Computational Issues in Long-term Fairness Among Groups of Agents

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

By

Gabriel Catalin Balan
Bachelor of Science
Politehnica University of Bucharest, 2000

Director: Dr. Sean Luke, Professor
Department of Computer Science

Fall Semester 2009
George Mason University
Fairfax, VA

Dedication

To my wife, Cristina

Acknowledgments

I would like to express my gratitude to my advisor, Dr. Sean Luke, for his guidance, patience, and encouragement throughout my years as a graduate student. I am also grateful to Dr. Dana Richards, whose advice and encouragement were crucial throughout the course of this work. Special thanks go to the other members of my committee, Dr. Kenneth De Jong and Dr. Robert Axtell, for their excellent advice throughout this dissertation.

I thank Dr. Octav Olteanu for his invaluable help with some of the early mathematical proofs, and Dr. Liviu Panait, Dr. Tim Sauer, Dr. Jay Shapiro, Dr. Jyh-Ming Lien, Adrian Grăjdeanu and Dr. Hakan Aydin for our discussions. I would also like to extend my appreciation to Keith Sullivan, Jeff Basset, Joey Harrison, Brian Hrolenok, Christopher Vo, Dr. Jyh-Ming Lien, Dr. Zbigniew Skolicki and Deepankar Sharma for their many helpful comments on my drafts and presentation dry-runs. I thank Dr. Zoran Duric, Dr. Alexei Samsonovich, and Dr. Alexander Brodsky for helping me with the related work literature in Russian.

I am indebted to Dr. Sean Luke, Dr. Kenneth De Jong, Dr. Claudio Cioffi-Revilla, Dr. Robert Axtell, and Dr. Gheorghe Tecuci for funding me all these years. The fellowships from the Volgenau School of Information Technology and Engineering, and the subsidies from the Graduate Student Travel Fund were also greatly appreciated.

I am grateful to my parents for their never-ending love and support while being far away from home.

Finally, I thank my wife Cristina for everything; she should have been included in every phrase on this page (except the related work literature in Russian). I just cannot thank her enough.

Table of Contents

List of Tables

List of Figures

Abstract

COMPUTATIONAL ISSUES IN LONG-TERM FAIRNESS AMONG GROUPS OF AGENTS
Gabriel Catalin Balan, PhD
George Mason University, 2009
Dissertation Director: Dr. Sean Luke

Fairness within groups is important to a very broad range of problems, from policies for battery-operated soccer robots to distributed traffic control. While no single action may be fair to everyone, it is possible to achieve long-term optimal fairness for everyone through choice of repeated actions. I explore the issue of achieving such long-term fairness among multiple agents, and provide a unified view of the problem and solutions to it. The issue of constructing long-term fair policies among multiple agents has not been well studied in the literature.

I concentrate on the "average reward" utility model, where one's utility is defined as the average of the rewards one has received from past interactions. Also, I focus on a particular definition of fairness, called "leximin" fairness, but most of the results apply to other measures as well.

After examination of fairness through an infinite series of repeated actions, I extend analysis in several directions. First, I consider how to achieve as fair a result as possible given a finite series of actions, where the length of the series is not precisely known beforehand but rather is chosen from an unknown or stochastic distribution of time horizons. My solution guarantees the beneficiaries the fairest possible long-term results, minus a bounded

worst-case loss due to the game ending unexpectedly. I show that finding sequences of actions with optimal worst-case loss is NP-hard, and I propose a family of approximation algorithms.

Second, I examine stateful domains, where one's choices have side-effects that influence the effects of actions in the future. I introduce a multi-objective genetic algorithm for finding good tradeoff points between beneficiaries utilities and their worst-case losses.

Third, I focus on decision-making processes which have been decentralized in the form of hierarchies. I propose an algorithm based on my stochastic time-horizon solution, and show empirically that an agent hierarchy running that algorithm is able to achieve optimal long-term utilities.

Chapter 1: Introduction

Multi-agent systems (MAS), the study and engineering of complex phenomena emergent from interacting agents, is becoming increasingly important, propelled by several different forces. One force comes from the field of pervasive computing: all of our appliances must work together to offer the best customized experience, regardless of their brand/manufacturer. The field of robotics provides another motivational force: while a huge mining robot might be preferable to several smaller ones, a large number of slow, coordinating robots are very likely to outperform a single faster robot in a surveillance task. The multi-agent approach offers robustness and modularity, which is important since a team of robots must gracefully handle robots joining the team or breaking down. Yet another reason is that people interact more and more through virtual media (auctions, IM, online-gaming, etc.) and the agents acting on behalf of these people need to display intelligence and even some autonomy in their interactions among themselves.

Some modern multi-agent systems require *fairness* in addition to efficiency. Fairness is a highly desirable feature in applications where agents (e.g. web-browsers, email clients, etc) act on behalf of different people and must share some scarce resources (CPU cycles, bandwidth, etc). Alternatively, there are applications where fairness is not germane, but their measure of efficiency can be cast as a fairness measure. Specifically, goals such as network security and full network connectivity abide by the motto "a chain is only as strong as its weakest link," which coincides with the *maximin* fairness concept: "a society's measure of fairness is how it treats its poorest individual." Another example is finding *robust* solutions when dealing with uncertainty in a single-agent setup: a course of actions can have one of several outcomes depending on an unknown state of the world, and bad

outcomes are given larger weights than good outcomes [133]. This maps directly into *prioritarianism*, which assigns larger weights to the worse-off individuals of the society.

Fairness-efficiency tradeoffs are often considered [78,101,197], as fairness and efficiency are frequently conflicting (efficiency might require sacrificing the few for the good of the many, while the converse is true for fairness). A *social welfare measure* combines elements of both fairness and efficiency into one single measure.

The computational aspect of incorporating fairness becomes challenging as the number of agents increases and their interactions get more complex and dynamic. The most obvious example is the internet, with millions of people sending and receiving packets simultaneously. Others come from social choice [22,122,123,147,150] and welfare economics [167, 172]: finding social welfare measures that make things efficient and equitable for societies of hundreds of millions of people.

1.1 Long-term Fairness

I use the terms short-term fairness and long-term fairness to make the following intuitive distinction. Short-term fairness means choosing a fair outcome; long-term fairness refers to a series of outcomes whose aggregation is fair (although the individual outcomes may be short-term unfair). I argued earlier that fairness and efficiency may be conflicting, so decisions optimizing short-term fairness may lead to inefficient outcomes, which in turns makes the entire process inefficient. Alternatively, one may be able to achieve long-term fairness by combining efficient outcomes, leading to superior fairness-efficiency tradeoffs in the long-run. The same principle generalizes to social welfare measures (i.e. combinations of fairness and efficiency): optimizing the immediate social welfare of some population may cause inefficiencies which could have been avoided in the long-run. I present several examples to help with the intuition why long-term fairness has the potential to produce (in the long-run) fairness-efficiency tradeoffs superior to those produce by making short-term

optimizing decisions:

Urban Traffic Control. This problem (the main motivation for this research) consists of controlling the traffic at multiple intersections to achieve a safe and efficient flow of cars through a network of streets. Now imagine traffic lights able to recognize individual cars and access their past stopped-at-red histories. Such a traffic light should be able to take into consideration how much time different cars spent waiting for the green light at other traffic lights, and it should make a more fair decision [9]. The technology already exists: ad-hoc radio networks for inter-vehicle communication [54, 81, 145], and GPS and on board diagnostic (ODB) systems for eliciting position and speed information. A traffic light should be able to identify a car, the proportion of time it spent at red lights, its current position and speed, and use this information, for instance, to decide whether to extend the green light until you cross the intersection, or give the green light to the cars on the perpendicular street.

Assuming the goal is to reduce cars' weighting time the efficient decision in this case would be to give the green light to the street with more cars; the short-term fairness would be to award the green light to the street where cars spent more time waiting at red light(s). A better decision (in the spirit of long-term fairness) might be to take the efficient action, but mark those cars deserving the green light based on fairness such that they receive the green light faster on future (less crowded) intersections [9].

The Urban Traffic Control Problem is large, inherently distributed, highly dynamic and stochastic – overall a very difficult problem. Adding fairness-related goals to its original efficiency-oriented goals complicates the problem even further. For these reasons, the work in this dissertation will not directly solve this problem. I mention the Urban Traffic Control Problem solely as big-picture motivation.

Not only are the problems I end up tackling in this dissertation relevant on their own, but they also share some of the complicating features of the Urban Traffic Control Problem.

Solving these problems should ultimately facilitate approaching the Urban Traffic Control Problem.

