evolution<br>for structure evolution?<br>heterogeneous representation?    |
| designing<br>embryogenesis    | keep simple  | two levels of evolution<br>adjusts automatically?  |

needs a lot of exploration. Therefore a weak selection together with long runs would be a good traditional choice.

The usage of island models may partially help by focusing on different parts of the search space. Island models, in which every island focuses on a part of the final solution should effectively result in task decomposition and a big decrease of the search space size. Of course there exists the problem of non-linear relations between the parts.

## **Selection too strong**

A too strong selection pressure makes EA prematurely converge to a suboptimal solution. If only the very best individuals survive, new ways of solving a problem (new combinations of alleles) get killed before they manage to adjust and become competitive. Good building blocks only compete with each other, and not combine. Evolution has no space to experiment with temporary weaker solutions. Too strong selection kills diversity and the search turns into hill climbing.

Standard EA approach would be to weaken the selection. However, if it is too weak, an EA may never find the solution, just exploring the search space.

Island models can help here at least two-fold. Premature convergence in one island can be prevented by a migration from another. Secondly, as individuals inside an island become more similar and some general structure (although maybe not optimal) is fixed, the structure is not put under selection pressure anymore. Therefore island models lets weaker general patterns to live and get optimized - evolution can experiment with new solutions within a separated island. A possible extension is to

maintain different selection pressure on different islands. The islands may adjust their parameters to the better doing neighbors.

## **Selection too weak**

Too weak selection can significantly prolong the runs. Good solutions may still be discovered by chance, but a weak selection will not actively promote them. Moreover, if the selection is too weak, the stochastic effects, caused by both genetic operators and by randomness in selection, may occur. If the mutation level is not sufficient, *genetic drift* can finally cause the population to stall at some random point.

A standard solution is obviously to make the selection stronger, however doing so we may easily evolve suboptimal solutions. Also a bigger population will lose alleles slower.

Migration policies in which the average fitness of emigrants is higher than the average fitness of replaced individuals naturally increase the selection pressure. Also, having several islands may prolong the time until stochastic losing of alleles fixes genes to some random values, because of limited crossover and each island converging to a different genotype. However, if we want to maintain the same total size of individuals, we will have the islands smaller and the drift will be more visible. An extended island model with different selection pressures in different islands can help setting the correct amount of selection.

## Crossover ineffective/non-constructive

An EA works good, if crossover is able to combine existing solutions and create different ones of hopefully better characteristics. A standard crossover which recombines two totally different solutions, is usually not very effective, because the parts of differently optimized solutions do not match together. Increasing the level of crossover may increase the probability of constructing new building blocks, but it also increases its disruptiveness (see next section) and causes evolution to focus more on building blocks than on whole solutions, which may be undesired for non-linear problems.

Researchers experimented with a huge variety of specialized crossovers, tailored to the needs of particular problems and using some knowledge about the domain. No wonder, such operators usually are much more constructive. However, sometimes we do not have enough information about the nature of a problem, to construct a proper operator.

Crossover between similar individuals is more likely to result in a good solution, although, of course, more similar the parents, less novel the children. Because island models allow for similar individuals to group in separate islands, they can increase the overall success rate of crossover (in terms of creating a fit offspring). Heterogeneous island models, in which there are different operators in different islands could help in two ways. It is possible that some operators turn out to be more appropriate, although we don't know *a priori* which ones. It is also possible that by interleaved usage of different operators new building blocks are created.



## Crossover disruptive

A related issue to the above one is the one of disruptiveness of crossover. Crossover that cuts parents blindly, may disrupt some complex structures that make them fit - it can damage building blocks. Diminishing the level of crossover diminishes its disruptive effect, but also diminishes its constructive power. Moreover, it makes the evolution to focus more on the whole solutions as opposed to building blocks, which can make the search longer.

Again, a construction of specialized crossovers, that preserve values of related genes, helps in this situation. Such operators would try to exchange logically connected genes together.

For island models, the same reasoning as above holds - which is that individuals within an island become similar to each other, and thus crossover is not so disruptive acting on them. Similarly, different recombination operators in different islands may be beneficial.

## Mutation non-constructive

Mutation is in general considered as operator of stronger disruptive than constructive capability (Spears, 1992). However, in ES it plays a major role. Since mutation is generally point-wise, there are not many ways in which we can redesign it, although we can change its strength and rate. If we set mutation too low, there will be no possibility of getting out of a local optimum. Too little mutation may make it difficult for EA to escape from a limited number of solutions constructed from the initial

parents' gene pool. If we set it too high - the evolution will lose ability of operating on delicate genome structures and resemble random search.

Standard ways of dealing with the problem is to either statically, or dynamically set the mutation level observing its success rate. The 1/5 rule coming from ES, where mutation is the only operator to construct new solutions, suggests that 1 child out of 5 should be better than its parent. Hypermutation technique keeps mutation low for most of the time, but every once in a while it dramatically increases the mutation level. Sometimes mutation is not constructing a better building block, because it would have to do it in several steps, and the problem is deceptive. A different representation, with more independent genes could help in such situation.

Island models do not help much in this situation, because mutation operates on a single individual. An extended island model with various representations could help, because a single-point mutation in one encoding may correspond to a general rearrangement of genotype in another encoding.

## **Mutation disruptive**

Similarly to crossover operator, also for mutation its disruption and construction ability are tightly connected and are in some way opposing each other. Too big mutation may be destroying what was evolved, instead of adjusting it - such high level is the prime reason of mutation being disruptive. Very high level of mutation causes reproduction useless, because children are not similar to parents, their fitness may vary greatly from parents' ones and so there is no correlation between them. Another reason of mutation being disruptive is high epistasis (see further) and a

rugged fitness function. In such situation a small mutation may break many building blocks.

The standard counteractions are similar to the ones described previously and they are setting a proper level for the operator, both in terms of how big a local change is, and how often changes occur. Using a different representation may help. A general rule that small changes in genotype should produce small changes in phenotypes applies to the problem. Also, a redundant representation changes the exploration distribution around the solution and in this way it may make mutation less disruptive.

Also in this case, island models do not seem to help much, although different representations used simultaneously can be useful. Representing building blocks by single genes in another representation may help avoid disruption.

## Multiple objectives

Multi-objective approaches produce a large number of solutions on the Pareto front that are neither better nor worse from each other. What is usually sought is a nice distribution of solutions which shows the trade-off between objectives. Finding the true Pareto front can be quite difficult, especially for smaller populations, although the quality of multi-objective algorithms have recently improved.

Naturally a big amount of solutions is the price we pay for advantages of multi-objective approach. An older and sometimes insufficient approach is to use a weighted combination of the objectives as a single function.

Island models are not particularly useful for the problem, unless we assign different fitnesses to different islands. If the migration is frequent enough, or if there is a finely grained chain of islands with gradually changing fitness computing procedure, we can

expect those island models to produce good solutions of the multi-objective problem. Using island model may be seen as an intermediate approach between single combined function and truly multi-objective approach.

## Constraints

In most real life problems, there are always different constraints, which make some solutions illegal. Because the constraints may result from complex relations, the search space may become highly irregular.

There are two main ways to handle constraints - preventing generation of illegal solutions, and/or filtering them when they appear. The first approach is more effective, if we only know how to do it. Preventing the construction of illegal solutions requires wisely designed operators, that do not step out of the legal area. Second approach is easier to implement, however it takes more time, because illegal solutions are constructed in vain, and because it may happen that we have to wait some time before a correct child is finally produced.

It seems that island models should address the issue of constraints, at least to some degree. If different islands explore different niches, they are less likely to produce very different results, which may be invalid. In other words, an irregular legal area of the search space can be described as a sum of much more regular smaller subareas, which in turn may correspond to particular islands. If this was a case, topology of connections between islands would be similar to the general view of genotypic search space. It is not obvious what extensions to island model would be useful for this problem.

## Incorporating knowledge

For each problem we can either put as much knowledge as possible into the system, or let the system work on its own. Both approaches have advantages and disadvantages and in practice some approach in between is used. Of course, adding knowledge seems an obvious choice at the first sight. The search space becomes much smaller, most probably we can parameterize solutions, we can design special operators that are not disruptive and nicely put together already existing blocks. However, if we know everything about the domain - we probably don't need an EA at all. Adding too much knowledge to the system may impose artificial restrictions on the resulting solutions and most probably nothing novel will emerge. There is a second possibility, where we provide the tiniest possible building elements and believe in the power of evolution to produce a fresh and unexpected design. This approach however also fails very often, due to its high complexity - after all we don't want to evolve cars from atoms. Therefore choosing the right level of knowledge is indeed a problem - or maybe it is rather separating knowledge from false assumptions about the final solution.

Standard EAs use a few approaches. One is to incorporate knowledge into representation and operators. Another is to use several fitness functions and either combine them or use in a multi-objective way. Moreover, for complex domains, initial parents can be supplied, to start the evolution in the right place and with lower level blocks already discovered.

Standard island models probably don't help much. Increasing number of islands for multimodal problems would be some way of using knowledge about the domain. However, we can extend the model and differentiate islands. Instead of incorporating

a certain view of the solution, we could try to maintain several of them and maintain islands to perform different roles by focusing on various aspects of the solution. Evolution should be able to promote good designs, that have migrated between islands and “learned” something in each of them.

## **Bad representation**

Some problems are caused by the improper choice of representation. Representation may cover only a subarea of the search space and the optimum may be unrepresentable. Of course, in such case, evolutionary algorithm cannot find the optimum. Such situation may occur for example when we use rough binary representations for real numbers.

An obvious way to correct the problem is to choose a different, better design representation, or at least increase the precision of the current one.

Island models cannot help with this problem. However if we used different representations in different islands, we could come as close to the optimum as possible.

## **Bloat**

It is obvious that parameterizing a problem and using a fixed length genome for every problem is an easy but quite often inappropriate solution. It is difficult to create a good set of parameters, and by doing so, we only search in a subspace of a more general problem. Alternatively, we could allow for an open-ended evolution, where the solutions may grow arbitrarily big, to fit the solution as close as possible. However, of course, managing huge genotypes is difficult and may hinder the evolution. When a

genome grows big, fractions of it become unimportant, and may even not influence the phenotype at all. Although sometimes useful if controlled (see the Neutrality Theory), the uncontrolled growth of meaningless genotype material in genomes, called *bloat*, is undesired.

One of the solutions to counteract bloat is to set a maximum size above which genotypes are illegal. Another possibility is to use a parsimony pressure either in the form of combining size and raw fitness into one function, or by using a multi-objective approach (Luke and Panait, 2002).

Surprisingly there are reports about island models reducing bloat, although there is no clear understanding why this was observed ((Galeano *et al.*, 2002)). We could try to impose different pressures in different islands - the function of some of them would be to shrink representations, whereas others would focus on improving fitness. In such setup we would have to support migrants, or otherwise they would quickly die. Another possibility would be to use different representations and shrink genotypes as a side effect of migrating and transforming (with some possible loss of accuracy of solutions). However this is no different than simply shrinking genomes in a single population, and it would require some knowledge on how to do it, which we don't have in the first place.

## Epistasis/linkage

When designing a representation space, it is often unavoidable to have genes interrelated. In fact in any more complex problem it is difficult to imagine that it would not be so. Regardless whether the dependency of genes, called *epistasis* comes from a non-linear problem or just poorly chosen representation, it presents

substantial difficulty to EAs. It is so, because EAs usually do not know (and often neither do the researchers) which genes are linked and which are not - it is difficult for operators to maintain the relations between separate genes. The problem is that such linkage should be learnt before the alleles are chosen for particular genes, so the evolution at detail level should be slowed down, in comparison to the learning of interconnections.

One solution is to try finding a better representation of a smaller epistasis. There also exists algorithms that use special changeable representations and try to learn the linkage of genes (Goldberg *et al.*, 1989; Harik and Goldberg, 1997). An interesting approach to deal with separated local optima is to create a redundant representation. Having extra space in genome we can have different representations for one phenotype. Assuming we can move between them, one of them may be much closer to other peaks, or maybe contain a smaller number of linked genes. The situation is similar to cartographic problems. Each map always has poorly represented regions (e.g. Earth poles, edges). However, switching maps as you travel helps avoiding such singularities.

In the light of previous remarks, island models should help with linkage, because they should support evolution at different levels. One can imagine that even if the EA of a particular island would quickly converge to some specific structure of genome, other islands could evolve completely different roles of particular genes and different inter-gene relations. A higher-level evolution should then appear as a result of migrations. Heterogeneous island models could explicitly assign different learning roles to different islands. Also different representations could help – a big group of linked genes of one representation can correspond to a single gene (or a small group) in another representation. Island models with different representations of the same



individuals share some properties with redundant representations.

## Interdependent elements

It may happen that in engineering design different parts of solutions are coupled with each other, which will result in an epistatic representation (see above). Physically coupled components present a difficulty, which is internal to the problem and is not just an artifact of a representation or an evolutionary algorithm. Therefore it is hard to alleviate. If some solutions are physically coupled and others are not, then we could try to guide the algorithm for example by using some smart and restrictive parameterization. This, however, may restrict the spectrum of possible solutions as already mentioned in the section on incorporating knowledge.

The alternative way out of the situation would be to widen the search space (so that more unexpected solutions could potentially be discovered) and try restarts to escape from the deep local optima.

Island models may behave similar to EA with restarts, in which different islands may discover different attractors. Further, smaller islands are “lighter” - it is easier to move them in the search space. Therefore it seems that using island models should help. Unfortunately most changes in representation will not have much influence on the solvability of the problem, because as mentioned above, the problem lays deeper, in the very nature of the domain.

## Structure and detail

Sometimes we need a major change in design - usually without changing the behavior of simpler components. We may decide to add one more bay to a building, or one more axis to a truck. Other times we need to tweak a small detail. However, it would be nice if the change to a genotype was always be small (this wouldn't be possible with phenotypical space where big changes would easily kill the smaller features). Therefore some genes should represent more general features, and some should represent details. The genes representing the detail level should however depend on the general design. This is often achieved with morphogenetic approach, where a final solution is grown from a seed, and the genotype may define the seed and regulate the growth. In nature genes can turn on and off other genes, which makes it possible for evolution to use parts of genome as a “scratchpad” for experimenting (although it is hard to explain what would drive the evolution in absence of selection pressure on such “turned off” areas).