Some examples may serve to illustrate the problems solved in this dissertation:

Assigning classes to professors. Consider the case where two AI professors must decide which one will teach each of the two AI classes offered by the department every semester. One class is much harder to teach than the other, so the one stuck with the hard class will perceive the situation as unfair. A (short-term) fair solution could be for the professors to teach both classes together, but they would end up working harder overall than when each teaches one class. Although the situation is unfair every semester, (long-term) fairness can be achieved across semesters by making sure they each teach the hard class the same number of times. In this dissertation I will follow up with more complex versions of this example, adding multiple professors and classes, placing restrictions on the number of semesters the professors work together, and imposing constraints on sequences of assignments of professors to classes.

Battery-operated Robots Consider several robots in charge of patrolling a museum [4, 36, 166]. Some robots may have to work harder than others, given that they may have to cover more (or larger) sections than others. The robots are battery-operated, and it is critical that no robot runs out of power. This can be accomplished by having the robots go through their batteries at roughly the same rate, or giving lighter tasks to the robot lowest on power. These are fairness issues, although treating the robots fairly is not germane to the problem domain.

One possible solution is to keep the patrolling pattern unchanged and have the robots swap their roles in the plan every now and again (or the robot with the harder task will run out of power). Alternatively, one may alternate between several patrolling patterns, specially if the robots are heterogeneous (so an easy task for one robot might be hard for another).

A similar example is the network connectivity problem in a battery-operated wireless network [71]. There are multiple ways to get all the nodes connected, and each solution involves different nodes consuming energy at different rates, because they broadcast at different ranges. The network connectivity goal translates into a fairness problem: having the nodes go through their batteries at the same rate. No broadcasting configuration is very balanced in terms of energy consumption, but one can work around that by switching between broadcasting configurations.

What these examples have in common is *long-term* fairness. A number of beneficiaries interact with a system, and each interaction results in rewards for them. A beneficiary's utility is some aggregation of the rewards it received and/or expects to receive. *Short-term fairness* is the equity of a set of rewards, and *long-term fairness* is the equity of the beneficiaries' utilities (over their lifetimes). Lau and Mui [85] call these concepts *intratemporal* and *intertemporal* fairness.

In my framework the decision process is deployed on one or more *controllers*, whose common goal is to optimize the social welfare of the *beneficiaries*. The idea illustrated by the examples above is that one can combine efficient and unfair short-term *actions* (e.g. task assignments, resource allocations) into an efficient and fair long-term solution. The concept is clearly germane to various multi-agent applications, ranging from "cake-cutting" problems (inheritance division, chore division or rent assignment) [27, 148] to job scheduling [151, 156, 183], satellite sharing [24, 90] and urban traffic control [9, 96, 106]). Yet, most solutions offered to these problems only cover one-shot optimization of various efficiency and fairness metrics. The most notable exception is the body of research in internet traffic control [180, 194, 197], where flow-based fairness is a form of long-term fairness.

In this dissertation I study long-term fairness starting with a very basic model and extending it along several dimensions. In this basic model (called BASE PROBLEM and formally defined in Section 3.5), a single controller chooses infinitely often from a discrete

set of actions. There are a number of passive beneficiaries, and different beneficiaries get different rewards from different actions. The actions have no other effects than to award these rewards, so the process can be regarded as a repeated game, since the actions do not change the state of the system. The goal of the controller is to choose actions such that the beneficiaries' utilities (defined as averages over received rewards) are optimized in a fair way. This paradigm can model repeated resource allocations (e.g. CPU sharing), task (chore) assignments, or social alternatives (such as "spend this year's budget on a sauna" versus "spend this year's budget on a pool," and "elect candidate A to office" versus "elect candidate B"). See Chapter 2 (Section 2.3) for a more detailed discussion of the difference in generality between these actions and the framework in the scheduling literature. Although I assume throughout this dissertation that the actions are given as inputs, I also introduce an algorithm for efficiently generating such actions in a special subclass of resource (task) allocation problems (Section 4.5).

1.1.1 Discussing the Terminology

Before I describe my extensions to the BASE PROBLEM, I will attempt to connect the terminology in my framework with the terminology in related fields, such as game theory, multi-agent systems (and multi-agent learning), queueing networks, and scheduling. By pointing out similarities and differences to elements in these related fields, this section aims to make the concepts in my framework (controller, beneficiary, action) more intuitive. A detailed discussion of the differences between my work and problems in these fields is presented in Chapter 2.

Relation to the game theory literature Game theory [116, 124, 143] models real-world problems using abstract *games* involving *players*. Players receive rewards (in this respect they are the beneficiaries), but they also have actions that affect the rewards (they are also controllers). In my framework the two sets are purposely non-overlapping, in order to have

the controllers *benevolent* (towards to beneficiaries) instead of self-interested.

A widely studied family of games in *cooperative game theory*[1] is called *coalitional games* in [124] and *fair division and costsharing* in [116]. A number of players generate surplus by working together, and the goal is to find a fair way to divide the surplus generated by the *grand coalition* (everybody working together) such that no subset of players has the incentive to deviate and for their own coalition.

In my framework, the controller decides the fair solution, but the beneficiaries are not allowed to opt-out. More importantly, my work is not concerned with how to choose a long-term fair solution, but with how to achieve a given long-term solution (outcome). If one were to generalize (join) these frameworks, then one can use a fairness concepts proposed for coalitional games (e.g. the *core*, or the *Shapley value* [111, 124]) to select a fair, *stable*[2] outcome, and then use my work to find a way to achieve that outcome in the long-run.

Relation to the social choice literature In social choice the beneficiaries are *people* [28] (or *individuals* [97], or *agents* [108]). The actions in my framework correspond directly to *social alternatives*; as a mater of fact, the modeling decision to have the actions affect all beneficiaries to different degrees was borrowed directly from social choice. In a simple voting example, the actions are each of the candidates, and the beneficiaries are the people affected by which candidate gets elected.

Relation to the multi-agent literature Controllers and beneficiaries are both agents (albeit of a very different type). In a survey of issues in multi-agent resource allocation [35], the authors acknowledge that the term *multi-agent* applies both to a distributed allocation procedure (i.e. multiple controllers) and to an aggregation of individual preferences (i.e. multiple beneficiaries). Controllers also correspond to *learning-agents* (or *learning automata*

[1] "The objective of cooperative game is to study ways to enforce and sustain cooperation among agents willing to cooperate" [116].

[2] No group of beneficiaries has an incentive to leave the coalition and form their one group.

[132],[3] or simply *learners* [79,80]) in the multi-agent learning literature [126].

Relation to the queueing networks and scheduling literature A queueing network consists of a finite number of *servers* and *jobs* (or customers) moving from one server to another; when a server is busy, jobs seeking service at that server are placed in a waiting queue. In this framework the beneficiaries are the jobs, the actors are the servers, and an action is selecting from the waiting queue the job that will be served next. For scheduling processes on a multi-CPU machine, an action corresponds to assigning a process to each CPU.

My work is concerned with fairness across multiple interactions, which prunes out a large part of the literature on *scheduling single-stage jobs*. In *job-shop scheduling*, the state space is acyclic (each executed operation takes the job closer to its end), while my work on stateful domains is focusing on taking advantage of cycles in the state-transition graph. A particular formulation of *periodic scheduling* is very similar to a heuristic I use to solve the problem in Chapter 4. This problem is, intuitively, an weighted version of this particular periodic scheduling problem, so although one can use algorithms from the literature to solve my problem, the results would be inferior to those produced by my algorithms (see Section 4.4.2). See also Section 2.3 for a more detailed discussion of the differences between my work and periodic scheduling.

1.2 Overview of Studied Problems

In the previous section I presented the urban traffic control domain as a motivational problem, and argued that the full-blown application is very complex. There are many cars that need to be treated in a fair and efficient matter. There are multiple intersections, and different drivers' utilities depend on the actions of traffic lights at different subsets of intersections. Drivers enter and leave the system asynchronously and disclose little

[3] In [132] a learning agent consists of a hierarchy of learning automata; this is relevant to the work in Chapter 6, where a hierarchy of controllers learn the optimal long-term fair outcome in a task-decomposition problem.

information about their trips. Moreover, the decision process is distributed among traffic lights, which have very limited communication bandwidth available to coordinate their actions.

While this dissertation does not aim at solving such a complex problem, it makes a first step in that direction: it investigates a specific subset of complicating factors that show up, not only in the traffic domain, but in many other complex real-world problems. I start with a basic theoretic model and extend it along the following dimensions:

Finite Time Horizon In real-life applications things rarely last forever, so it makes sense to consider a finite time horizon and study its implications.