A hierarchical structure of a genome, in which some genes decide which other (and how) are going to be used, is commonly used in nature. Stewart claims that such structure, together with a changing environment leads to enhancement of evolvability and makes evolution “cognitive” (Stewart, 1997). He states that such representation together with a small mutation may help genes useful in the future to survive until later generations. This is because small mutation turns on inactive genes infrequently and doesn't expose them to the selection. In this way evolution can discover combinations that do not bring any immediate profit.

Such problems require EAs to operate at different levels of precision. If a given

general solution is much better than the other, then the selection will clearly choose it. However, if a solution is better than another only after adjusting the details, the evolutionary process may be cheated by random details, which influence strongly the fitness of solutions. The problem is that selection will promote the better details, whereas we want the algorithm to choose better general strategy.

The existing approaches would often try a representation, in which a structure can be changed smoothly as a result of detail manipulation - i.e. an unneeded feature would become less and less important, until evolution could safely remove it without affecting other features. Another approach would reduce selection pressure from time to time to allow for more random walk in the search space.

In the case of island models, I have already argued that they should have a property of supporting evolution at various levels. Different islands can converge to different structures and develop them simultaneously. An example of extended island model that uses slightly different technique are Injection Islands, where different islands use representations of different precision. More rough solutions are “injected” into islands of more detailed representation.

## **Designing embryogenesis**

Biology is a source of inspiration for indirect representations, such as a morphogenetic approach, in which the final solution is grown from an embryo according to some rules. The rules may be more or less sophisticated, some of them trying to mimic the biological environment, and others using very simple rewriting rules of symbol systems. Nonetheless, there are at least several things in common. Different genes can affect the meaning of other genes and some of them are used earlier and some

later during the development process. This means that the genomes can undergo both local changes at detail level - if we happen to mutate a gene that is active at the end of the development process - as well as global changes. The global changes surprisingly do not have to disturb the details. For example, in biology, it is possible to grow an organism with duplicated or displaced parts of body, by manipulating genes active at the very early stage of development. The modified parts of body still look normally, when examined at a low level. Developing a good embryo with a working set of rules is a very difficult task, due to the very high nonlinearity of the development process.

Since embryogenesis is a very new research area, there is no agreement as to what makes it working. An obvious advice would be to keep the system relatively simple, so that the development process can be tracked and its behavior to some degree predicted, because if too many factors come into play we don't understand how to control the system. Keeping the system too simple may however limit the expressive power of embryogenesis.

It is very difficult to judge whether island models would help in a morphogenetic system. It is possible that different general set of rules would evolve in different island. They would get optimized by each evolutionary process, and only then complete working subsets of rules would be exchanged between individuals. In this way island models would provide a framework for running evolution at two levels of generality. Particular islands would be "responsible" for more detailed evolution, whereas inter-island exchange would be responsible for evolving higher level structures in genomes.

## Appendix B: Measures

In this appendix I present a set of various measurements I use in my research. I believe that even if some of them were known in the past, it is useful to catalog them in one place to create a toolbox for studying island models.

### **Fitness**

The first, most obvious measurement is the fitness function. It represents what a practitioner is ultimately interested in; however, it doesn't explain well what the reasons are for good or bad performance. I will use at least several methods for reporting fitness, as explained below.

### **Best-so-far and best-in-run**

The *best-so-far* fitness is the best fitness seen so far (in the current run). Although it highlights the general fitness improvement, it can be always derived from the history of current best values just by taking a maximum (or visually drawing a flat line from the highest peak of fitness seen so far). As such, it carries less information than the best-in-the-system curve (see below). In particular, it could happen that the best-so-far fitness was hit by accident and is much higher than any fitness in the future generations. It also could happen that the fitness of individuals dropped considerably, and the measurement would still report the same high value. Formally,

$$\text{best-so-far}(t_c) = \max\{f(\text{ind}, \text{pop}, t) : t = 0 \dots t_c, \text{ind} = 1 \dots M, \text{pop} = 1 \dots N\},$$

where  $f(\text{ind}, \text{pop}, t)$  denotes fitness of  $\text{ind}$ -th individual in  $\text{pop}$ -th island in generation  $t$ , and  $t_{\text{current}}$  is the current generation.

A single *best-in-run* fitness value is the final value of the best-so-far fitness. If  $T$  is the last generation, then we can write

$$\text{best-in-run} = \text{best-so-far}(T).$$

### System-best

A slightly more informative value is the best in the system (*system-best*) fitness. This value is simply the best fitness observed in the current generation in all islands (and so it is equal to the maximum of the best fitness values for each island). Formally,

$$\text{system-best}(t_{\text{current}}) = \max\{f(\text{ind}, \text{isl}, t_{\text{current}}) : \text{ind} = 1 \dots M, \text{isl} = 1 \dots N\}.$$

It shows any improvement found in the current generation, even if it is below the limit of the previously found best-so-far value. However, it still tells us little about the dynamics of the search because it does not show what is happening “under the surface” with the contents of the islands.

Whereas system-best value may show progress due to evolution, migrations will have no immediate effect on it, because moving good individuals from one island to another does not change the overall best fitness in the system.

## Island-best

Another measurement is the best fitness for each island, in short *island-best* fitness.

Formally,

$$\text{island-best}(t_{\text{current}}, \text{isl}) = \max\{f(\text{ind}, \text{isl}, t_{\text{current}}) : \text{ind} = 1 \dots M\}.$$

Usually I average island-best over all islands,

$$\text{island-best}(t_{\text{current}}) = \frac{\sum_{\text{isl}} \text{island-best}(t_{\text{current}}, \text{isl})}{N}.$$

This measurement is obviously always less or equal to the system-best fitness. If a migration of better individuals occurs into an island (such that the best migrant is better than the best local), we will see a sudden increase of this measure. On the other hand, if one island evolves a better individual, it has a smaller impact on the average than in the case of the system-best value. For heterogeneous IMs it makes sense to draw this measurement separately for each island.

## Average

Finally yet another measurement is the average fitness in an island. Formally,

$$\text{average-fitness}(t_{\text{current}}, \text{isl}) = \frac{\sum_{\text{ind}} f(\text{ind}, \text{isl}, t_{\text{current}})}{M}.$$

Again, very often I average this value over all islands,

$$\text{average-fitness}(t_{\text{current}}) = \frac{\sum_{\text{isl}} \text{average-fitness}(t_{\text{current}}, \text{isl})}{N}.$$

Because it takes into consideration all the individuals, it is more representative of the current state of the measured islands. Furthermore, average fitness is often used in statistical analysis. A drawback of this measurement is its slow reaction to changes. A significant part of an island must have changed its fitness for the average fitness value to noticeably change its value. On the other hand, when migrants have a different average fitness than locals it is possible or easy to see an influence.

Average fitness is prone to the oversensitivity to the extreme individuals. A really bad or good single individual can affect the average value more than the majority of other individuals. A median value could help in this situation, but for my purposes the average value is sufficient.

### **Number of times optimum reached**

Values to which EAs converge are determined by fitness landscape, which may be very complex. Therefore, sometimes it makes little sense to take mean island-best fitness or island-average fitness, and it may be more useful to count how many times the global optimum is reached. This way we avoid a bias resulting from distribution of fitness values of sub-optimal solutions.

### **Selection intensity**

Selection intensity is usually measured to show how much the population average fitness increases due to some selection method. The value is normalized with regard to the fitness standard deviation prior to using the selection operator, and is created with the assumption of the Gaussian distribution of fitness values. The formula for computing selection intensity is:



$$I = \frac{\text{average-fitness}(t + 1) - \text{average-fitness}(t)}{\sigma(t)},$$

where  $\sigma(t)$  is the standard deviation of fitness values at generation  $t$ .

Even if the original assumptions are not met, I still use the same computation to represent relative improvement of the average fitness, with regard to the spread of individual fitness values. Furthermore, this improvement must not only be due to the selection operator, but can be computed in general. For example, I will compute this *selection intensity* to report the changes caused by migrations.

A zero selection intensity denotes a stable situation. A positive value denotes an improvement to the overall fitness. If the fitness values of individuals were spread a lot, then it is easy to observe a little improvement, maybe due to the stochastic reasons — and the selection intensity will be small. If the population was very much converged and the fitness spread was small, then even a small improvement is significant — and the selection intensity will be large.

This measurement may be substituted by two separate measures, one for evolution and one for migration. The evolution-related measurement takes into consideration only consecutive average-fitness values resulting from evolution from generation to generation, and disregards (skips) the change resulting from migration. Under this approach can see improvement due to evolution following migration, and not migration itself. The migration-related measurement takes into consideration only changes to average fitness caused by migration, and is a good indicator of whether there is an improvement of fitness due only to the insertion of migrants, regardless of what happens to them next.

This method of computation should be treated more as a general indication of what is happening inside a population/island, rather than a rigorous measurement. Converged islands have a very small fitness variance, and for such cases even small differences in the average fitness result in a high selection intensity. It is even more visible for smaller islands, and if a whole island converged to a single individual, selection intensity becomes undefined. Also, when I average results over multiple islands and runs, those cases when selection intensity is much higher, dominate the average. For example, we will see that for stabilized islands (average) selection intensity often drops below zero. This may be caused by the fact that an average fitness decreases more often for converged islands and increases for less converged islands. Finally, I use nonparametric selection procedures, so computing selection intensity should be treated with caution.

## **Number of individuals**

For some analysis (e.g. analyzing drift in absence of any particular fitness function) I simply count the number of distinct individuals either in the system,  $w(t)$ , or in an island/population,  $m(t)$ . Instead of giving an absolute number, it is sometimes more useful to refer to percentages, which I denote  $\bar{w}(t)$  and  $\bar{m}(t)$  respectively. As islands converge, and either whole individuals or particular alleles are gradually lost, many individuals become similar and finally identical genetically. This measurement is related to diversity (see next page), but doesn't analyze distances between genotypes or phenotypes.

If two identical individuals were created in two islands, they would behave identically. However, to understand the flow of genes between individuals and islands

it is useful to track each individual separately. Therefore, when counting unique alleles (and individuals), I assume that for a given locus each initial individual has different allele in the whole system. In practice this is implemented by genes that can have any integer value (as opposed to e.g. binary representation), and each initial individual consists of alleles equal to a unique ID of this individual. Thus, in my experiments, two individuals are counted as identical only if they consist of alleles coming from the same initial individuals. This approach does not change the computation of diversity in the population, which is based on actual gene values.

Counting all alleles as unique has some consequences. One is that the convergence in my experiments occur slower than if I simply looked at gene values, which might be equal even if coming from different initial individuals. Another side effect is that the number of initial unique individuals ( $w(0)$  and  $m(0)$ ) is equal to the island size  $M$  (or system size  $W$  respectively), because creating duplicates by chance is impossible. In reality those numbers are limited by  $L^k$ , where  $L$  is genome length, and  $k$  number of possible alleles per locus. Nevertheless these measures are mostly used for a theoretical analysis in Chapter 5, and therefore, the discrepancy with more real-life EAs is acceptable.

## Diversity

Diversity is a measurement that looks deeper inside islands since it tells us about the spread of genomes. Although, “on the surface,” the behavior of an algorithm may look similar, and individuals may have comparable fitness, diversity tells us how much the islands have converged, and thus what are the chances of a qualitatively new solution being evolved soon.

## Local diversity

For computing diversity inside an island, I compute standard deviation of its genotypes (they are fixed length) (Morrison, 2002) and divide by  $\sqrt{L}$  for normalization. For binary genotypes, I treat them as if these were genotypes with possible values of 1.0 and 0.0. The formula for island  $i$  is:

$$\text{div}_i = \sqrt{\frac{\sum_{j=1}^M \|x_{ij} - c_i\|^2}{LM}},$$

where  $M$  is the number of individuals,  $L$  is the length of genotypes,  $x_{ij}$  is the  $j$ -th individual in the  $i$ -th island and centroid  $c_i$  is given by the formula:

$$c_i = \left( \frac{\sum_{j=1}^M x_{ij,k}}{M} \right)_{k=1\dots L} = \frac{\sum_{j=1}^M x_{ij}}{M}.$$

For binary genomes the following transformation holds ( $x_{ij,k}$  is the  $k$ -th gene of  $x_{ij}$ , and  $c_{i,k}$  is the  $k$ -th coordinate of the centroid  $c_i$ ):

$$\begin{aligned} \frac{\sum_{j=1}^M \sum_{k=1}^L (x_{ij,k} - c_{i,k})^2}{M} &= \frac{\sum_{k=1}^L \sum_{j=1}^M (x_{ij,k} - c_{i,k})^2}{M} = \\ \frac{\sum_{k=1}^L ((\sum_{j=1}^M x_{ij,k})(1 - c_{i,k})^2 + (M - \sum_{j=1}^M x_{ij,k})(0 - c_{i,k})^2)}{M} &= \\ \frac{\sum_{k=1}^L (M c_{i,k}(1 - c_{i,k})^2 + (M - M c_{i,k})c_{i,k}^2)}{M} &= \\ \sum_{k=1}^L (c_{i,k}(1 - c_{i,k})^2 + (1 - c_{i,k})c_{i,k}^2) &= \sum_{k=1}^L c_{i,k}(1 - c_{i,k}), \end{aligned}$$

and computing  $d$  simplifies to

$$\text{div}_i = \sqrt{\frac{\sum_{k=1}^L c_{i,k}(1 - c_{i,k})}{L}} = \sqrt{\frac{c_i^T(1 - c_i)}{L}}.$$

Diversity inside each island may differ, but I generally report an average local diversity  $\text{div}_L = \frac{\sum_i \text{div}_i}{N}$ .