Stateful problems If the controller's actions have side-effects beyond giving rewards to beneficiaries, then early decisions affect the results of later decisions.

Multiple controllers One may be forced to distribute the decision making among multiple "controllers," as a centralized solution might not be scalable.

I tackle these issues independently, and leave combining the solutions as future work. Solving each of these problems has value in its own, as there exist problem domains with only a subset of these features.

While these are not the only complicating factors present in complex real-life fairness-oriented applications, casting light on the three issues should make it easier to tackle problems with even more complex issues. Examples of these additional issues include stochastic rewards (see discussion in Section 7.1.1), allowing beneficiaries to enter and leave the system asynchronously (Section 7.1.2), and limited information about the future (lookahead).

In order to motivate the need for studying the impact of all these issues on long-term fairness, I next present several different applications exhibiting these issues.

1.3 Illustrative problem scenarios

In this section I provide examples of applications featuring the complicating issues I identified in the previous section, in order to illustrate how these issues play out in real-world scenarios.

Finite horizon + stateful domain I revisit the "assigning classes to professors" example, where each semester one has to decide the professor that will teach each of the classes being offered. Some classes are more advantageous for some faculty to teach than others, so it makes sense to investigate ways to make the assignments as painless as possible to all professors. The professors are the **beneficiaries** in this application. A **reward** is a professor's degree of satisfaction with a particular assignment. A reward may cover not only the classes the professor teaches under that assignment, but also the times and locations of those classes, and even an envy-like component dependent of the other professors' classes. The department head is the single **actor**, and the **actions** are the set of feasible assignments, which stays the same every semester.

This process is not likely to go on forever, as management might switch to a different assignment policy (finite time-horizon). In this domain the sequence of assignments might be subject to some constraints: for instance some classes cannot be offered two semesters in a row (stateful domains). Similar scenarios apply for scheduling nurse shifts and assigning postal worker delivery routes.

Multiple controllers This is another battery-powered robot example, with applications to surveillance and military readiness. The task is managing a swarm of unmanned air vehicles (UAVs) patrolling the borders. The connection to fairness is identical to that of my earlier museum-patrolling example: the more energy the robot lowest on energy (poorest beneficiary) has, the longer one can keep the entire team in the air. If several sectors need to be covered, one can divide the UAVs into squadrons and have an controller repeatedly

assign squadrons to sectors. Imagine each sector is further divided into areas; it makes sense to divide squadrons into squads and have additional controllers overseeing the activity in each of the sectors.

I will revisit these two examples throughout this dissertation to motivate the problems I tackle and illustrate the solutions I propose. I will now present additional examples, to further illustrate the wide applicability of my research.

Multiple controllers + stateful domain Consider assigning shifts to nurses in a large hospital. There are restrictions to a nurse's sequence of shifts, making this a stateful problem. If the hospital consists of several wards, one can have several controllers handle each ward separately. This becomes a coordination problem if wards are allowed to "loan" nurses to other wards, or if there is a small pool of unassigned (floater) nurses that wards can share.

Multiple controllers Another inherently decentralized application domain is the urban traffic control. The cars are the beneficiaries, and the traffic lights are the controllers in charge of dealing with cars in a fair and efficient way. The control must be decentralized for reasons including scalability (the problem is too big), reliability (one cannot afford to have the central computer crash) and real-time constraints.

Multiple controllers Consider the problem of making a list of songs for a NPR (National Public Radio). In this problem the listeners are beneficiaries which synchronously interact with one controller (the radio station) by listening to each individual song, and derive different utility levels from it. The goal is to choose the right combination of actions (i.e. songs), such that everybody is "equally happy" in the long run. This problem is interesting on its own, as it is not immediately obvious how to rotate the songs. The problem extends naturally to multiple controllers: there are multiple NPRs with overlapping reception areas, and each beneficiary is free to switch between them. A similar example is the DVD rental

domain of Abraham et al. [3].

Time-discounted Rewards Another aspect of the long-term fairness framework is the beneficiaries' impatience. Consider the situation of the countries in the European Union, which use a rotation system to assign the presidency of the Council of the EU. There are economic advantages associated with holding the presidency of the EU council, and since one's turn comes once every many years, there is a strong tendency towards impatience.[4] Because the set of beneficiaries keeps changing (new countries join EU), the incentive to "go first" in the rotation is stronger.

Formally, beneficiaries can use different functions to compute their intertemporal utilities from the rewards they get from the interactions. In the traffic application the order in which a beneficiary receives his rewards or penalties is irrelevant, and so summing or averaging the rewards is suitable for that domain. In the EU application, a time-discounted approach is appropriate. Deciding on a suitable utility function is connected to the frequency of the interactions.

The cross-temporal framework presented here allows one to choose (1) an aggregation function used to compute a beneficiary's utility based on its interactions and (2) an aggregation function to compute the social welfare based on individual utility levels for each of the beneficiaries. This dissertation focuses on the *average-reward* utility model for the first function and a specific concept called *leximin* (Section 2.1) for the second function. The results presented here hold for social welfare measures other than leximin (see Section 4.3.4, and footnote 10 in Section 5.4.1). However, they are incompatible with the *time-discounted reward* utility model (as a matter of fact, I argue in Section 7.2 that outside a narrow class of instances, finding the optimal utility profile under the time-discounted reward model is an open problem).

[4]The motivation for why getting money sooner is better than later is better explained through bank interest. If one is offered the choice of receiving $1K today or in a year, one should take the money today: one can always put it in a bank, and in a year one will still have the $1K, plus the bank interest.

1.4 Overview of Contributions

Extend leximin fairness to situations applicable to real-world problems. Finite stochastic time-horizons, stateful and distributed domains are ubiquitous features of real-life problem domains. Therefore, in addition to providing concrete algorithms, my work extends the formal understanding of leximin fairness to more realistic scenarios. Leximin is widely used in the fairness literature, so many previous research efforts (e.g. satellite sharing [24, 90]) can benefit from my work. (1) The quality of their solutions will improve by taking advantage of the repeated-interaction nature of their applications (which they have ignored so far). (2) They can tackle a wider, more realistic range of problems.

Provide ground-breaking work in the area of long-term fairness in repeated games. Although long-term fairness has the potential for increased fairness and efficiency over single-shot interactions, this issue has been insufficiently studied in the literature. Both the solution concept and the actual algorithms proposed in this dissertation are directly transferable to fairness measures other than leximin (see the discussion in Section 4.3.4). Of particular importance are the tradeoffs between leximin fairness and efficiency, such as stratified egalitarism and ordered weighted averaging (Section 2.1), particularly for applications where the efficiency guarantees of leximin fairness are too weak.

Improve fairness and efficiency in real-world problems. In this introduction I have used the problem of repeatedly assigning classes to professors to illustrate and motivate my work. This research is directly applicable to other similar fairness-oriented, *repeated, chore-division tasks*, such as shift rotation for nurses and route assignment for postal workers. In all these cases, by increasing the employees' perceived fairness, one also increases their utilities and their productivity.

One example of *resource allocation* application for this research comes from Massively

Multi-player Online (MMO) games, where players in a "guild" get together to play cooperative games several times a week, but only k of them can play at a time. In one such game's on-line forum the players actually expressed their need for a fair rotation scheme, which is precisely what my algorithms would provide. A second example comes from politics: the European Union uses a rotation system to assign the presidency of the Council of the EU, and my work would provide a principled approach to achieving a fair rotation. A third example is deciding the location of a periodic athletic event. The objectives to be leximin-optimized can be either the revenues generated by the event to the hosting cities (resource allocation) or the distances driven by the attendants (chore division). Recently, it has been increasingly common for a large international sport event to be hosted by two countries: Japan and South Korea (2002 Soccer World Cup); Austria and Switzerland (2008 European Soccer Championship); and Poland and Ukraine (2012 European Soccer Championship). This shows the wider applicability of my framework (where actions affect multiple beneficiaries to different degrees) rather than the traditional scheduling framework (where one job is served and all the others wait).

1.5 Dissertation Layout

The rest of this dissertation is organized as follows: Chapter 2 contains a literature review as it relates to long-term fairness (fairness measures, fair scheduling and resource allocations), and multi-agent coordination. Chapter 3 gives a theoretical analysis of an abstract infinite-length, single-state, single-controller model. This model is extended with a finite time-horizon (Chapter 4), multiple-states (Chapter 5) and a controller hierarchy (Chapter 6). Finally, Chapter 7 provides future work directions and a list of open problems, and Chapter 8 concludes the dissertation, with emphasis on contributions.

Chapter 2: Related Work

This chapter consists of several parts. I start with a review of fairness and social welfare measures (Section 2.1), with emphasis on *leximin*, the social welfare measure I will use extensively throughout the dissertation. I follow with a discussion of related work in the fields of game theory (Section 2.2) and queueing theory –scheduling in particular– (Section 2.3), which are relevant to the long-term fairness concept. An overview of the literature on fairness in resource allocation is presented in Section 2.4, as it is relevant to the work in Section 4.5. Section 2.5 contains related work relevant to the stateful problem I solve in Chapter 5. I end with a multi-agent coordination discussion (Section 2.6), relevant to the controller-hierarchy work I described in Chapter 6.

The research framework in this dissertation consists of controllers making sequences of decisions that affect multiple beneficiaries. There are three key concepts at work here: (1) the beneficiaries are separate from the controllers, (2) there are repeated interactions, and (3) each interaction may affect all beneficiaries to different extents. In Game Theory there are repeated games, different joint actions affect different players differently, but the players are both controllers and beneficiaries. In scheduling, the schedulers (i.e. the controllers) are different from the jobs (the beneficiaries), but the actions are more restrictive than in my case: they typically affect all but one beneficiary in the the same way (while one job gets the CPU, all the others are waiting). In resource allocation (e.g. CPU scheduling) each resource is assigned to a single beneficiary (job). In task allocation (e.g. assigning classes to professors), each task (class) to assigned to a single beneficiary (professor). The actions in my framework are more general: in the urban traffic domain, the resource (the green light) is awarded to multiple beneficiaries (all the cars in several non-conflicting lanes). Furthermore, one can use the actions in my framework to model each of the candidates in

an election. See Section 2.3 for a more detailed discussion of the differences between my framework and the scheduling framework (including multiple CPUs with different speeds).

2.1 Fairness and Social Welfare Measures

Fairness is an inherently desirable trait in human societies. Intuitively, *it is only fair that people in identical situations should get equal treatment*; however, this intuition alone is not always able to settle which of two alternatives has a more fair social outcome. If one alternative helps some people and hurts others (when compared to the other alternative), it may not be obvious which alternative is more fair. One example is from the realm of taxation: it is fair that people with the same income pay the same amount of taxes; furthermore, it is intuitively fair the higher the income the higher the taxes. However, different people may be similar, but not identical (they may have different needs, abilities, or preferences); same might be true for their situations. Two people have the same income, but one drives much more to get to work, or he needs insulin shots every day. If one person can produce w_1 widgets per day and a second person can produce $w_2 < w_1$ widgets, what would be a fair way to divide the widgets at the end of the day? What if the two people have equal production abilities, but the second one is lazy and chooses to produce less? Another example is offering the available stock of a vaccine to children and the elderly: people of the same age get the same treatment; but should the vaccine go to a 40 year-old father of four rather than to a 65 year-old with no immediate family? Based on these examples, it comes as no surprise that fairness has been the focus of a large body of work in philosophy, economics, social choice, law (justice), etc. [22, 28, 31, 45, 58, 59, 102, 103, 122, 139, 143, 144, 147, 167, 170, 172, 181]. In this dissertation I ignore the philosophical aspects and instead focus on computational issues of fairness. In this section I review a number of classes of fairness measures proposed in the literature, with emphasis on those related to the fairness concept I will use throughout the dissertation, namely *leximin*.

One of the classes of fairness measures that have been proposed in the literature is the

Gini family of indices: the index of dissimilarity, Gini's coefficient, absolute mean difference, relative mean difference, etc. [64, 117, 149]. A second class of fairness measures is based on the *variance* statistic: standard deviation, coefficient of variation, and Jain's *fairness index* [75]. The members of the first class are strategically equivalent to summing the absolute values of the differences between any pair of beneficiary utility values. The members of the second class are strategically equivalent to summing the squares of those differences. One can easily generalize these by summing the absolute value of all differences raised to power p, and putting increasingly more weight on large differences than on small differences simply by increasing p. More generally, the Hölder's L_p norms [84] sum the actual values raised to power p. Thus, as long as all values are non-negative, these functions cover both variance-related measures such as coefficient of variation ($p = 2$) and leximin ($p = \infty$), which will be introduced shortly. Interestingly enough, for $p = 1$ they model *efficiency* (i.e. social utility). Another example is the *ordered weighted averaging* family of functions [35, 185, 186], which can model Gini's coefficient (the values need to be normalized first), approximate *leximin*, and again, model efficiency.

A third class consists of *maximin* and its refinements: *discrimin*, *leximin*, and *max-min*. I plan to focus my research on refinements of maximin fairness, and leximin in particular. Maximin [177, 178] chooses between alternatives solely based on the utilities of the worst-off under each alternative. This fairness concept is motivated by the "veil of ignorance" argument of philosopher John Rawls [144]: if one were to choose a society before knowing one's role in that society, one should choose based on how well a society's worst-off are treated. In the beginning of this dissertation I argued for the necessity of fairness in self-interested agent domains; but also that maximin is applicable to cooperative problem domains: maintaining connectivity in battery operated wireless network (maximizing the time until the first node runs out of power), computer security (the chain is as strong as its weakest link), etc.

The major drawback of maximin is its failure to satisfy Pareto-efficiency: society should

be better off if at least one person is better-off and no one is worse off in the process. It turns out that all the fairness measures related to variance and Gini's coefficient can lead to poor Pareto-efficiency. Imagine a situation where a group of N people, each in possession of one indivisible unit of a certain good, comes across another unit; as a result, one of the people will have two units. While maximin cannot decide if the extra unit of good improves the social welfare or not, the variance and Gini-based measures consider the extra good unit as detrimental.

One example of a maximin and Pareto optimal measure is *discrimin* (equivalent concepts were independently published as "protective criterion" and "no reason for regret" [61]). Under the discrimin definition, an outcome A dominates another outcome B if either of the following conditions holds.

- A Pareto-dominates B.
- Whoever is worse-off in the alternative A is still better off in A than a different person would have been in B, and for which A brings a strict improvement.

A refinement of the discrimin concept is *leximin* fairness [24, 25, 82, 117, 162]. Leximin chooses between alternatives by comparing the utilities of the worst-off in those situations; but unlike strict maximin, leximin breaks ties by comparing the utilities of the second-worst individuals, and so on. Leximin considers two outcomes equal if and only if the corresponding utility vectors are permutations of each other.

I use an example from [61], to show the difference between leximin and discrimin. Two social alternatives A_1 and A_2 produce the following tuples of rewards for the two beneficiaries: $A_1 = [0.8, 0.4]$ and $A_2 = [0.4, 0.5]$. Leximin prefers A_1 to A_2, while discrimin considers them equal.

Leximin is further refined in the networking community by the *max-min* fairness concept [141]. An outcome is max-min-optimal if and only if any alternative that makes somebody strictly better off also makes somebody else already worse-off even worse. There cannot

be more than one max-min-optimal solution, but it is possible that none exists, and this weakens the appeal of the concept. If a max-min optimal solution does exist, it is the unique leximin optimal solution, but the converse is not true: reusing the example from the previous paragraph, $A_1 = [0.8, 0.4]$ is the unique leximin-optimum, yet there is no max-min optimal solution. Thus, among maximin measures in the literature, leximin is the most powerful one that still guarantees a solution.

A more general argument for leximin is the following result from [22]: leximin is the only social welfare ordering to satisfy Hammond equity,[1] Pareto efficiency and anonymity[2] — three very reasonable properties. Other properties resulting from this axiomatic formulation of leximin include transitivity and *separability with respect to unconcerned individuals*: the social ordering relation between two alternatives is not influenced by the actual utility values of the persons unaffected by the decision.

Leximin also finds a wide use in a broad range of applications, including multi-objective constraint satisfaction [24, 46, 61, 107, 186]; multi-agent resource allocation [35] (satellite sharing [24, 25] or bandwidth allocation [110, 121, 141]); telecommunication network design [119]; facility (e.g. emergency services) placement [117, 162]; and movie encoding [73] (for reducing variation in the quality of individual frames). In the field of game theory, leximin has been used in the context of coalition games to define the *nucleolus* [41, 124], a fairness-oriented relaxation of the *core*, a stability concept where all players are self-interested and no subset of players has an incentive to defect.