When an island converges to a single individual, mutation induces some level of diversity. By computing diversity, I am in fact measuring the standard error for real-valued representation and Gaussian mutation with variance  $\sigma^2$ , and the local diversity is on average  $\sigma$ . If only some genes are mutated (e.g.  $1/L$  ratio), then the local diversity will be appropriately smaller. For binary representation, and bit-flip mutation at a rate of  $1/L$ , on average  $1/L$  fraction of individuals will have a mutated gene (bit) for a given locus, and therefore the local diversity will be

$$\text{div}_i = \sqrt{\frac{1}{L} \frac{L-1}{L^2}} = \frac{\sqrt{L-1}}{L} \approx \frac{1}{\sqrt{L}}.$$

## Global diversity

Global diversity  $d_G$  treats individuals from all islands as one pool and computes their diversity, so it is system-wide. Similarly, I compute a global centroid:

$$c = \frac{\sum_{i=1}^N \sum_{j=1}^M x_{ij}}{NM},$$

$$\text{div}_G = \sqrt{\frac{\sum_{i=1}^N \sum_{j=1}^M \|x_{ij} - c\|^2}{LNM}}.$$

If islands have high local diversity, there must also be high global diversity, but not vice-versa. In fact to simplify, is a sum of local diversities in populations and a diversity resulting from populations being different from each other. This is explained deeper in the next paragraph.

### Inter-island diversity

Differences between local and global diversity help us understand whether islands converge to different suboptima or not. To analyze it, the total variation can be divided into “within islands” and “between islands” parts. The following equation expresses this fact (it is easy to derive, substituting  $x_{ij} - c = x_{ij} - c_i + c_i - c$ ):

$$\sum_{i=1}^N \sum_{j=1}^M \|x_{ij} - c\|^2 = \sum_{i=1}^N \sum_{j=1}^M \|x_{ij} - c_i\|^2 + M \sum_{i=1}^N \|c_i - c\|^2.$$

The  $c$  variable is now the global centroid and  $c_i$  are island ones. The left-hand side is in fact from the definition equal to  $LWd_G^2$  and the first component of the right-hand side is equal to  $LM \sum_i d_i^2$ . Therefore, inter-island diversity is equal to

$$\text{div}_P = \sqrt{\frac{\sum_{i=1}^N \|c_i - c\|^2}{LN}} = \sqrt{\frac{W \text{div}_G^2 - M \sum_i \text{div}_i^2}{MN}} = \sqrt{\text{div}_G^2 - \frac{\sum_i \text{div}_i^2}{N}}.$$

If I assume that islands on average have similar diversities (as it should be with homogenous islands), then I can approximate  $\text{div}_i \approx d_L$ , therefore  $\frac{\sum_i \text{div}_i^2}{N} \approx \text{div}_L^2$ , and so

$$\text{div}_P \approx \sqrt{\text{div}_G^2 - \text{div}_L^2}.$$

## Represented individuals and islands

Each island may converge to a unique combination of gene values. Each island may preserve or introduce unique alleles absent in other islands. Therefore, it is important to understand how individuals and genes flow between islands. I mark with different IDs genes originating in each initial population of each island<sup>1</sup> and track each gene's flow in the system. I will say that a given island is *represented* in another island if individuals or alleles from the first island are present in the second island. Such individuals (or genes) I will call *representatives* or *representative genes* and I measure the degree to which they are present. Several variants of measurements are possible, as described below.

For a given island  $i$ , let me denote by  $k_{i_2 j_2}^{i j}(t)$  the number of genes from individual  $j_2$  in initial island  $i_2$  in genome  $j$  of island  $i$ , after generation  $t$ . If I drop an individual index, then I mean a whole island, e.g.  $k_{i_2}^i(t)$  is the number of representatives from island  $i_2$  in island  $i$ , after generation  $t$ .

## Represented populations in a population

One of the most obvious measures is to take each individual in an island, see what initial island population it originates from, and count the number of islands

---

<sup>1</sup>For these measurements a population refers to a single island.

represented in a given island, called *represented populations in a population* or RPP.<sup>2</sup>

RPP can be written as

$$RPP_i(t) = |i_2 : k_{i_2}^i(t) > 0| = |i_2 : \sum_{j=1}^M \sum_{j_2=1}^M k_{i_2 j_2}^{ij}(t) > 0|.$$

RPP is a sufficient indicator of how much the current island population is a mixture of individuals from other island populations. When recombination is in effect, original individuals are obviously not preserved. In these cases I track single genes and extend *RPP* to the number of initial populations that have at least one allele present in any individual in the measured island. In the case of no recombination, this extended definition matches the one counting individuals. Obviously we have  $RPP \leq \min\{N, M \cdot L\}$ , since there are  $N$  islands, and  $M \cdot L$  genes in all individuals of an island.

### Represented populations in an individual

When there is no recombination, each individual originates from some initial island population. However, when recombination splits and joins individuals, genes from different initial populations may be present in a single individual. Therefore I introduce a second measurement counting the number of *represented populations in an individual*, *RPI*, which is written:

$$RPI_{ij} = |i_2 : \sum_{j_2=1}^M k_{i_2 j_2}^{ij}(t) > 0|.$$

---

<sup>2</sup>Alternatively, I could count the percentages for each initial population, but *RPP* gives just one number.



We have  $RPI \leq \min\{N, L\}$ , since this measurement considers  $L$  genes of each individual separately.

### Represented individuals in an individual

Similar to tracking initial populations, one can track initial individuals. In these cases, I designate genes in each initial individual and observe how they spread in the system due to recombination and migration. Even in a single-population model, as a result of recombination, genes are mixed inside individuals, and the average number of represented individuals in any individual grows. This number will be reported as a third measurement, *represented individuals in an individual* or RII:

$$RII_{ij} = |(i_2, j_2) : k_{i_2 j_2}^{ij}(t) > 0|.$$

Without migration we have  $RII \leq \min\{M, L\}$ . When migration is in effect, each initial individual from any island has a theoretical chance to have its gene in the measured individual, and so in general  $RII \leq \min\{N \cdot M, L\}$ .

### Represented individuals in a population

For compatibility, the last measurement is to measure the number of initial individuals in an island (i.e. in any individual of a measured island), *represented individuals in a population* or RIP:

$$RIP_i = |(i_2, j_2) : \sum_{j=1}^M k_{i_2 j_2}^{ij}(t) > 0|.$$

With no migration, we have initially  $RIP = M$ , which gradually decreases due to convergence. No recombination results in a similar curve, although migration may exchange original individuals for ones from different islands. With migration and recombination, we have  $RIP \leq \min\{N \cdot M, M \cdot L\} = M \cdot \min\{N, L\}$ , since I track all individuals, considering all genes of a given island individuals.

### Relations between above measures

Obviously, we always have  $RPI \leq RPP$ , because more initial populations can be represented in a whole island than on average in a single individual. On the other hand, even assuming that different individuals or islands are represented in each individual/island, we have upper boundaries resulting from the size of island:  $RIP \leq M \cdot RII$  and  $RPP \leq M \cdot RPI$ . However, because of the convergence of islands to single individuals,  $RPP$  will gradually drop during a run until it approaches an average  $RPI$  (the  $RPI$  average over multiple runs will remain stable if the choice of an individual to which an island converges is random). Similarly, we obviously have  $RII < RIP$ , and due to convergence the average  $RIP$  will gradually drop toward the average  $RII$ . In addition, we naturally have  $RPP \leq RIP$  and  $RPI \leq RII$  because each represented individual originates in a single island.

All the above relations are shown in the following diagram:

$$\begin{array}{rcc}
 RPI & \leq & RPP & \leq & N, M \cdot RPI \\
 | \wedge & & | \wedge & & \\
 RII & \leq & RIP & \leq & N \cdot M, M \cdot RII \\
 | \wedge & & | \wedge & & \\
 L & & M \cdot L & & 
 \end{array}$$

Intuitively, whereas RPP put more emphasis on migrations, all three measurements RII, RPI and RIP indicate more accurately how well recombination mixes genes after migrations. Nevertheless all of them describe the combined effect of recombination and migration spreading alleles widely in the system.

## Distance from migration equilibrium

We can measure how well islands are mixed with each other due to migrations, by measuring how close they are to a “perfect mixing.” To do so, for a given island I first measure the percentage of individuals originating in other islands. Let us call the point  $(\frac{1}{N}, \frac{1}{N}, \dots, \frac{1}{N})$  representing the perfect migration-mixed island, the *equilibrium point*. At this point, an island has the same number of individuals originating initially from any other island. With finite island populations finding such a point may be unachievable because the percentages must be a multiplicity of  $\frac{1}{M}$ .

### Population distance

I compute a distance  $d$  from the equilibrium point for each island, as follows:

$$d_i(t) = \sqrt{\sum_{i_2=1}^N \left( \frac{k_{i_2}^i(t)}{M} - \frac{1}{N} \right)^2}.$$

Since original individuals may not be preserved when recombination is in effect, I extend the definition of equilibrium to a state of island population in which the number of representative *genes* from each island is the same. If there is no recombination, the extended definition matches the previous one (assuming same

lengths of genomes). Now, I can compute the island's distance from the equilibrium point in the following way:

$$d_i(t) = \sqrt{\sum_{i_2=1}^N \left( \frac{\sum_{j=1}^M k_{i_2}^{ij}(t)}{ML} - \frac{1}{N} \right)^2}.$$

Distance  $d(t)$  gives a good estimate of how well the genotypic material of an island is mixed with material from others.

### Average individual distance

Knowing that an island is in a state of equilibrium still does not show how much individuals in an island are mixed with individuals from other islands. In fact, genes may be differently split between individuals, so there are many possible states of an island, which we can describe as being in an equilibrium point. We would expect that with perfect mixing, in each individual there is an equal number of representative genes from each island, and with no damage to previous measurements, I narrow the definition of an equilibrium to such state. Similar to  $d$ , we can plot an average distance  $d'(t)$  of individuals from this equilibrium, as shown in this formula:

$$d'(t)_i = \frac{\sum_{j=1}^M \sqrt{\sum_{i_2=1}^N \left( \frac{k_{i_2}^{ij}(t)}{L} - \frac{1}{N} \right)^2}}{M}.$$

Measure  $d'$  shows how much individuals of islands are mixed with individuals from other islands, or more precisely how close they are on average to a perfect equilibrium

point, where each individual has the same number of alleles coming from each island (from some individual initially from that island).

If  $d'(t)$  is small, so must be  $d(t)$  since if every individual has genes from many islands, then the island containing those individuals has a good mixture of representative genes even more. In particular, if all individuals are in gene equilibrium, then the island must be in gene equilibrium as well. The opposite is not necessarily true, since even though particular individuals may come from various islands making the contents of an island very heterogeneous, each of them may still be quite homogeneous. Also, the genome length plays a role here, because with shorter genomes the right percentage is more difficult to keep and the  $d'(t)$  distance will be larger.

## **Gene measurements**

A few more measurements are possible which compute statistics about alleles of single genes.

### **Average gene value**

An obvious possibility is to measure an average of alleles present in a given locus or over all loci. This measurement indicates the pace of gene convergence and also the direction of such a convergence.

### **Genes fixation**

The decrease of diversity and individuals becoming more and more similar to each other may be measured not only at the individual level, but as well at gene (or locus)

level. In other words, for a fixed-length individual, various alleles exist for each locus, and we can count their number in an island. These numbers will decrease for each locus, until there is only one allele left, present in all individuals. Therefore, the following measurements are possible:

- the number of distinct alleles for each gene.
- the number of fixed genes (i.e. the ones for which the previous measurement returns one).
- the average value of all fixed genes.
- the average value of remaining alleles, for each gene (useful specially in case of real-valued representation).
- the variance of the remaining alleles, for each gene.

### **Allele survivability**

After each migration, migrants are included into the standard evolution cycle of the target island. They are recombined, mutated and evaluated. The chances of survival of their alleles depend on how many of them there are, how good they are compared to the rest, and whether they can produce good children in combination with other individuals (probably from the target island). Even if all migrants as wholes are removed, their alleles may spread in the island.

We can measure the average number of copies or percentage of new alleles in an island. Just after the migration this measure will be equal to the percentage of new migrants in the island. It may grow or decrease in next generations. Measuring allele

survivability should help us understand how successful the migrations are and thus how much they can be credited for a better or worse behavior of the whole system. When the selection is random, the expected allele survivability of both migrants and original individuals (*locals*) is proportional to their number.

A small problem in measuring allele survivability is that individuals mutate. It may be a smaller concern for real numbers with Gaussian mutation, since the mutated version is somehow similar to the original one, so one would hope its behavior is similar. For binary genes the situation is worse, since after mutation gene values turn into opposite values. Nevertheless, I count mutated genes the same way I would count them if they did not mutate. The linkage with other genes may be very complex and thus it still makes sense to track the origin of a given allele (because its survival, even if mutated, may be due to its role in the context of other genes).

There are at least several possible ways to measure the allele survivability and each has some advantages and disadvantages. For example, one can mark each allele to denote whether it has been in any migration in the past and report the percentage of them, or count the number of migrations for each allele and report the average. This could indicate whether the same alleles are successful in consecutive migrations of a given run. I have experimented with the above, but found the following ones more useful:

- mark alleles present in a migration at a given generation and keep track of them for a length of migration interval.
- mark alleles present in a migration at a given generation and remember its index until this allele is present in another migration. This is an extension of the first

possibility which allows for limited tracking of the origins of alleles.

These measurements help understand the influence of migration. In the first option, with short intervals it is difficult to see what happens to the genes, because the markers are reset with every new migration. Therefore, I will generally use the second option. One could also keep multiple markers for each gene, so that its history could be deduced and any statistics computed, however, it is impractical in implementation.

## **Types of individuals**

We can classify individuals as belonging to various types, and count the number of them for each type. The distinction can be made either based on individuals' genomes (e.g. first half of the genome has certain value), or origin (e.g. migrants versus locals). In addition, one can compute the change in the average-fitness due to certain individual types, from generation to generation.



## Appendix C: Test suite

In experiments described in this dissertation I use various test problems. Even though I do not use all of the test problems in any single experiment, I gather them here to create a complete set. It will enable the reader to focus on the experiments and results and one can always return here for details. All problems use fixed-length string representations, because these representations are easier to analyze and in the same time show various phenomena characteristic for IMs.

### **Pseudo-landscapes**

Some experiments may be done without any explicit fitness function, as described below.

#### **No function or flat landscape**

Initial experiments with drift are often performed in absence of any actual fitness function. This is possible, because in such study the fitness does not play any role and the only thing that counts is the number of distinct individuals in the populations. They are selected randomly to survive or migrate.

#### **Arbitrary assigned fitness**

Another set of experiments is possible without any explicit fitness function, but with fitness values arbitrarily assigned to individuals. Without much loss of generality one

can assume that every individual has a unique fitness value. If this is the case, and the selection depends only on the rank (as in tournament selection), and not on the fitness value itself (as would be e.g. in fitness proportional selection), then without further loss of generality, we can assign to individuals fitness values being numbers between 1 and  $N$ , where  $N$  is the number of individuals in the population.

## Simple functions

I used very simple functions deliberately, so that we can understand exactly why an IM algorithm behaves the way it does. In the same time, I wanted them to possess at least some features characteristic for more complex problems. They are all maximization problems.

The naming scheme is not consistent because it maintains names from previous publications. In  $IM_n$  functions, “IM” stands for “island model.”

### Function IM1

Function IM1 consists of one wide low peak and one narrow and high peak. The domain for both arguments is  $[0, 1]$  and the formula for the function is:

$$\begin{aligned} IM1(x_1, x_2) = & \max(0, -(2x_1 - 1)^2 - (2x_2 - 1)^2 + 1, \\ & -(5(x_1 - 1))^2 - (5(x_2 - 1))^2 + 1.25). \end{aligned}$$

It can be used to measure to which peak islands will converge, and how these convergence processes interact (see section 4.4.1).

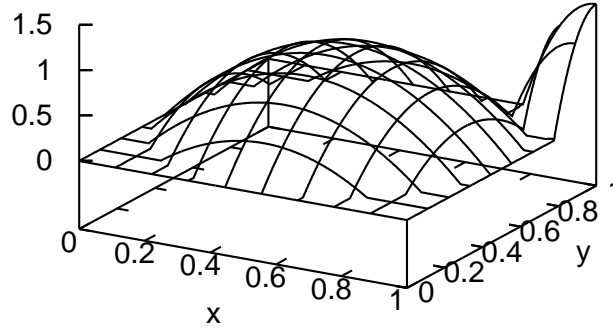


Figure C.1: Function IM1.

### Function IM1'

Function IM1' has two peaks for the first argument. The second argument is independent and its contribution is parameterized by  $a$ . IM1' is defined over  $[0, 1]^2$ , by

$$\text{IM1}'(x, y) = \max(1 - x, 10x - 8.9) + ay.$$

It can be used to measure how one gene convergence affects another gene convergence. The function is introduced in the section 4.2.1.

### Function IM1''

Function IM1'' is a function in which arguments have various strength of contribution. It is defined over  $[0, 1]^L$  and given by

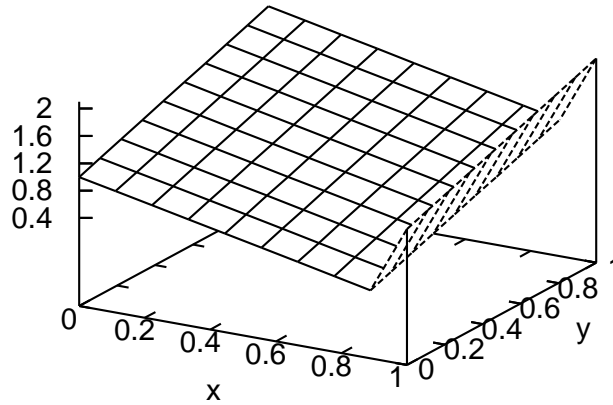


Figure C.2: Function  $IM1'$ ,  $a = 1$ .

$$IM1''(x) = \sum_{i=1}^L (2x_i)^i.$$

Selected components of this function are shown in Fig. C.3. Similarly to the  $IM1'$  function, the  $IM1''$  function can be used to measure independence of gene convergence at different loci (see the section 4.2.1).

## Function $IM2$

This function consists of two wide and one narrow peak, reachable by a crossover of solutions from the wide peaks. The formula for the function (defined over  $[0, 1]^2$ ) is:

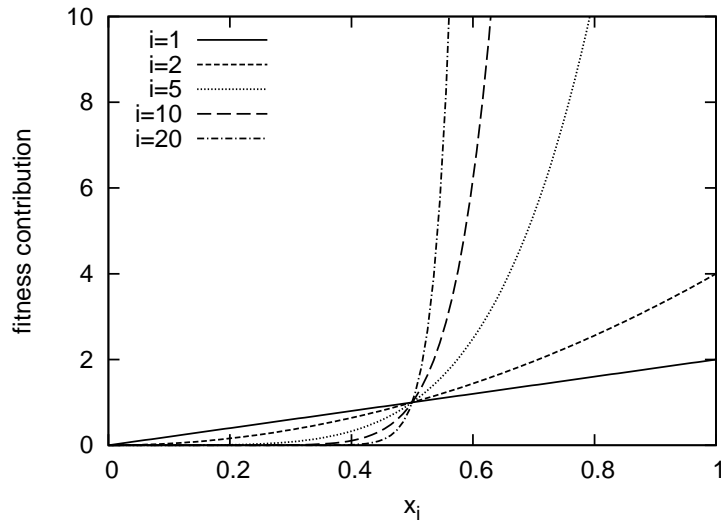


Figure C.3: Gene contributions for the IM1'' function.

$$\begin{aligned}
 \text{IM2}(x_1, x_2) &= \max\left(0, -x_1^2 - \left(\frac{x_2 - 1}{2}\right)^2 + 1, \right. \\
 &\quad \left. -\left(\frac{x_1 - 1}{2}\right)^2 - x_2^2 + 1, \right. \\
 &\quad \left. -(30(x_1 - 1))^2 - (30(x_2 - 1))^2 + 1.25\right).
 \end{aligned}$$

The IM2 function can be used to measure the effectiveness of recombination and interaction between islands in an IM (see the section 4.4.5).

### Function IM2'

The IM2' function may be seen as a modification of the IM2 function, in which the wide peaks are of different heights. It can be used to measure the sensitivity of an

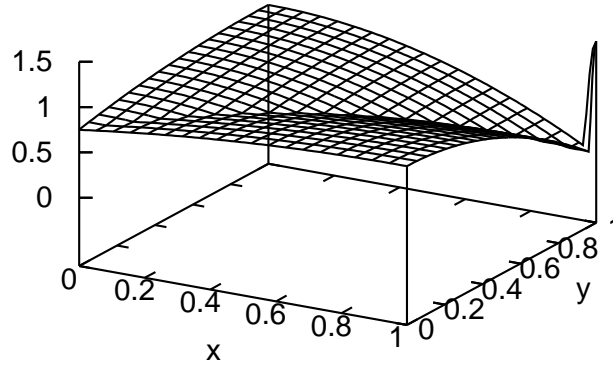


Figure C.4: Function IM2.

IM algorithm to such asymmetries (see the section 4.5.2). It is given below and plot in Fig. C.5.

$$\text{IM2}'(x, y) = \begin{cases} 3 & \text{if } x, y > 1 - \frac{1}{10^3} ; \\ \max(x^2(y-1)^2, 2(x-1)^2y^2) & \text{otherwise} . \end{cases}$$

### Function IM-Trap

The IM-Trap function is a binary function. It is designed to be deceptive. While the maximum of  $L$  is reached at  $0^L$  and  $1^L$ , it is easy for a recombinative EA to get stuck at a plateau with  $0.9 \cdot L$  value (see the section 4.5.1). The function is given by the following formula:

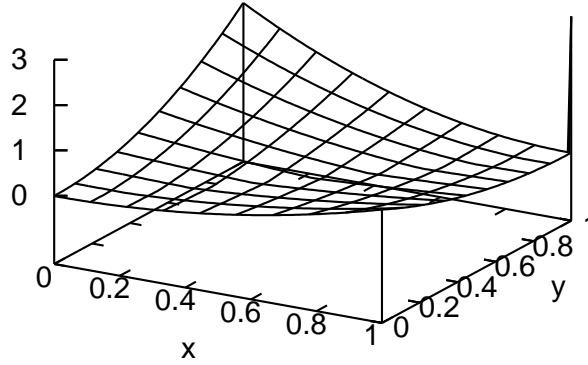


Figure C.5: Function IM2'.

$$\text{left}(a) = \max_{i=0 \dots L} \{i : \forall j \quad 1 \leq j \leq i \quad a_i = 0\},$$

$$\text{right}(a) = \max_{i=0 \dots L} \{i : \forall j \quad 0 \leq j \leq i - 1 \quad a_{L-i} = 0\},$$

$$\text{IM-Trap}(a) = \begin{cases} 0.9 \cdot L & \text{if } \text{left}(a) > \frac{L}{4} \text{ and } \text{right}(a) > \frac{L}{4}; \\ \max\{\text{left}(a), \text{right}(a)\} & \text{otherwise.} \end{cases}$$

## Modular and hierarchical functions

In modular problems different subgoals must be found independently, and then combined into final solution. Hierarchical problems are an extension of modular functions, in which the next level can be solved only after finding solution to a lower level. In general, complex problems, where a high-level structure of solution must be found before working on the details can also fit into this category. Modular and

hierarchical functions can be used to test compositional evolution, which heavily depends on successful recombination.

## Quadratic function

This is a common quadratic function, but with asymmetrically chosen limits (it is a function  $f : [a, b]^n \rightarrow \mathcal{R}$ , where  $|a| \neq |b|$ ), as used in (De Jong, 2006). I have used a version with  $a = -3$  and  $b = 4$ . Such function is multimodal (it has  $2^n$  peaks) and modular. Recombining lower peaks may facilitate discovering higher peaks. The formula is of course given by

$$f(x_1, \dots, x_n) = \sum_{i=1}^n x_i^2.$$

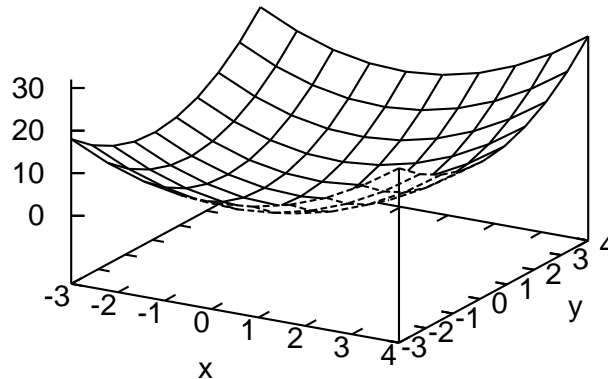


Figure C.6: Quadratic function (two-dimensional case).



## Function IM3

This function is an  $n$ -dimensional equivalent of function IM2. There are  $2^n - 1$  peaks in this function, located at points  $P = \{0, 1\}^n \setminus 0^n$ . The height of each peak depends on the number of 1s in its location, more 1s giving higher value. However, higher peaks are narrower, it is unlikely to find them without crossing over solutions from lower peaks and, therefore, function IM3 is an example of a hierarchical function.

The domain for all arguments is  $[0, 1]$ , and the formula for the function is:

$$\text{IM3}(x) = \max_{p \in P} \text{peak}_p(x),$$

where  $x = (x_1, \dots, x_n)$ , and if  $\sum_i p_i = 1$  then

$$\text{peak}_p(x) = 1 - \sum_{i=1}^n \frac{(x_i - p_i)^2}{n},$$

and if  $\sum_i p_i > 1$  then

$$\text{peak}_p(x) = \frac{n-1}{n} + \sum_{i=1}^n \left( \frac{p_i}{n} - k p_i (x_i - 1)^2 - \frac{(1-p_i)x_i^2}{n} \right).$$

I have set  $n = 10$ ,  $k = 100$ . This function was used in earlier experiments, and was later substituted with the IM6 function.

## Function IM6

Function IM6 was created as a simpler and faster “version” of the IM3 function. It is also a modular function. The basins of attraction of the higher peaks are now

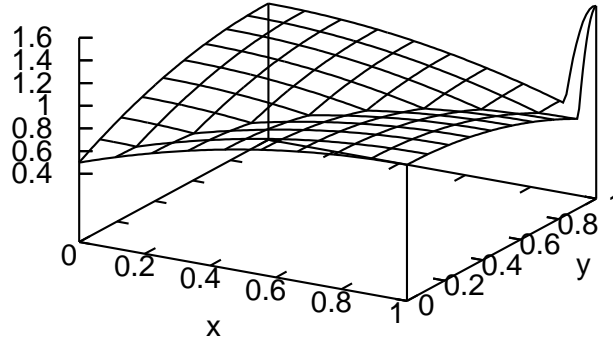


Figure C.7: Function IM3 (two-dimensional case).

smaller corresponding to the previous function. The IM6 function is defined by a simple formula

$$\text{IM6}(x) = \prod_{i=1}^n \frac{(x_i^2 - 0.45)^2}{0.3025}.$$

This function also has local optima in each corner of  $[0, 1]^n$ , and the global optimum in  $(1, \dots, 1)$ , equal to 1. A two-dimensional version is shown in Fig. C.8.