To summarize, leximin has a strong theoretical foundation (it uniquely follows from a very few, reasonable, axioms), it is Pareto efficient, and has been widely used and analyzed. There are however a few disadvantages. Leximin fails to satisfy the *continuity* property, which, intuitively, ensures that small changes in individual utilities do not produce large

[1] The *Hammond equity* principle states that when comparing two alternatives in a two person situation (i.e. the alternatives differ on only two objectives), if the first person is worse off under his preferred alternative than the other person is under his less preferred alternative, then the poorer person's preferred alternative should be socially preferred.

[2] The *Anonymity* principle requires that any permutation of the utility levels of an social outcome should result in an equally desirable alternative.

changes in the social ordering. This may prove relevant when computing approximations of leximin. Another disadvantage for leximin is that it "does not introduce directly any scalar measure" [117], meaning, there is no function for this partial order relationship. You can compare two outcomes, but you can't assign scores to outcomes and then compare them using the scores. This is related to the continuity property, since any continuous social welfare ordering can be represented by a social welfare function [22].

2.1.1 Leximin-efficiency Tradeoffs

It was argued in the literature (e.g. [167]) that giving absolute priority to those worst-off is too drastic for some applications, and milder alternatives have been proposed. An example is *stratified egalitarianism* [107], which uses a "poverty threshold" to prefer outcomes that improve the utilities of some of those below the poverty line, even at the expense of those over the line. Another measure is *ordered weighted averaging* [35,185], which is parameterized to optimize efficiency at one extreme and leximin fairness at the other, with a continuum of behaviors in between.

Prioritarianism is yet another generic family of functions between leximin and utilitarianism. It consists of selecting a positive, increasing, and strictly concave 1-d function, which gives the additive contribution of an individual's reward to the social welfare. The name comes from the fact that it gives priority to the worst-off (but not absolute as in leximin, and not none as in utilitarianism).

Consider the following p-parameterized family of measures: $sgn(p)\sum_b U_b^p$ [35, 109, 147] (with the convention $\sum_b \log(U_b)$ when $p = 0$). Note that it is a particular case of *prioritarianism* when $p \leq 1$ (for $p > 1$ one gets an elitism-flavored prioritarianism, which prefers helping the rich more than the poor). Furthermore, it is equivalent to the utilitarian social welfare when $p = 1$, and converges towards leximin as $p \to -\infty$.

Another parameterized family of measures is $sgn(p)\sum_b e^{pU_b}$ (with the convention $\sum_b U_b$ when $p = 0$) [108,147]. This is equivalent to the utilitarian social welfare when $p = 0$ and

converges towards leximin as $p \to -\infty$.

For completeness I mention that there are other ways of mixing efficiency and fairness in a single function, such as the maximization of the Nash product [170], the minimization of the sum of squared delays [165], Young's stochastic stability concept [189], etc.

2.1.2 Algorithms for optimizing leximin

This section presents a number of algorithms proposed for the solving of various problems with leximin objectives. I include this section because the long-term fairness algorithms in this dissertation use the solution of such problems as inputs, and here I discuss how such solutions can be found.

The common approach to optimizing leximin in the literature is to solve a sequence of maximin optimization problems. This pattern is a direct consequence of the definition of leximin: the leximin optimal solution maximizes the utility of its worst-off person; out of all solutions that achieve that maximal value for their "min", the next worst-off is maximized, and so on. Solving the maximin subproblems is domain specific: one should use constraint satisfaction for CSP domains, linear programming for convex search spaces, etc.

The algorithm proposed by Potters and Tijs in [138] searches a compact and convex space for solutions that leximin-optimize a number of linear *objective functions*. An objective function F is the function associated with an optimization problem: solving the problem means finding a solution x such that $F(x)$ is optimized. The problems in this section have multiple objectives, and one uses leximin to find the solution y that leximin-optimize the values of those functions in y. In my framework the objective functions are the beneficiaries' utilities, so I will use objective function and utility function interchangeably in this section.

The algorithm of Potters and Tijs works by restricting the search space as the values of the worst-off objective functions are iteratively optimized. The closed form of the objectives (linear functions in this case) allows for a convenient representation of the sequence of search space restrictions, and the algorithm is reduced to solving a sequence of linear

programs (LP). This approach can also be applied to convex search spaces [119].

Bouveret and Lemaître [24,25] provide an alternative way to restrict the domain between maximin optimizations. The k^{th} maximin problem is of the form: find the largest value that is only larger than at most $n - k$ objectives, whichever they are. This meta-constraint, which limits the number of constraints that can be violated by a feasible solution, is actually implemented by introducing integer variables in the linear program. LPs that require some of their variables to take integer values (integrality constraints) are considerably harder to solve, so the approach of Bouveret and Lemaître is suitable if the domain already imposes integrality constraints on its solutions.

Stephen et al [163] optimize leximin in the context of resource allocation. A company is able to perform a number of *activities* (e.g. it can manufacture a number of different products); and performing a unit of an activity consumes different quantities of various resources. The goal is to decide the *activity levels* that leximin-optimize a number of objective functions, subject to budget constraints for each resource type. The authors provide algorithms for the multi-period extension (where unused resources can be stored and new resources become available between periods), and for the case when some resources can be substituted for others. Klein et al [82] improve the algorithm by eliminating variables (dimensions of the search space) when appropriate. Note that in the multi-period extension the number of periods is known, unlike my work Chapter 4, where the number of periods is unknown (or stochastic).

Ogryczak [117] solves the discrete facility location problem, based on the observation that leximin is invariant to re-writing the objective functions by trading values between them.

As stated in Section 2.1, there is no equivalent *function* for the leximin *order relationship* in general; however, this is no longer true when the objective functions are restricted to discrete values. Yager [186] proposes such a function when the difference between any two utility values is bounded by some constant Δ. The idea can be used to find leximin-optimal

solutions in non-convex search spaces, but integer variables are required. Ogryczak and Śliwiński [121] extend the idea and propose an algorithm for computing leximin when utility functions are restricted to an arbitrary set of discrete values. In a slightly more general framework, Ogryczak et al [118] consider reducing the number of allowed values to reduce computation time; they prove a result on the domination relationship between the solutions found in the original problem and the reduced value-domain problem.

2.2 Game Theory and Long-term Fairness

In the introduction I presented a basic setup as a starting point for my research. The setup consists of one controller making a infinite sequence of decisions (I define this formally in Problem 1, Section 3.2). This formulation resembles an infinitely repeated game from game theory. So much so that long-term periodic solutions are intuitively motivated in the literature [18, 85, 86, 178] using example borrowed from game theory. The fundamental difference is that in game theory the controllers are also the beneficiaries of the systems, and –most importantly– they are self-interested.

Consider the "Battle of the Sexes" game: two players choose simultaneously between going to a boxing game or a ballet performance. Each player prefers a different destination, but prefer each other's company even more. A player gets a utility of 1 if he or she ends up in the same place as the other player, plus another point if he or she is at his or her preferred place. They both get 0 utility if they miscoordinate. This game might have been created for studying coordination, but it is useful in motivating a periodic solution: players should alternate between destinations (i.e. "take turns").

Bhaskar [18] and Lau and Mui [85–87] studied the emergence of turn-taking and its stability in repeated "Battle of the Sexes" games. Alternating between the destinations is more fair and just as efficient as the pure Nash equilibria, and just as fair but more efficient than the mixed Nash equilibrium. However, there is the issue of who makes

the first concession (especially in time-discounted scenarios). To this end, Lau and Mui propose *turn-taking with independent randomization* (TTIR) [85–87], where the two players keep flipping coins until they manage to coordinate, and then they alternate. Neill [112] uses simple learners to study the emergence of cooperation and coordination in the "turn-taking dilemma," a family of repeated games derived from prisoner's dilemma. In all this body of work players consider long-term plans for selfish reasons; fairness is just the ingredient for stability. Although the issue of coordinating a long-term plan and its stability are important, to my knowledge they have only been studied in 2×2 games.

Nowé et al. [177] and Verbeeck et al [178] start with the repeated Battle of the Sexes game, and proceed to games involving more than two players. However, their framework assumes the players are actually cooperative learning agents (rather than self-interested) trying to coordinate on what the authors call a "periodic policy." The process consists of the learners playing selfishly to discover a pure Nash equilibrium, being interrupted periodically to compare accumulated utilities. The player gaining the most (in the current Nash equilibrium and overall) has its action off-limits until the others catch up. Alternatively [177], after the learners discover all pure Nash equilibria, they create a periodic policy consisting of those joint actions with the fairest outcomes. In the authors' examples the players have only two actions: a highly lucrative one and a social one they fall back on while waiting for the other(s) to catch up. I believe that the first approach could be arbitrarily inefficient when players have more than two actions: the players do not learn from one period to the next that some Nash equilibria should not be played. The second approach is inefficient as well, since myopically choosing the fairest NE(s) could be arbitrarily worse than the optimal periodic policy. For example, consider augmenting the Battle of the Sexes game by adding another destination they equally dislike: the combination of unfair outcomes $[1, 2]$ and $[2, 1]$ is more efficient than and just as fair as an outcome of $[1.2, 1.2]$, for example. The notation $[a, b]$ refers to the rewards the two players get when playing a joint action. Peeters et al. [130–132] extend the work in [177, 178] to tree-shaped multi-state domains. The authors'

hierarchies of learning automata are able to converge to more complex multi-stage NEs, but my previous argument still stands.