### **Function H-IFF**

The H-IFF function is a hierarchical IFF function proposed by Richard Watson (Watson, 2006). The domain is binary. The evaluation procedure splits genome in pairs of genes and checks whether each pair of adjacent genes consists of the same alleles. If they do, then 4 adjacent genes (composed of two pairs) are checked. If

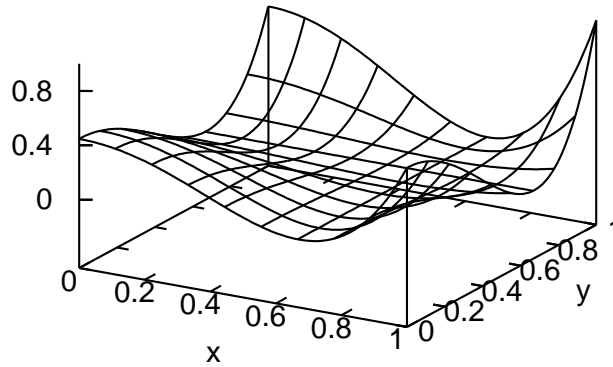


Figure C.8: Function IM6 (two-dimensional case).

they do, eight-tuples of genes (composed of two quadruples) are checked at the next level and so on, until the whole genome (presumably of length being a power of 2) is tested. Each next level has twice fewer potential blocks, but its weight is doubled, so it has the same contribution to the general fitness. This is done on purpose to balance the importance of intra-module and inter-module relations.

The H-IFF function may serve to test compositional evolution, or the effectiveness of evolution by repeated recombinations.

### **Function Deceptive**

The Deceptive function is a concatenation of  $N$  4-bit trap functions (Deb and Goldberg, 1993) (I used  $N = 10$ ). The basic trap block has the following contributions:

$$f(1111) = 30, f(0111) = 0, f(1011) = 2, f(1101) = 4, f(1110) = 6, f(0011) = 18,$$

$f(0101) = 16$ ,  $f(0110) = 14$ ,  $f(1001) = 12$ ,  $f(1010) = 10$ ,  $f(1100) = 8$ ,  $f(0001) = 26$ ,  
 $f(0010) = 24$ ,  $f(0100) = 22$ ,  $f(1000) = 20$ ,  $f(0000) = 28$ .

The total fitness is the sum of all contributions,

$$\text{Deceptive}(x_1, \dots, x_{4N}) = \sum_{i=1}^N f(x_{4i+1}x_{4i+2}x_{4i+3}x_{4i+4}).$$

The trap function values are assigned in such a way, that increasing the number of 0s always increases fitness, but the maximum is 1111. In Fig. ??, I show possible configurations of the trap functions. Lines denote possible one-bit mutations.

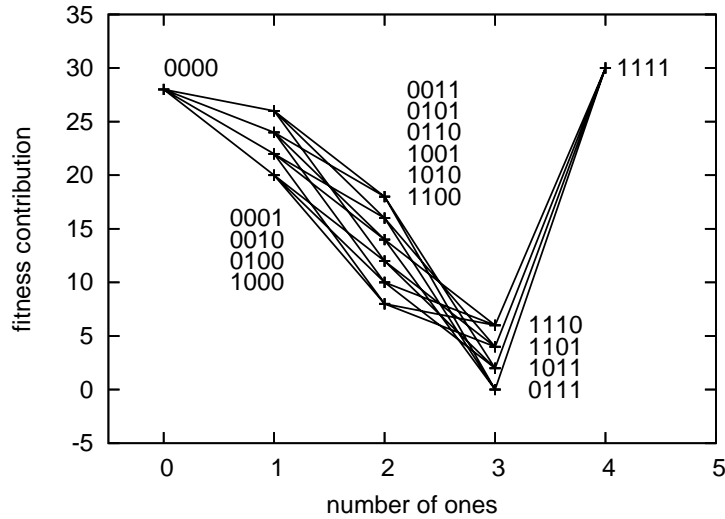


Figure C.9: Possible configurations of a 4-bit trap function.

The Deceptive function can serve to test different recombination operators in conjunction with compositional evolution. To make the function more difficult for standard recombination operators, blocks are often interleaved, that is the first block is located at genes 1,  $N + 1$ ,  $2N + 1$ ,  $3N + 1$ , etc.

## Standard multi-modal functions

I also use several standard multi-modal functions, listed below, for the sake of completeness and comparison with other studies. These are minimization functions.

### Rosenbrock function

The formula for the function is:

$$f(x_1, \dots, x_n) = \sum_{i=2}^n (100(x_i - x_{i-1}^2)^2 + (1 - x_{i-1})^2).$$

The domain for all arguments is  $[-2.048, 2.048]$ . The Rosenbrock function has 2 minima for  $n \geq 4$  (Shang and Qiu, Spring 2006).

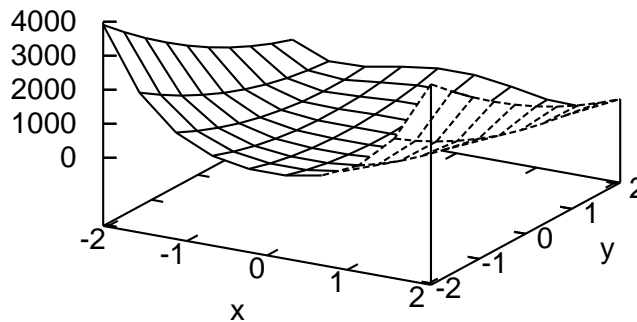


Figure C.10: Rosenbrock function.

## Schwefel function

The formula for the function is:

$$f(x_1, \dots, x_n) = \sum_{i=1}^n -x_i \sin(\sqrt{|x_i|}).$$

The domain for all arguments is  $[-500, 500]$ .

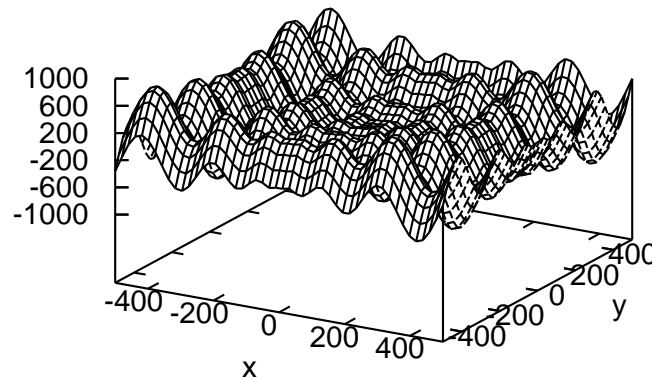


Figure C.11: Schwefel function.

## Rastrigin function

The formula for the function is:

$$f(x_1, \dots, x_n) = 10n + \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i)).$$

The domain for all arguments is  $[-5.12, 5.12]$ .

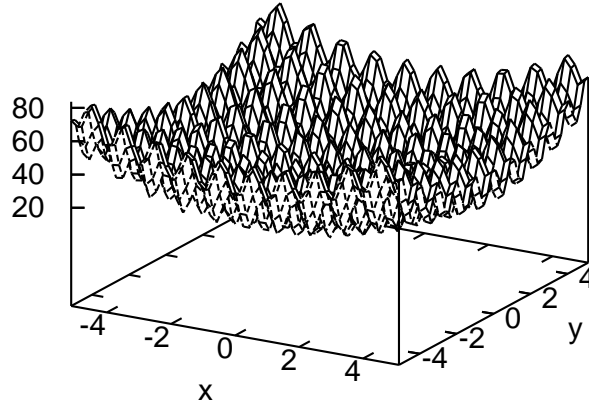


Figure C.12: Rastrigin function.

## Griewank function

The formula for the function is:

$$f(x_1, \dots, x_n) = 1 + \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right).$$

The domain for all arguments is  $[-600, 600]$ .

## Heterogeneity functions

Test functions used for analyzing heterogeneity are described in chapter 7. They are functions in which local minima are located in different places depending on representation. They all operate on binary genomes of length  $n$ . Below I shortly remind the definitions

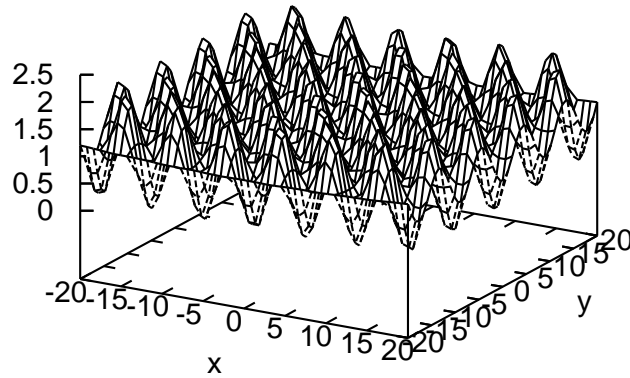


Figure C.13: Griewank function.

$$\text{easy}(a) = \text{hamming}(a, 0^n)/4,$$

$$\text{deceptive}(a) = \begin{cases} 3 * \text{hamming}(a, 0^n) & \text{if } \text{hamming}(a, 0^n) < \frac{1}{4}n; \\ n - \text{hamming}(a, 0^n) & \text{otherwise.} \end{cases}$$

## Function F

Function F is a minimum of two functions, one intended to be difficult for binary representation, and the other for Gray representation.

$$F_1(x) = \text{deceptive}(\text{gray}(x)) + \text{easy}(\text{binary}(x)),$$

$$F_2(x) = \text{deceptive}(\text{binary}(x)) + \text{easy}(\text{gray}(x)),$$



$$F(x) = \min(F_1(x), F_2(x)) .$$

## **Function F2**

Function F2 is a simpler function than F, but maintaining similar properties. It is given by

$$F2(x) = \text{deceptive}(\text{binary}(x)) + \text{deceptive}(\text{gray}(x)) .$$

The binary and gray representation operators may be substituted by any other two representation operators, if needed.

## Appendix D: Recombination revisited

Some combinations of alleles may exist more often than others in a population. If the probability of a certain combination is not a multiplication of the probabilities for each allele, computed as the percentage of such allele in a population, then the population is in *linkage disequilibrium*. Recombination operators, by splitting individuals and “shuffling” alleles, bring populations back to a *linkage equilibrium*, called also *Robbins equilibrium*. A well known Geiringer theorem states that provided that each two genes have a non-zero chance to be split by a recombination operator, such recombination operator will ultimately bring an infinite population to the Robbins equilibrium. Finite populations may not be able to reach this point, but they come as close as possible to this point.

Some recombination operators are more conservative, i.e. they are more likely to preserve combinations of genes, at a cost of smaller probability of trying new combinations of alleles. A good example is a one-point crossover. Other recombinations, like a uniform crossover, do not preserve combinations of alleles, but in doing so they achieve a higher exploration level. It may be sometimes very difficult to decide how “aggressive” recombination one needs, because usually we want to preserve good combinations and in the same time explore new possibilities. Additionally, it is good, if the operator is not only non-destructive for all good combinations, and explorative but also constructive for new good combinations - i.e. it should not search at random, but rather be directed at promising combinations.

Operators that mix more, cause particular alleles to evolve more independently. In particular a perfectly mixing operator would cause each gene to evolve as if in a separate population, even if we use a single population model, because it would not matter in which individual a certain allele is, as the probability of it surviving depends on its average fitness contribution from the whole population (this would correspond to a co-evolutionary system). Therefore if stochastic effects can be disregarded (for example for large populations) using such operator for IMs seems dubious (Watson, 2006). However, very often stochastic effects do play role (see Shifting Balance Theory (Wright, 1932)), and moreover even a uniform operator is not mixing perfectly (see below). More conservative operators make evolution to focus more on longer blocks of genes, and in a special case when there is no recombination (i.e. all individuals are preserved), blocks of the full individual length are compared to each other, without a possibility of rearranging. This shows again, that a proper choice of the recombination operator is needed.

It is tempting to think that a uniform recombination is a perfectly mixing recombination, because it takes alleles at random from both parents. In fact, many authors seems to implicitly suggest so. This is not exactly so, because of at least two reasons. The first one is that each standard recombination operator, including uniform crossover operates on only 2 parents. If we imagine a population consisting of the following genomes:  $\{0^{i-1}10^{L-i}, i = 1 \dots L\}$ , it is obvious that a genome  $1^L$  cannot be achieved in one generation.<sup>1</sup>

The second reason for uniform recombination not being a perfectly mixing operator is that even if it can perfectly mix individuals *potentially*, it is more

---

<sup>1</sup>It would require at least  $\log_2(L)$  generations, if we are extremely lucky to combine the right alleles each time.

likely that percentages of individuals in the next generation will be affected by the previous generation percentages. Imagine the following population: {00, 00, 00, 00, 11, 11, 11, 11}. Both genes have 0.5 percentage of both alleles, so the Robbins equilibrium for such population is {00, 00, 01, 01, 10, 10, 11, 11}. This equilibrium is however unlikely to be achieved in one generation. In fact, the most probable outcome of applying the uniform recombination to the population is {00, 00, 00, 01, 10, 11, 11, 11}, which is easy to check. For example, the expected percentage of 11 individuals is<sup>2</sup>

$$P'_{11} = \frac{1}{4}P_{00}P_{11} + \frac{1}{4}P_{11}P_{00} + P_{11}P_{11} = \frac{1}{4}\left(\frac{1}{2}\right)^2 + \frac{1}{4}\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2 = \frac{3}{8},$$

where  $P_{xy}$  is a percentage of individual  $xy$  in the population. If we were guaranteed that different individuals, that is 00 and 11 are always chosen for recombination, then we could get a perfect mixing. However, same individuals can be chosen repeatedly for recombination. The probability of this happening will be significant for populations with only two types of individuals, which happens often after migrations.

In the presence of selection proper recombination may be quite important. In reality this means that proper building blocks should be created before selection (and drift) manages to fixate all individuals to the same alleles, and sometimes finding better individual depends on the level of exploration. The following example illustrates this, although in this case we do not need to care about preserving any building blocks, so more exploration is always better. Imagine the following population of 48 individuals: 16 individuals with 000 genome, 16 with 001, 16 with 010

---

<sup>2</sup>In each component of the equation I multiply the probabilities of choosing appropriate parents and the probability that the crossover mask will choose the right alleles.

and 16 with 100. Let us assume that 111 is the searched optimum, and further assume a quite strong selection. If the recombination was mixing perfectly (i.e. creating a Robbins equilibrium), it is easy to compute from alleles percentages that it would then produce on average 27 copies of 000, nine of 001, 010 and 100 each, three of 011, 101 and 110 each, and one 111. In the next generation selection could promote 111 as the optimum. If we used a uniform recombination though, the expected numbers would be 22 of 000s, 12 of 001s, 010s and 100s, two of 011s, 101s and 110s and none of 111! Only after yet another generation a recombination of two genomes from 011, 101, and 110 could produce the optimum. However selection and drift, specially if the problem was deceptive could by then significantly lower the number of those genomes, possibly removing them at all from the population. We see that the optimum might never be reached.