2.3 Fair scheduling

Queueing theory is concerned with predicting statistics for a server system handling jobs featuring various arrival rate distributions and processing requirement distributions. Sometimes arriving jobs cannot be "served" right away, and are placed in waiting queues. The part relevant to my work is the scheduling part, which decides which job should be processed next. One can see it as a prioritized queue: the scheduler assigns priorities to all jobs pending in the queue, and then selects the one with the highest priority. Job scheduling is concerned with the design and analysis of on-line algorithms for scheduling dynamically arriving jobs to one or more servers. Most commonly, but not necessarily, a single job can be processed at a time. In preemptive scheduling a job currently running will be postponed if a higher-priority job arrives; in non-preemptive scheduling jobs cannot be suspended.

Most scheduling policies fall in one of the following three categories: age based, size based, or remaining size based. The age-based category contains policies such as First Come First Served, Last Come First Served and Least Attained Service. Size-based policies include Shortest Job First, Longest Job First, and their preemptive equivalents. In the third category one finds Shortest Remaining Processing Time (Least Work Left) [12, 72, 156], Longest Remaining Processing Time and Fair Sojourn Protocol [63].

Some of those heuristics adhere to a *proportional* fairness concept: longer jobs should wait longer than short jobs; and that waiting time should be proportional to the job's size. First Come First Served disadvantages short jobs, because they get high delays relative to their size when stuck behind long jobs. Shortest Remaining Processing Time addresses this issue, and by doing so it minimizes the mean response time of the system. See [182] for a taxonomy of scheduling algorithms with respect to proportional fairness in M/GI/1

servers. The categories are always fair, sometimes unfair and always unfair, meaning that jobs of all sizes get a fair treatment under *all*, *some* or *no* combination of server utilization and service time distribution.

A different fairness concept is the "temporal fairness" one: it is not fair to a long job that short jobs keep on cutting in line although they arrive later than the long job. Under the temporal fairness concept, First Come First Served is fair and Shortest Remaining Processing Time is unfair to long jobs. Sabin et al [151] argue for a fairness concept that will not allow a later arriving job to delay an earlier arriving job, and propose a way to quantify the unfairness of a given non-preemptive algorithm with respect to the said fairness concept.

Tradeoffs have been proposed to balance the two fairness concepts: Order Fairness [8], Resource Allocation Queueing Fairness Measure [146], Discrimination Frequency [152,153].

Two of the main applications for the policies in this section are the scheduling of computationally-intensive jobs on mainframe computers [156] and processing of HTTP requests by web-servers [63,72].

The fairness concepts (and algorithms) so far refer to fairness between jobs; the long-term fairness concept I focus on is appropriate for a situation where beneficiaries repeatedly send jobs, and the goal is to have the beneficiaries achieve maximally fair utilities out of all their jobs.

Periodic scheduling A particular case of job scheduling that is more relevant to the problems in this dissertation is the periodic scheduling problem [129,179]. In this problem a fixed set of beneficiaries send jobs periodically (e.g. a process needs 2 seconds of CPU every 5 seconds). This setup is relevant to operating systems for applications with real-time requirements and battery-operated wireless sensor networks, which can save power by broadcasting in turns to prevent simultaneous broadcasts. In [26] the time is split into unit time intervals, and each interval can be assigned to a single beneficiary. The goal is to come up with a long term schedule such that everybody meets all their periodic deadlines. One

goal is to give each beneficiary a fraction of time intervals to closely approximate his service demand. Another goal is to have a beneficiary's time units allocated as evenly as possible.

The research I present in this dissertation has many features in common with previous work on periodic scheduling (such as periodic solutions and the use of approximations), but periodic scheduling algorithms are not directly applicable to my problem formulations.

- A high-level difference is that in periodic scheduling the goal is to allow the beneficiaries to meet their periodic goals; my work aims at periodic plans that optimize beneficiaries' utilities in fair way. Furthermore, I focus on the leximin measure, which is harder to accommodate than maximin [26, 129].

- An important difference is the way the actions that make up the long-term plans affect the beneficiaries. In my work, an action affects all beneficiaries' utilities in different ways, while scheduling-type actions are of a very specific form: "process job x" moves only job x closer to completion and delays every other job. There is also work on periodic multi-processor scheduling [14, 15, 142, 188], but it follows on models less general than my framework. In the "identical parallel machines" model the CPUs are identical, which means the set of possible reward values consists of exactly two elements (run for one time unit and wait for one time unit). In the "parallel uniform machines" model, all jobs progress at the same rate on each CPU (the rate is the CPU's speed), so the size of the set of reward values is at most one plus the number of CPUs. Conversely, the number of reward values in my framework is potentially much larger (at most the number of beneficiaries times the number of actions).

- Lastly, consider the professors and classes example from Section 1.1. The classes correspond to the jobs (they are the tasks), and professors correspond to the CPUs (they execute the tasks). However, in my framework the professors are the beneficiaries, while in the scheduling domain the beneficiaries are the jobs. Furthermore, professors can teach more than one class in a semester, while most results in the scheduling

literature depend on the assumption that a single job can run on any CPU at any given time [142].

The periodic scheduling framework has an indirect application to my work: in Chapter 4 I propose a family of algorithms based on the heuristic that actions should be scheduled such that their usage proportions are always as close as possible to a fixed vector of proportions $F^\star$. This is very similar to the CHAIRMAN ASSIGNMENT PROBLEM formulation, where one has to minimize the largest deviations (both positive and negative) between $F^\star$ and the actions' usage proportions at all times. It follows that algorithms proposed for the CHAIRMAN ASSIGNMENT PROBLEM [93, 155, 169] can be used to solve my problem in Chapter 4. As I argue in Section 4.4.2), my ultimate goal is not to make sure no action deviates too much from its ideal proportion, but that no beneficiary deviates too much from its utility in the optimal utility profile. One can actually get better results by allowing some actions to deviate more than others, and the algorithms I propose take advantage of this observation (see Section 4.2.1). Therefore the scheduling algorithms proposed for the CHAIRMAN ASSIGNMENT PROBLEM can be used to solve my problem in Chapter 4, but they yield suboptimal results because they constrain all actions' deviations uniformly (see Section 4.4.2).

Here are some other efforts in periodic scheduling that are relevant (to a lesser degree) to the work in this dissertation:

- In *perfectly* periodic scheduling each job (action) needs to be used exactly every some predefined number of time steps.
  - In [13] the authors restrict the values in $F^\star$ to $\{\frac{1}{k} | k \in \mathbb{N}\}$, which conflicts with the type of guarantees I offer (I need my actions to be used in *exactly* the right proportions at the limit).
  - In [33, 129] the schedule contains idle time steps, when no job gets the resource. In the original problem, the loss in the utilization of the resource is compensated

by distributedness: each agent knows exactly when it's its turn to get the resource (e.g. broadcast on the wireless channel), so there is no need for a centralized arbitrator. Idle steps make little sense in my framework (an action must be chosen at each time step); moreover, there is no gain in having a different controller in charge of each action.

- Litman and Moran-Schein [94, 95] study *smooth schedules*, where the number of time slots a job got during any interval is close to the number of time slots it was entitled to (the job's $F^\star$ value multiplied by the size of the interval). However, the resulting schedules have idle steps or there are restrictions on the values in $F^\star$.

- Dawande et al [38] study periodic solutions to scheduling activities for a robotic arm serving several machines. This is a stateful domain: one can only take a product (part) from a machine if there is a part there and the machine is done processing it. The problem's goal is maximizing long-term throughput; there is no fairness-related goal.

Proportional share scheduling Another scheduling subarea is concerned with the following problem: there are a number of job classes, and each class is entitled to an a priori decided share of resources. The goal is to have every class get the share it is entitled to, which is a fairness goal. This is a common formulation in operating systems (different fractions of CPU bandwidth are set aside for different processes) [11, 52, 53, 76, 113] and network routing (e.g. 10% of capacity is reserved to multimedia traffic) [89, 180, 194, 197].

Examples of CPU scheduling policy is Decay-Usage scheduling [52] and ALPS [113]. Popular scheduling policies for networks include Weighted Fair Queueing, Self-Clocked Fair Queueing, Worst-case Fair Weighted Fair Queueing, Deficit Round Robin and Elastic Round Robin. Aside from deciding the flow that sends the next packet, a router needs to decide what flow loses a packet when the router is overflooded. Examples of policies include Random Drop, Early Random Drop, Random Early Detection, Fuzzy Threshold, Fair Early Random Detection, and Fair-Buffering Early Detection [194].

In this section, the resources are time-shared, i.e. the jobs are switched with high frequency (microseconds for networks, seconds for CPU), while previously the resource would switch jobs only when a job was finished or a high priority job arrives. This time-sharing allows for a fine control over the allocation of resource fractions to different classes. However, the crucial item is the same as before: each action of the scheduler affects a *single* beneficiary, while in my formulation actions affect *all* beneficiaries in different ways. Intuitively, it is much simpler to get fair utilities by combining reward vectors where all entries but one are zero then by combining arbitrary value reward vectors. Using the professor-assignment domain, this maps into a situation where professors have identical preferences, and all classes but one are equally hard; a trivial solution exists: the professors teach the easy class in a Round Robin fashion. This observation makes my work different from scheduling policy research in general. The scheduling problems are, of course, anything but trivial: the decision process is distributed, the world is highly dynamic, and the schedulers need to make their decisions very quickly.

Somewhat closer to my approach of having actions affect all beneficiaries to different degrees is the work in [53]. The authors study a multi-controller case for the CPU sharing situation. Rather than having each CPU treat the classes of jobs according to their entitlement shares, the authors investigate having different CPUs give preferential treatment to different classes, such that, overall, the jobs still make progress proportionally to their classes' fair shares.

Job Shop Scheduling The Job Shop Problem is another generalization of the single-server scheduling problem discussed in the beginning of Section 2.3. The Job Shop Problem [69, 77, 158, 168, 173, 187] is concerned with the execution of jobs consisting of different *operations*, which can be executed on a set of machines. The machines are heterogeneous, so an operation can have different durations when executed on different machines. The most common optimization goals in the Job Shop Problem are the minimization of make-span

(the amount of time needed to complete a given set of jobs), the minimization of the number of jobs missing their deadlines, balancing the utilization of resources and maximization of the production rate [77]. A variant of this problem is concerned with optimizing the cost (rather than the performance) of a schedule. The cost of a schedule comes from machine setups (reconfiguring a machine between operations), inventory costs, penalties for missing deadlines, etc. [69].

This problem is relevant to this dissertation because jobs interact with the system multiple times, and taking advantage of repeated interactions is an important part of this research. Moreover, the constraints imposed on a feasible schedule (a job must have its operations executed in a precise order) make the Job Shop Problem bear some similarity to the Stateful Problem I solve in Chapter 5. There are, however, a number of important differences. In my framework the number of times a beneficiary interacts with the system is infinite or unknown, whereas the jobs' decomposition into operations is given a priori in the Job Shop Problem. Another difference is that my work focuses on taking advantage of the cyclic nature of some problem domains (e.g. Section 2.2, Chapter 5), which is not present in the Job Shop Problem: every executed operation takes the job closer to its completion.

2.4 Fair Resource Allocation

Fair division of infinitely divisible resources has been studied extensively in the "cake-cutting" literature [27, 148]. In this section I focus on the case of indivisible resources, because it is relevant to my work in Section 4.5.

One goal of this problem is to allocate a set of items to a set of beneficiaries, such as to maximize smallest utility over all beneficiaries (i.e. the utility of the poorest beneficiary is maximized) [7, 17, 32, 68]. Note that this is in fact *max-min*, a weak form of *leximin* (see Section 2.1). Another goal is finding an *envy-free*[3] allocation, or at least one with

[3] An allocation is envy-free if no beneficiary could gain by giving up all items it received and taking all item some other beneficiary received.

small bounded envy [92]. Both problem variations are NP-hard, even for additive utility functions.[4] Notice that the first problem with linear utility functions is closely related to the minimum makespan scheduling problem.[5]

Most of the proposed algorithms for this problems compute an optimal fractional, then transform it into an feasible (integral) allocation. While a fractional allocation is infeasible in an "one-shot" interaction, my framework allows me to use multiple assignments to achieve the optimal fractional allocation *on average* in the long run. Therefore only the way the optimal fractional allocation is computed in this body of work is relevant to this dissertation, not the rounding methods. For additive utility functions, the optimal fractional allocations can be computed by solving one linear program for max-min [68], and a quadratic number of linear programs for leximin [120, 138].

Abraham et al. [3] studied a model of the DVD rental market, where clients (beneficiaries) provide a list of titles (ordered by preference) and each day receive a title from a fixed set of available titles. This is a long-term fairness problem (the goal is to provide fair treatment to multiple beneficiaries over multiple rounds), but there are important differences from the models studied here:

- The nature of the domain dictates that beneficiaries receive rewards only the first time a title is allocated to them. It follows that a particular assignment can (should) only be used once, and so there is an obvious upper bound on the time horizon (the authors use the term deadline). In my work the number of rounds is unrestricted, so using an action multiple times is not only acceptable, but sometimes necessary.

[4] A beneficiary has an additive (or linear) utility function if the utility it derives from any set of items is equal to the sum of utilities it derives from each of the items separately.

[5] The minimum makespan scheduling problem consists of distributing a number of jobs to a number of heterogeneous machines such as to minimize the makespan, i.e. the time until all jobs are finished. There is an obvious correspondence to the one-shot allocation problem with linear utility functions: the jobs are the items, the machines are the beneficiaries, and the time it takes a job to run on a certain machine corresponds to the utility of an item to a beneficiary. As for the goals, in the resource allocation case one has to maximize the smallest utility, while in the job allocation problem one has to minimize the largest workload. In spite of the overwhelming similarities, the best known algorithms and approximation bounds for the minimum makespan scheduling problem do not transfer to the resource allocation problem [17].

- Abraham et al. restrict the beneficiaries valuation of titles to a class called "universal ranking:" beneficiaries rank all titles, and the reward beneficiary b_1 gets from the j^{th} title on its list is equal to the reward any other beneficiary b_2 gets from the j^{th} title on its list. Conversely, I place no restrictions on beneficiaries' rewards.

- The multi-round problem is an online one (beneficiaries are allowed to make changes to their lists), and the authors' solution considers each round as a separate single-shot optimization problem. This greedy algorithm performs rather well on this problem (the authors show a small constant approximation ratio), but it would behave poorly on the problems in this dissertation, which are all offline. Actually, the central theme of this dissertation is "when there are multiple rounds, one could do better than repeatedly using the single-round solution."

2.5 Fairness in Stateful Domains

The large body of work in reinforcement learning [19, 44, 57, 79, 80, 127, 165] is concerned with optimizing the accumulation of one-dimensional rewards observed while traversing a state graph. In essence, estimates of the "goodness" of states (or state-action pairs) are iteratively built. The ideal goodness of a state s (or action a) is the expected value of the accumulation of rewards after visiting s (executing action a) followed by optimal decisions.

Such an approach cannot be applied verbatim to fairness-oriented goals, where the reward signal is multi-dimensional (each beneficiary gets its own reward). Note that (1) the optimal decision in state s is not only dependent on all rewards expected in the future, but also on those received in the past. (2) It follows, that the goodness of a state s is more accurately described by a Pareto-front of expected accumulation of rewards, and (3) such a Pareto-front may contain a prohibitively large number of points (exponential in the number of states [133]). Further more, (4) the Bellman principle does not hold: an optimal path may consists of suboptimal subpaths.

A different approach is taken by Peeters et al. [130–132] (extending [177, 178]) to use reinforcement learners for fairness-oriented goal. Each beneficiary is also a controller, and they act selfishly for predetermined periods of time, then compare accumulated rewards and the "richest" beneficiary/controller gives up his most lucrative action. As I already argued in Section 2.2, such an approach my achieve an long-term utility profile arbitrarily far from the optimum.