In general the Robbins equilibrium requires that the percentage of a given combination of any two genes is the following

$$P'_{xy} = P_{x*}P_{*y} = (P_{xy} + P_{x\bar{y}})(P_{xy} + P_{\bar{x}y}),$$

where by  $\bar{x}$  I denote the other possible allele (or  $1 - x$ , if the possible alleles are 0 and 1). Generalizing the derivation for the uniform recombination, we have

$$\begin{aligned}
P'_{xy} &= \frac{P_{\bar{x}\bar{y}}P_{xy} + P_{xy}P_{\bar{x}\bar{y}} + P_{\bar{x}y}P_{x\bar{y}} + P_{x\bar{y}}P_{\bar{x}y} +}{4} + \\
&+ \frac{P_{\bar{x}y}P_{xy} + P_{xy}P_{\bar{x}y} + P_{x\bar{y}}P_{xy} + P_{xy}P_{x\bar{y}}}{2} + P_{xy}P_{xy} = \\
&\frac{(P_{xy} + P_{x\bar{y}})(P_{xy} + P_{\bar{x}y}) + (P_{\bar{x}\bar{y}} + P_{x\bar{y}} + P_{\bar{x}y} + P_{xy})P_{xy}}{2} = \\
&\frac{P_{x*}P_{*y} + P_{xy}}{2}.
\end{aligned}$$

We see that in fact uniform crossover brings the population only half way towards the Robbins equilibrium. This observation correspond to the fact that an average expected length of a block that a uniform recombination copies from a parent is of length close to 2.<sup>3</sup> If we wanted to achieve a “perfectly” mixing recombination, it would have to copy blocks of average length of one gene, which means that for each two genes it would have to ensure copying from a different parent, which is obviously impossible for lengths bigger than two.

We can further generalize the equation for any recombination for which the probability of copying any two genes together is  $s$ . For uniform recombination,  $s = \frac{1}{2}$  and for a non-existing perfectly mixing recombination, we would have  $s = 0$ . Assuming that each parent can still be chosen for the first gene with probability  $\frac{1}{2}$ , the appropriate derivation is

---

<sup>3</sup>The expected block length is  $\sum_{i=1}^L \frac{i}{2^i} = 2 - \frac{2+L}{2^L} \approx 2$  for large  $L$ .

$$\begin{aligned}
P'_{xy} &= \frac{s}{2}(P_{\bar{x}\bar{y}}P_{xy} + P_{xy}P_{\bar{x}\bar{y}}) + \frac{(1-s)}{2}(P_{\bar{x}y}P_{x\bar{y}} + P_{x\bar{y}}P_{\bar{x}y}) + \\
&+ \frac{(s+(1-s))}{2}(P_{\bar{x}y}P_{xy} + P_{xy}P_{\bar{x}y} + P_{x\bar{y}}P_{xy} + P_{xy}P_{x\bar{y}}) + P_{xy}P_{xy} = \\
&(1-s)(P_{xy} + P_{x\bar{y}})(P_{xy} + P_{\bar{x}y}) + s(P_{\bar{x}\bar{y}} + P_{x\bar{y}} + P_{\bar{x}y} + P_{xy})P_{xy} = \\
&(1-s)P_{x*}P_{*y} + sP_{xy}.
\end{aligned}$$

We see that when the probability of recombination copying two alleles together is bigger, the percentages of individuals change slower from generation to generation. In particular for the case with no recombination, when alleles are never split, and the selection effectively operates on whole genomes,  $s = 1$  and  $P'_{xy} = P_{xy}$ . Of course, for more complex recombination operators the probability of copying two alleles depends on their position in genomes and therefore more complex models are needed. If we however wanted to compute an average  $s$  for e.g. one-point or two-point recombination (for  $L \geq 3$ ) we would get  $s > \frac{1}{2}$ .

There exist recombination operators mixing alleles faster than uniform crossover, which I will now quickly show. Their use for EAs is limited, because we usually want to maintain some blocks of alleles, but nevertheless they are interesting from a theoretical point of view. Deterministic recombination operators, i.e. such that use certain masks.<sup>4</sup> in certain order can achieve  $s < \frac{1}{2}$ . An example can be a single mask 10 used repetitively for genomes of length two, which is a perfect recombination

---

<sup>4</sup>One can describe a crossover using a mask consisting of '1's and '0's, denoting whether an allele should be copied from one parent or another.

operator ( $s = 0$ ),<sup>5</sup> or masks 110, 101 and 011 for genomes of length three, for which  $s = \frac{1}{3}$ . For longer genomes we can imagine crossover  $X$  that repeats the following set of masks. If  $L = 2k$  for some integer  $k$  (that is  $L$  is even), then we take all possible subsets of size  $k$  out of  $L$  genes, and each such subset defines a mask (it will have 1 for the members of the set and 0 otherwise). There is  $\binom{2k}{k}$  of such masks. For any two genes, masks that copy their values from the same parent will be those that have either two 1s, or two 0s in appropriate places. There is  $\binom{2k-2}{k-2} + \binom{2k-2}{k}$  of such masks. Therefore we have

$$s_X = \frac{\binom{2k-2}{k-2} + \binom{2k-2}{k}}{\binom{2k}{k}} = \frac{2 \frac{(2k-2)!}{(k-2)!k!}}{\frac{(2k)!}{k!k!}} = 2 \frac{k(k-1)}{2k(2k-1)} = \frac{k-1}{2k-1}.$$

If  $L = 2k + 1$  for some integer  $k$  ( $L$  is odd), we also take all possible subsets of genes of size  $k$ . In such case we have

$$s_X = \frac{\binom{2k-1}{k-2} + \binom{2k-1}{k}}{\binom{2k+1}{k}} = \frac{\frac{(2k-1)!}{(k-2)!(k+1)!} + \frac{(2k-1)!}{k!(k-1)!}}{\frac{(2k+1)!}{k!(k+1)!}} = \frac{(2k-1)!}{(2k+1)!} \left( \frac{k!(k+1)!}{(k-2)!(k+1)!} + \frac{k!(k+1)!}{k!(k-1)!} \right) = \frac{k(k-1) + k(k+1)}{2k(2k+1)} = \frac{k}{2k+1}.$$

In general we have  $s = E \left[ \left( \binom{x}{2} + \binom{L-x}{2} \right) \binom{L}{2}^{-1} \right]$ , where  $x$  denotes the number of '0's (or '1's) in a mask. We can expand the formula

---

<sup>5</sup>In fact, this is the exact reason why the Hardy-Weinberg principle (describing percentages of heterozygotes and homozygotes) holds in genetics, because for a diploid genome two alleles of a certain locus are guaranteed to come from two parents.



$$\left( \binom{x}{2} + \binom{L-x}{2} \right) \binom{L}{2}^{-1} = \frac{x(x-1) + (L-x)(L-x-1)}{L(L-1)} = 1 - \frac{2x(L-x)}{L(L-1)}.$$

We know that  $x(L-x) \leq (\frac{L}{2})^2$  and thus we can bound  $s$  from below. If  $L = 2k$  for some integer  $k$ , then

$$s \geq 1 - \frac{2k^2}{2k(2k-1)} = \frac{k-1}{2k-1}.$$

If  $L = 2k + 1$  for some integer  $k$ , then

$$s \geq 1 - \frac{2k(k+1)}{(2k+1)2k} = \frac{k}{2k+1}.$$

We see that the deterministic crossover described before is therefore optimal in the sense of mixing alleles and bringing a population towards the Robbins equilibrium as fast as possible. We also see that with longer genomes  $s \approx \frac{1}{2}$ , so even the optimal crossover is not perfect.

## Appendix E: Convergence to migration equilibrium

In this appendix I give a proof for the fact that bidirectional migrations and infinite populations lead islands to an equilibrium point, defined as a point in which each island has the same number of representative genes in any other island. While this may be quite intuitive, a formal proof confirms our predictions.

Let us denote by  $c_{i,j}(t)$  a percentage of representative individuals from island  $i$  in island  $j$  at time  $t$ . In each migration a percentage  $\alpha > 0$  of individuals in a given island is exchanged for individuals from each other island ( $\alpha$  being dependent on the size of migration). Assuming migrants are chosen randomly, for each  $i$  and  $j$  we have

$$c_{i,j}(t+1) = (1 - (N-1)\alpha)c_{i,j}(t) + \sum_{k \neq j} \alpha c_{i,k}(t).$$

If we denote by  $C(t)$  the matrix of  $\{c_{i,j}(t)\}_{i,j}$ , the above equations may be written in a matrix form

$$C(t+1) = C(t)A,$$

where  $A$  is an  $N$  by  $N$  matrix, independent of  $t$ , with  $a_{11} = 1 - (N-1)\alpha$  on diagonal, and  $a_{1j} = \alpha$  everywhere else. Of course, we have

$$C(t) = C(0)A^t.$$

It is easy to show that  $C(t)$  converges to  $\{\frac{1}{N}\}_{i,j}$ , that is to a matrix with  $\frac{1}{N}$  in every position. Because  $C(0)$  is an identity matrix, we need to show that  $A^t$  converges to  $\{\frac{1}{N}\}_{i,j}$ .

Elements of  $A$  have two possible values,  $a_1$  and  $b_1$ . If we assume that  $A^t$  has elements of only two possible values,  $a_t$  on the diagonal, and  $b_t$  everywhere else, then  $A^{t+1}$  also consists of only two values, because for elements on a diagonal we will have

$$a_{t+1} = a_t^2 + (N - 1)b_t^2,$$

and for other elements

$$b_{t+1} = 2a_t b_t + (N - 2)b_t^2.$$

Therefore, from induction, we know that for each  $t$  the matrix  $A^t$  consist of only two values and we can construct two series  $\{a_t\}$  and  $\{b_t\}$  denoting them.

Subtracting the second equation from the first, we have

$$a_{t+1} - b_{t+1} = (a_t - b_t)^2 = \dots = (a_1 - b_1)^{t+1} = (1 - N\alpha)^{t+1}.$$

Because  $\alpha > 0$  and for simplicity assuming  $\alpha \leq \frac{1}{N}$ .<sup>1</sup> we know that

$$\lim_{t \rightarrow \infty} (a_t - b_t) = 0.$$

It is also easy to show by induction that the following equation always holds:

---

<sup>1</sup>In fact it is enough to assume  $\alpha \leq \frac{2}{N}$ . On the other hand if we have  $\alpha$  individuals immigrating from each of  $N - 1$  islands, it is undefined what the contents of the island is for  $\alpha > \frac{1}{N-1}$  since the immigrants would have to overwrite each other. One of the solutions is to assume that in such case migrations are equally important and  $\alpha = \frac{1}{N-1}$  is the actual immigration level.

$$a_t + (N - 1)b_t = 1.$$

The equation holds for  $a_1$  and  $b_1$ , and also we have

$$a_{t+1} + (N - 1)b_{t+1} = a_t^2 + (N - 1)b_t^2 + 2(N - 1)a_t b_t + (N - 1)(N - 2)b_t^2 = (a_t + (N - 1)b_t)^2 = 1.$$

Substituting  $a_t$  for  $1 - (N - 1)b_t$  we have

$$a_t - b_t = 1 - (N - 1)b_t - b_t = 1 - Nb_t,$$

and thus

$$\lim_{t \rightarrow \infty} b_t = \frac{1}{N}.$$

Therefore we know that the limit for  $\{a_n\}$  exists too, and

$$\lim_{t \rightarrow \infty} a_t = \frac{1}{N}.$$

## Appendix F: Notes on experimenting with island models

This appendix is a collection of notes on designing, implementing and running experiments with IMs. While it may be very subjective, I hope it is useful for future Ph.D. candidates.

### Designing IMs

It is important to realize that implementing an island model with migrations requires many decisions with regard to the model, which are sometimes difficult to make.

Immigrants may be inserted into the target population in one of three possible phases of evolution. They may substitute individuals in the target population (or be added, increasing the size of the population) before selecting individuals for recombination. Alternatively, they may be added to the group of selected “parents” before applying the reproduction operators. The third option would be to add them to the offspring pool after the reproduction operators, but before the survival selection.<sup>1</sup>

I used the first option.

The order of performing particular migrations between islands does matter. In the case of asynchronous model, the effects would probably be less notable due to the stochasticity and thus a different order of migrations each time. In the case of a

---

<sup>1</sup>When an EA use only one of survival and parent selection, this option is equivalent to the first option.

synchronized model, it is important to make proper design decisions. An example of such situation is a situation with two big migrations into the same target population. In such case the individuals from the later migration would overwrite the individuals from the earlier one. To avoid such situation, one can create an immigration pool to which all the potential immigrants are inserted. After all migrations into a given population are analyzed, immigrants from the immigration pool are all treated equally, and if there are more of them than the number of individuals in the target population, a random subset of them may be chosen. Such solution may cause a temporary significant growth of the immigration pool in case of many migrations.

The emigration process is usually done by copying individuals, and in such case the order does not matter. However, if one chose to remove individuals from the source population, then the order matters again. It would be advisable to choose individuals for all migrations originating in a given population in some rotating mode.

Another solution to both immigration and emigration order issues would be to randomize the order of migrations each time. However, although it reduces the effect of populations being treated differently, it may not reduce the effect entirely. In the example with two big migrations with the same target population, we would still be overwriting immigrants from the earlier with the immigrants from another.

When exchanging migrants in both directions, rather than just moving them one way, a problem occurs with choosing emigrants. If the same emigrant is chosen twice and two exchanges take place, the individual received in the first exchange will end up being sent in the second exchange instead of the original individual. Therefore one should avoid choosing the same individual twice for exchange.

To achieve a maximally fair treatment of exchange migrations the following

procedure can be used. All source populations send migrants to the target populations. Migrants are removed from the source population, so there is no possibility of choosing the same individual twice. The migrants are collected in immigration pools of the target populations. In the next stage, all the migrations are processed. Migrants (possibly not all) are chosen from the immigration pools and paired with individuals in the target population, put in their place, and the corresponding individuals are sent back to the source populations. Migrants from immigration pools that are not chosen to immigrate, are also sent back. This way no individual can participate in more than one exchange, which situation could result in moving this individual between two populations not joined by a migration. A potential problem with this approach is that once we move migrants from a given population, they cannot be chosen for exchange with immigrating migrants any more.

Another approach is to simply allow exchange of individuals in totally random order, hoping that with relatively small migrations, possible cases of individuals participating in more than one exchange do not affect the general outcome too much.

## **Randomness and determinism in EAs**

Issues related to the determinism in IMs and EAs have been commented in the past by others, and the guidelines are “theoretically” well known. Nevertheless, it is a common practice to disregard these suggestions. I feel it does not harm to repeat them, as it seems to me that they need being stressed.

It may be tempting to think that “the more randomness we insert into the algorithm, the bigger chances we have for finding good solutions,” because “somehow” the algorithm will search all over the search space and return the optimum, because

“EA is a stochastic search.” This is a *naïve* understanding, if not to say a *misunderstanding* of EAs. The more random choices there are, the bigger the chance that we lose some alleles or building blocks or partial solutions. Performing various experiments, I have observed quite a big influence of randomness on faster decrease of diversity, which can preclude the algorithm from finding better results.

Some choices regarding determinism might require a potentially significant changes to the algorithm (e.g. by using bidirectional migrations). However, other choices do not change the general shape of the algorithm. My recommendation would be to consider the following suggestions:

- select recombination parents in a deterministic way
  - if selecting randomly, make sure they do not repeat.
  - if selecting by a tournament, select tournament groups deterministically.
- use two-children recombination
- use bidirectional migrations
- choose emigrants deterministically (in particular using random policy, make sure you do not choose the same individual twice)

Generally, one should try to design as much deterministic algorithm as possible, and only permute the order of elements (e.g. which parent recombines with which).

## Source code

In the process of writing this dissertation, I have used several implementations of evolutionary algorithm. In particular two of them are notable. One is *Inventor*, a



GUI implementation based on *ECKit* engine by Mitchell Potter and designed with engineering domains in mind. Although it was very flexible and I performed multiple experiments with this software, toward the end of my work I realized that the overhead in this system is too big, specially that I could hardwire certain parameters and remove some features. Therefore, I wrote a simple program, with a core of a few hundred lines, in a single file, which could achieve exactly the same results, but the computing time was in terms of seconds comparing to minutes if not hours before.<sup>2</sup> Of course, with time, this second implementation started to grow in size, so it is maybe time to abandon it. . .

More experiments I ran, more I realized how easy it is to influence the results with small programming nuances. Not only must the model correspond to the real system, but also the implementation must precisely follow the model. This is especially tricky when dealing with stochastic behavior and introducing any ordering of events (thus, each loop is a potential source of mistake) and in the past I have not been saved from running multiple experiments using a flawed code. I do hope that the final implementation I wrote corresponds to the model I have in mind, and to ensure it, I have spent much time thinking about every aspect.

I tried to optimize the code as much as it could significantly speed up computations. A few more optimizations were definitely possible, but they would speed up the computations hardly any more, and would greatly obfuscate the code.

---

<sup>2</sup>For example, I simplified each gene to be just a number in an array, rather than being an instance of a class inheriting from a hierarchy of more general classes, with polymorphic methods, checking the legality of each new allele through a chain of nested methods. . .

## Running experiments

During the process of writing the dissertation I have learnt quite a few lessons about running experiments. Most of them I learnt hard way! Therefore, here I gather some informal remarks, which may occur useful for others (e.g., prospective Ph.D. students, and especially those who would prefer to graduate faster than I did).

- Plan first, away from your computer. Only when you are able to predict the likely results of an experiment, implement and run it to (hopefully) confirm your prediction. Doing research bottom-up, that is running a full range of experiments to analyze them later takes long time and usually has little benefit. If you want to get an insight you do not need to know the result for each possible argument value - just choose a few (maybe two) values.
- Invest time in creating scripts, or utilities. You will always find you have to re-run experiments. This also applies to recreating charts.
- In spite of above, when creating charts, make sure they look decent even in a draft (i.e. include axis labels, use proper axis values etc.). Even if you have proper scripts (somewhere...) recreating charts months later just to add labels will take much longer.
- You will keep adding features to your software after you have run previous experiments. Since you cannot predict all possible extensions, I do not have a good advice how to keep your results in a consistent format. Try to predict as much as possible, and store all the parameters, even if you think they are

constant. You may decide to play with them later. Optimally have a clear plan of experiments...

- Keep old names for the data files, scripts etc... Do not rename them to your newest fancy way of naming, because you will want to re-name them again anyway. Keeping the old names at least prevents from having multiple names for the same thing.
- I have extensively used shell scripts to analyze data, which is a rather bad idea. Do not try to write complex “statistical package” using shell scripts. Both obeying all the syntactic rules and debugging scripts is a nightmare. It is probably better to use your favorite programming language (or a specialized tool, like *R*). *Gnuplot* is okay for plotting charts, but only after you have learnt it well. You can put commands in a file and run it from a script. I found *R* too sophisticated for my purposes, but I know people who praised it a lot.
- Do not make each piece of your software as flexible and general as possible. You will spend hours considering a theoretical situation that never happens. Instead, accept some “hacking” and inconsistencies in your software.
- A GUI part of your software may be useful (I have realized the existence of some bugs, due to inconsistencies in charts), but generally it will be a burden. If you want to keep it synchronized with what happens inside the software, you will spend hours fighting with the layouts, updating graphical representations, mouse click events etc. Try to avoid it.

- Remember that when you want to run a large number of experiments this will require *significantly* more effort. Not only they will run longer, but you will encounter *a lot* of unexpected troubles. You will have to:
  - have a method to distribute computations and start them easily.
  - access your files remotely (I used *vi* to edit files on remote servers).
  - gather data automatically.
  - convert data automatically.
  - analyze data automatically.
  - hard drive space will be an issue, to keep the results (so often you will have to process them online).
  - computer memory may be an issue (to store large arrays of data).

You will have to work around each of these problems, writing proper scripts, redesigning your software, sending and storing files automatically etc. As already mentioned, you *will* have bugs in the system, which you may discover even after you have already analyzed the data (by the way, do not ignore small signs of misbehavior). Instead of discarding the faulty results entirely, try to consider, if maybe what you did is not another approach to the problem, or if it is not at least partially valid.

## Dissertation online

This dissertation, as well as any updates and other related files are available online at <http://cs.gmu.edu/~zskolick/dissertation>. All the online materials are provided “as is”, and I may remove them at any time.

## Bibliography

## Bibliography

- [Alba and Troya, 1999] Enrique Alba and Jose M. Troya. An analysis of synchronous and asynchronous parallel distributed genetic algorithms with structured and panmictic islands. In *IPPS/SPDP Workshops*, pages 248–256, 1999.
- [Alleaume-Benharira *et al.*, 2006] M. Alleaume-Benharira, I. R. Pen, and O. Ronce. Geographical patterns of adaptation within a species’ range: interactions between drift and gene flow. *Journal of Evolutionary Biology*, 19:203–215, 2006.
- [Arciszewski and De Jong, 2001] T. Arciszewski and K. A. De Jong. *Evolutionary Computation in Civil Engineering: Research Frontiers*, pages 161–184. Saxe-Coburg Publications, 2001.
- [Baker, 1987] J. Baker. Reducing bias and inefficiency in the selection algorithm. In J. J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21. Lawrence Erlbaum Associates, 1987.
- [Barabási, 2002] Albert-László Barabási. *Linked: The New Science of Networks*. Perseus Publishing, 2002.
- [Bentley, 1999] P. Bentley, editor. *Evolutionary Design by Computers*. Morgan Kaufmann, San Francisco, 1999.
- [Bessaou *et al.*, 2000] M. Bessaou, A. Pétrowski, and P. Siarry. Island model cooperating with speciation for multimodal optimization. In Hans-Paul Schwefel Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, editor, *Parallel Problem Solving from Nature — PPSN VI 6th International Conference*, Paris, France, 2000. Springer Verlag.
- [Blickle and Thiele, 1996] Tobias Blickle and Lothar Thiele. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, 4(4):361–394, 1996.
- [Branke *et al.*, 2004] Juergen Branke, Andreas Kamper, and Hartmut Schneck. Distribution of evolutionary algorithms in heterogeneous networks. In *Proceedings of Genetic and Evolutionary Computation Conference — GECCO 2004*. Springer-Verlag, 2004.

- [Cantú-Paz and Goldberg, 2003] Erick Cantú-Paz and David E. Goldberg. Are multiple runs of genetic algorithms better than one? In Erick Cantú-Paz et. al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference 2003*, 2003.
- [Cantú-Paz, 1999] Erick Cantú-Paz. *Designing Efficient and Accurate Parallel Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, 1999.
- [Cantú-Paz, 2001] Erick Cantú-Paz. Migration policies, selection pressure, and parallel evolutionary algorithms. *Journal of Heuristics*, 7(4):311–334, 2001.
- [Cobb, 1990] H. G. Cobb. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical Report AIC-90-001, Naval Research Laboratory, Washington, DC, 1990.
- [Cox and Durrett, 2002] Ted Cox and Rick Durrett. The stepping stone model: New formulas expose old myths. *Annals of Applied Probability*, 12:1348–1377, 2002.
- [Darwin, 1859] Charles Darwin. *On the origin of species*. John Murray, 1859.
- [Davis, 1989] Lawrence Davis. Adapting operator probabilities in genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications*, pages 61–69, San Mateo, California, 1989. Morgan Kaufman.
- [De Jong *et al.*, 1995] Kenneth A. De Jong, William M. Spears, and Diana F. Gordon. Using markov chains to analyze GAFOs. In L. Darrell Whitley and Michael D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 115–137. Morgan Kaufmann, San Francisco, CA, 1995.
- [De Jong, 1975] Kenneth A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, Ann Arbor, 1975.
- [De Jong, 2006] Kenneth A. De Jong. *Evolutionary Computation: A Unified Approach*. MIT Press, Cambridge, MA, 2006.
- [Deb and Goldberg, 1991] K. Deb and D. Goldberg. A comparative analysis of selection schemes used in genetic algorithms. In G. J. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69–93, San Mateo, CA, 1991. Morgan Kaufman.
- [Deb and Goldberg, 1993] K. Deb and D.E. Goldberg. Analyzing deception in trap functions. In D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 93–108. Morgan Kaufmann, 1993.
- [Deb and Spears, 1997] Kalyanmoy Deb and William M. Spears. Speciation methods. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of*

*Evolutionary Computation*, pages C6.2:1–5. Institute of Physics Publishing and Oxford University Press, Bristol, New York, 1997.

- [Deb, 2001] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001.
- [Dezinger and Kidney, 2003] Jörg Dezinger and Jordan Kidney. Improving migration by diversity. In *Proceedings of the 2003 Congress on Evolutionary Computation*, pages 700–707. IEEE Press, 2003.
- [Droste *et al.*, 2002] Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276:51–81, 2002.
- [Eby *et al.*, 1999] D. Eby, R. Averill, E. Goodman, and W. Punch. The optimization of flywheels using an injection island genetic algorithm. In P. Bentley, editor, *Evolutionary Design by Computers*, pages 167–190. Morgan Kaufmann, San Francisco, 1999.
- [Eshelman, 1991] L. Eshelman. The CHC adaptive search algorithm. In G.J.E. Rawlins, editor, *Foundations of Genetic Algorithms I*, pages 265–283, San Mateo CA, 1991.
- [Fernández de Vega *et al.*, 2000] Francisco Fernández de Vega, Marco Tomassini, III William F. Punch, and Juan Manuel Sánchez-Pérez. Experimental study of multipopulation parallel genetic programming. In *Proceedings of the European Conference on Genetic Programming*, pages 283–293, London, UK, 2000. Springer-Verlag.
- [Fogel *et al.*, 1966] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, New York, 1966.
- [Forrest and Jones, 1994] Stephanie Forrest and Terry Jones. Modeling complex adaptive systems with Echo. In R. J. Stonier and X. H. Yu, editors, *Complex Systems: Mechanisms of Adaptation*. IOS Press, 1994.
- [Freisleben and Merz, 1996] Bernd Freisleben and Peter Merz. New genetic local search operators for the traveling salesman problem. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Proceedings of the Fourth Conference on Parallel Problem Solving from Nature (PPSN IV)*, volume 1141, pages 890–899, Berlin, 1996. Springer.
- [Galeano *et al.*, 2002] G. Galeano, F. Fernandez, M. Tomassini, and L. Vanneschi. Studying the influence of synchronous and asynchronous parallel GP on programs’ length evolution. In *Proceedings of 2002 Congress On Evolutionary Computation*, pages 1727–1732, Piscataway, NJ, 2002. IEEE Press.



- [Geiringer, 1944] H. Geiringer. On the probability theory of linkage in mendelian heredity. *Annals of Mathematical Statistics*, 15(1):25–27, 1944.
- [Gen and Cheng, 2000] Mitsuo Gen and Runwei Cheng. *Genetic Algorithms and Engineering Optimization*. Wiley, 2000.
- [Goldberg and Richardson, 1987] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette, editor, *Proceeding of the 2nd Int. Conference on Genetic Algorithms*, pages 41–49, Hillsdale, NJ, 1987. Lawrence Erlbaum.
- [Goldberg *et al.*, 1989] D.E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5):493–530, 1989.
- [Goldberg, 1989] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Massachusetts, 1989.
- [Goldberg, 2002] D.E. Goldberg. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Boston, MA, 2002.
- [Gustafson, 2004] Steven M. Gustafson. *An Analysis of Diversity in Genetic Programming*. PhD thesis, The University of Nottingham, 2004.
- [Harik and Goldberg, 1997] Georges R. Harik and David E. Goldberg. Learning linkage. In Richard K. Belew and Michael D. Vose, editors, *Foundations of Genetic Algorithms 4*, pages 247–262, San Francisco, CA, 1997. Morgan Kaufmann.
- [Holland, 1962] J. H. Holland. Outline for a logical theory of adaptive systems. *Journal of the ACM*, 9(3):297–314, July 1962.
- [Holland, 1975] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, 1975.
- [Hu *et al.*, 2005] Jianjun Hu, Erik Goodman, Kisung Seo, Zhun Fan, and Rondal Rosenberg. The Hierarchical Fair Competition (HFC) framework for sustainable evolutionary algorithms. *Evolutionary Computation*, 13(2):241–277, 2005.
- [Kicinger *et al.*, 2005] Rafal Kicinger, Tomasz Arciszewski, and Kenneth De Jong. Evolutionary computation and structural design: a survey of the state of the art. *Computers & Structures*, 83(23–24):1943–1978, 2005.
- [Klug and Cummings, 1997] William S. Klug and Michael R. Cummings. *Concepts of genetics*. Prentice Hall, Upper Saddle River, NJ 07458, fifth edition, 1997.
- [Koza, 1992] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

- [Krink *et al.*, 1999] Thiemo Krink, Brian H. Mayoh, and Zbigniew Michalewicz. A PATCHWORK model for evolutionary algorithms with structured and variable size populations. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1321–1328, Orlando, Florida, USA, 1999. Morgan Kaufmann.
- [Krink *et al.*, 2000] T. Krink, P. Rickers, and R. Thomsen. Applying self-organised criticality to evolutionary algorithms. In *Proceedings of the 6th Parallel Problem Solving from Nature — PPSN VI*, pages 375–384, 2000.
- [Land and Belew, 1995] M. Land and R.K. Belew. No perfect two-state cellular automata for density classification exists. *Physical Review Letters*, 74:1548–1550, 1995.
- [Luke and Panait, 2002] Sean Luke and Liviu Panait. Fighting bloat with nonparametric parsimony pressure. In Juan Julian Merelo Guervos et al, editor, *Parallel Problem Solving from Nature — PPSN VII, LNCS 2439*, pages 411–421. Springer Verlag, 2002.
- [Mahfoud, 1992] S. W. Mahfoud. Crowding and preselection revisited. In R. Männer and B. Manderick, editors, *Proceedings of Parallel Problem Solving from Nature 2*, pages 27–36, New York, NY, 1992. Elsevier Science B. V.
- [Mathias and Whitley, 1992] K. Mathias and D. Whitley. Genetic operators, the fitness landscape and the traveling salesman problem. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 219–228. Elsevier, 1992.
- [Mitchell *et al.*, 1992] M. Mitchell, S. Forrest, and J. H. Holland. The royal road function for genetic algorithms: Fitness landscapes and GA performance. In F. J. Varela and P. Bourguine, editors, *Proceedings of the First European Conference on Artificial Life*, pages 245–254, Cambridge, MA, 1992. MIT Press.
- [Mitchell *et al.*, 2000] George G. Mitchell, Diarmuid O’Donoghue, and Adrian Trenaman. A new operator for efficient evolutionary solutions to the travelling salesman problem. In *Proc. Applied Informatics*, pages 771–774, Innsbruck, Austria, 2000.
- [Morrison, 2002] Ronald W. Morrison. *Designing Evolutionary Algorithms for Dynamic Environments*. PhD thesis, George Mason University, 2002.
- [Mühlenbein and Schlierkamp-Voosen, 1993] H. Mühlenbein and D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithms: continuous parameter optimization. *Evolutionary Computation*, 1(1):25–49, 1993.

- [Oler, 2004] Walt Oler. ME 1315 class notes: Engineering design. [http://www.me.ttu.edu/class-websites/ME1315/Notes/What is Engineering Design.pdf](http://www.me.ttu.edu/class-websites/ME1315/Notes/What%20is%20Engineering%20Design.pdf), 2004. accessed 23th February 2004.
- [Oppacher and Wineberg, 1999] F. Oppacher and M. Wineberg. The shifting balance genetic algorithm: Improving the GA in a dynamic environment. In Banzhaf et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 504–510, 1999.
- [Pagie and Mitchell, 2002] L. Pagie and M. Mitchell. A comparison of evolutionary and coevolutionary search. *International Journal of Computational Intelligence and Applications*, 2(1):53–69, 2002.
- [Pahl and Beitz, 1996] G. Pahl and W. Beitz. *Engineering design: a systematic approach*. Springer-Verlag, New York, 1996.
- [Poli and Langdon, 1998] Riccardo Poli and William B. Langdon. Schema theory for genetic programming with one-point crossover and point mutation. *Evolutionary Computation*, 6(3):231–252, 1998.
- [Popovici and De Jong, 2003] Elena Popovici and Kenneth De Jong. Understanding EA dynamics via population fitness distributions. In Erick Cantú-Paz et. al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference 2003*, pages 1604–1605, 2003.
- [Popovici, 2006] Elena Popovici. *An Analysis of Two-Population Coevolutionary Computation*. PhD thesis, George Mason University, 2006.
- [Potter, 1997] Mitchell A. Potter. *The Design and Analysis of a Computational Model of Cooperative Coevolution*. PhD thesis, George Mason University, 1997.
- [Rechenberg, 1965] I. Rechenberg. Cybernetic solution path of an experimental problem. Library translation no. 1122, Royal Aircraft Establishment, Farnborough, England, 1965.
- [Rogers and Pruegel-Bennett, 1999] A. Rogers and A. Pruegel-Bennett. Genetic drift in genetic algorithm selection schemes. *IEEE-EC*, 3(4):298, November 1999.
- [Ronnwinkler and Martinez, 2001] Christopher Ronnwinkler and Thomas Martinez. Explicit speciation with few a priori parameters for dynamic optimization problems. In Jürgen Branke and Thomas Bäck, editors, *Evolutionary Algorithms for Dynamic Optimization Problems*, pages 31–38, San Francisco, California, USA, 7 2001.
- [Rowe et al., 2004] J. Rowe, D. Whitley, L. Barbulescu, and J.-P. Watson. Properties of gray and binary representations. *Evolutionary Computation*, 12(1):47–76, 2004.

- [Rudolph, 1998] Günter Rudolph. Finite Markov chain results in evolutionary computation: A tour d’horizon. *Fundamenta Informaticae*, 35(1–4):67–89, 1998.
- [Rudolph, 2000a] Günter Rudolph. On takeover times in spatially structured populations: Array and ring. In K. K. Lai, O. Katai, M. Gen, and B. Lin, editors, *Proceedings of the Second Asia-Pacific Conference on Genetic Algorithms and Applications (APGA ’00)*, pages 144–151, Hong Kong, PR China, 2000. Global-Link Publishing Company.
- [Rudolph, 2000b] Günter Rudolph. Takeover times and probabilities of non-generational selection rules. In Darrell Whitley, David Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 903–910, Las Vegas, Nevada, USA, 2000. Morgan Kaufmann.
- [Sarma, 1998] Jayshree Sarma. *An Analysis of Decentralized and Spatially Distributed Genetic Algorithms*. PhD thesis, George Mason University, Fairfax, VA, 1998.
- [Sastry and Goldberg, 2002] Kumara Sastry and David E. Goldberg. How well does a single-point crossover mix building blocks with tight linkage? Technical Report IlliGAL Report No. 2002013, University of Illinois at Urbana-Champaign, May 2002.
- [Shang and Qiu, Spring 2006] Yun-Wei Shang and Yu-Huang Qiu. A note on the extended rosenbrock function. *Evolutionary Computation*, 14(1):119–126, Spring 2006.
- [Shapiro *et al.*, 1994] Jonathan Shapiro, Adam Prugel-Bennett, and Magnus Rattray. A statistical mechanical formulation of the dynamics of genetic algorithms. In *Evolutionary Computing, AISB Workshop*, pages 17–27, 1994.
- [Skolicki and De Jong, 2004] Zbigniew Skolicki and Kenneth De Jong. Improving evolutionary algorithms with multi-representation island models. In *Parallel Problem Solving from Nature — PPSN VIII 8th International Conference*. Springer-Verlag, 2004.
- [Skolicki and De Jong, 2005] Zbigniew Skolicki and Kenneth De Jong. The influence of migration sizes and intervals on island models. In *Proceedings of Genetic and Evolutionary Computation Conference — GECCO 2005*. ACM, 2005.
- [Spears, 1992] William Spears. Crossover or mutation? In Darrell Whitley, editor, *Foundations of Genetic Algorithms 2*, San Mateo, CA, 1992. Morgan Kaufmann.
- [Spears, 1994] William Spears. Simple subpopulation schemes. In *Proceedings of the Fourth Annual Conference on Evolutionary Programming (EP94)*, pages 296–307. World Scientific, 1994.

- [Spears, 1995] William M. Spears. Adapting crossover in evolutionary algorithms. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Proc. of the Fourth Annual Conference on Evolutionary Programming*, pages 367–384, Cambridge, MA, 1995. MIT Press.
- [Spears, 1998] William M. Spears. *The Role of Mutation and Recombination in Evolutionary Algorithms*. PhD thesis, George Mason University, Fairfax, Virginia, 1998.
- [Sprave, 1999] Joachim Sprave. A unified model of non-panmictic population structures in evolutionary algorithms. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1384–1391, Mayflower Hotel, Washington D.C., USA, 1999. IEEE Press.
- [Stewart, 1997] John Stewart. The evolution of genetic cognition. *Journal of Social and Evolutionary Systems*, 20:53–73, 1997.
- [Toussaint and Igel, 2002] Marc Toussaint and Christian Igel. Neutrality: A necessity for self-adaptation. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, pages 1354–1359, 2002.
- [Toussaint, 2001] Marc Toussaint. Self-adaptive exploration in evolutionary search. Technical Report IRINI-2001-05, Institute for Neuroinformatics, Ruhr-University Bochum, 2001.
- [Tsutsui and Fujimoto, 1993] S. Tsutsui and Y. Fujimoto. Forking genetic algorithm with blocking and shrinking modes (FGA). In Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 206–213, 1993.
- [Tsutsui *et al.*, 1997] S. Tsutsui, Y. Fujimoto, and A. Ghosh. Forking genetic algorithms: GAs with search space division schemes. *Evolutionary Computation*, 5:61–80, 1997.
- [Ursem, 1999] Rasmus K. Ursem. Multinational evolutionary algorithms. In Peter J. Angeline, Zbigniew Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 3, pages 1633–1640, Mayflower Hotel, Washington D.C., USA, 1999. IEEE Press.
- [Ursem, 2000] Rasmus K. Ursem. Multinational GAs: Multimodal optimization techniques in dynamic environments. In Darrell Whitley, David Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 19–26, Las Vegas, Nevada, USA, 2000. Morgan Kaufmann.

- [Ursem, 2002] Rasmus K. Ursem. Diversity-guided evolutionary algorithms. In *Proceedings of Parallel Problem Solving from Nature VII (PPSN-2002)*, pages 462–471. Springer Verlag, 2002.
- [Verel *et al.*, 2006] Sebastien Verel, Philippe Collard, Marco Tomassini, and Leonardo Vanneschi. Neutral fitness landscape in the cellular automata majority problem. In *Proceedings of the 7th International Conference on cellular automata for research & industry, LNCS 4173*, pages 258–267, Perpignan, France, September 2006.
- [Vose, 1999] M. D. Vose. *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, Cambridge, MA, 1999.
- [Watson, 2006] Richard A. Watson. *Compositional Evolution: The impact of Sex, Symbiosis and Modularity on the Gradualist Framework of Evolution*. Vienna Series in Theoretical Biology. MIT Press, Cambridge, MA, 2006.
- [Watts, 2003] Duncan J. Watts. *Six Degrees: The Science of a Connected Age*. W.W. Norton & Company, 2003.
- [Whitley *et al.*, 1999] D. Whitley, S. Rana, and R. B. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 7(1):33–47, 1999.
- [Wikibooks, 2007] Wikibooks. Cellular automata/counting preimages. [http://en.wikibooks.org/wiki/Cellular\\_Automata/Counting\\_Preimages](http://en.wikibooks.org/wiki/Cellular_Automata/Counting_Preimages), 2007. accessed 18th August 2007.
- [Wolfram, 2002] Stephen Wolfram. *A New Kind of Science*. Wolfram Media, Inc., 2002.
- [Wolpert and Macready, 1995] David H. Wolpert and William G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, Santa Fe, NM, 1995.
- [Wolpert and Macready, 1997] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.
- [Wright, 1932] S. Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. In D. F. Jones, editor, *Proceedings of the Sixth International Conference of Genetics*, pages 356–366, Brooklyn Botanic Garden, 1932.
- [Xiao and Armstrong, 2003] Ningchuan Xiao and Marc P. Armstrong. A specialized island model and its application in multiobjective optimization. In Erick Cantú-Paz *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference 2003*, pages 1530–1540, 2003.

## Curriculum Vitae

Zbigniew M. Skolicki was born on January 1, 1977, in Kraków, Poland, and is a Polish citizen. He graduated with distinction and received his Master of Science degree in Computer Science from Jagiellonian University, Kraków, Poland in 2000. He was employed as an assistant in the Institute of Computer Science of Jagiellonian University for a year. He received a Doctor of Philosophy degree in Computer Science from George Mason University, Fairfax, VA, USA in 2007.

He has published around twenty refereed papers including two journal articles and co-authored a book chapter. Many of the papers he presented at major international conferences.