In [133] Perny and Spanjaard consider the problem of finding the best path a graph where there is uncertainty about the costs associated with the edges. Specifically, each edge has n possible costs, one for n possible scenarios (i.e. state the world can be in). The authors give the example of a driver that must decide on a route, knowing that the total travel time will depend on whether the traffic in some tunnel or bridge becomes gridlocked. The authors want to favor "robust" solutions (i.e. give larger weights to worse costs) so use OWA (a social welfare measure) to aggregate the n costs of a path. Formally, the goal is to find a path from a start state to one of the end states, path whose n costs are OWA-optimal. The authors use domain knowledge to speed-up MOA* (a multi-objective extension of A*).

While MOA* is guaranteed to find the optimal solution, it may require an exponential amounts of memory. In [134], Perny and Spanjaard propose near admissible multiobjective search algorithms to approximate (with performance guarantees) the set of Pareto optimal solution paths in a state space graph. These allow the authors to balance time/space requirements with solution quality.

The setup in [133,134] bears many similarities with the framework used in Chapter 5: the nodes of the graph correspond to states in my framework, and the n scenarios and the rewards associated with them correspond to beneficiaries and the rewards they get out of each action. As a consequence, edges may be used more than once.

There is an important difference: in [133,134] the graph has end states (e.g. an ambulance needs to reach one of several hospitals), while I assume no control over how long the process lasts. Therefore the authors look for paths with good costs for the n scenarios *when* an end

node is reached, while I look for infinite paths for which *all* values for the n beneficiaries are *always* close enough to good reward values.

The work in [6] focuses on finding infinite sequences of activities in a stateful domain. Furthermore, the authors focus on periodic sequences, which are relevant to my work since it produces periodic sequences whenever possible (see for instance the discussion in Sections 3.2 and 3.5). However, their goal is not fairness-oriented, and there is a specific implicit structure to the graph, so it is not immediately clear how the authors' insights and approximation algorithms could be relevant to my work.

2.6 Coordination

I end this Related work chapter with a multi-agent coordination discussion relevant to the multi-controller work in Chapter 6. Section 2.6.1 focuses on cooperative agents, and Section 2.6.2 discusses non-cooperative agents (e.g. controllers in charge of disjoint sets of beneficiaries).

2.6.1 The cooperative case

In the context of cooperative multi-agent systems, coordination means having the agents select individual actions that together make an optimal joint action. Boutilier [23] lists three solutions to the coordination problem: communication, social conventions, and learning.

Learning In stateless cooperative games with noisy rewards, Panait [125, 127] has the learners disregard early bad rewards, allowing them to settle on risky actions that only pay off when paired with the right cooperating action.

Communication Fisher et al [57] propose a formal theory of communication (based on interaction frames) between reinforcement learners. They also train hierarchical reinforcement learning agents to add communication actions to their plans. These actions start by

having no utility, but as the agents learn to communicate, these actions gain utility, the added value of their coordination.

Dowling et al [44] propose *Collaborative Reinforcement Learning*, where reinforcement learning agents exchange policy information on issues of common interest. The authors validate the paradigm on the problem of routing in mobile ad-hoc networks and also propose using it for macro-level urban traffic control. "Macro-level" means that the traffic light agents base their decisions on aggregate information, such as flow rates along streets, while my work on fairness demands instead that the agents make decisions based on information on individual cars (i.e. "micro-level").

For some cooperative games the outcomes don't always depend on the actions of all agents, but just a subset (usually based on spacial proximity). This bias translates into a *coordination graph*: only the agents connected by an edge need to coordinate their actions. This allows the original problem to be decomposed into local subproblems. Kok [83] computes the coordinated joint action when the model (i.e. pairwise-dependencies, outcomes of joint actions) is given, and also the problem of learning sequences of coordinated joint actions without prior knowledge of the underlying model. Kok provides an intuitive example from the soccer domain: when the ball is near your end of the field, only the goalie and defenders' actions matter, so only those actions need to be coordinated. The author's *max-plus* algorithm tractably produces an approximation of the optimal solution through payoff propagation using message exchange. For stateful problems Kok proposes *SparseQ*, where each agent builds utility tables for the coordination with each of its neighbors.

In [100] the authors propose the following task-allocation methodology for robots in a three-level hierarchy. A new task is assigned to the robot (agent on the bottom level) that performed the most similar task in the past. If the new task is significantly different from anything seen in the past, a *contract net* is used between the top levels of the hierarchy and an *acquaintance net* between the bottom levels of the hierarchy. The authors use a predator-prey application with different robots having different speeds and sensor ranges.

Although fairness is not part of the goal, I believe the methodology could be extended to include it.

2.6.2 The non-cooperative case

The issue of coordination has been studied extensively in game theory. A very common coordination game in the literature is the "Battle of the Sexes" game (already described in Section 2.2). This example further refines the reasons coordination is important: the game has two pure Nash equilibria and one in mixed strategies; the players can't agree on either of the pure ones, and the third one is wasteful, since both players flipping coins opens them up for occasional miscoordination.

One solution to this dilemma is the introduction of a coordination device: the players get private signals of a common random variable. A set of strategies using such a coordination device are called *correlated strategies*. If the set of strategies is stable (in the sense that no player has a reason to deviate if the other player follows his) they make a *correlated equilibrium*. For the previous examples, a correlated equilibrium consists of both players using the strategy "if the common coin toss resulted in heads, go to the ballet."

In a matrix game the set of correlated equilibria is a convex polytope [70], so it can be efficiently computed using linear programming. Papadimitriou [128] proposes a polynomial algorithm for finding some correlated equilibria in a large class of multi-player games of "succinct" representation, such as symmetric games, congestion games (only the number of players playing each action counts, not their identities) or graphical games (players are nodes in a graph and their utilities depend only on the choices of their neighbors).

Alternatively, a correlated equilibrium can be found through learning. In the context of Markov (stateful) games Greenwald and Hall [70] propose an algorithm that can be set to converge to the correlated equilibrium with the highest social utility (sum of rewards), minimax fairness (minimum over rewards), or maximin elitism (maximum over rewards). The algorithm is not distributed, but the issue can be addressed, to some degree, by having

the players "see" the others' actions and rewards, then emulate all the other players locally).

Of more interest is what such sets of correlated strategies can be found when the players are separate learning agents. Foster and Vohra [62] show that for most repeated games, the set of correlated equilibria coincides with the fixed points of the histories produced by players using learning rules calibrated to forecast the distributions over the joint action space of all the other players, and then they play myopic best-response to that prediction. The restriction over the nature of the learning forecasters is rather weak: they should be *correlated* and should break ties in a stationary, deterministic way. A learning forecaster is correlated if, at the limit, an event E happened a fraction p of the times the forecaster predicted E would happen with probability p.

Welfare Engineering Mechanism design is concerned with finding rules (i.e. designing mechanisms) for multi-agent interaction paradigms such as voting and auctions, such that various desirable outcomes emerge. A well known example is the Vickrey-Clarke-Groves (VCG) mechanism, where agents have no incentive to lie (so they don't have to waste time looking for a strategic deceit). In *welfare engineering* [34, 50, 51] (a special type of mechanism design), the goal is to find ways to ensure an optimal social welfare outcome emerges out of agents acting selfishly. For instance, [50] studies a negotiation protocol in the context of resource allocation. Note that in this framework (i.e. welfare engineering) the beneficiaries are also controllers, while in this dissertation they are separate classes of entities.

Chapter 3: Infinite-length Model

3.1 Context

In this chapter I present a formal analysis for a simple model I use as stepping stone for the rest of my work in this dissertation.

Consider the example from the introduction where one has to repeatedly assign two professors to teach two classes offered by their department each semester. One class is much harder than the other one, so during any single semester any one-to-one assignment is unfair to one of the professors. One fair solution would be for the professors to teach both classes together. But assuming this requires more overall effort than teaching the classes separately, this solution would be inefficient over the long run.

There is of course a better solution. If the two professors instead took turns teaching the hard class, then in the long run their average utilities would be *more fair* than in either of the one-to-one assignments and *more efficient* than in the sharing assignment. This is the rough idea behind *long-term fairness*: repeated interactions offer opportunities for improved efficiency and fairness over the single interaction scenario. But in general the solutions will not be as simple as alternating assignments (e.g. suppose I extended the previous example with multiple assignments, involving many professors and classes).

The research in this chapter examines the following framework: there are a number of *beneficiaries* (e.g. professors), which receive different *rewards* from each of a finite set of *actions*[1] (e.g. class assignments). I use the term *utility profile* to refer to the vector of utilities (one utility per beneficiary) that the beneficiaries derive from past actions. The actions are

[1] This framework assumes the actions are given; I will partially address this issue in Section 4.5 by introducing an algorithm for generating small sets of actions for resource (task) allocation problems satisfying certain conditions.

chosen with replacement, and prior actions do not restrict what actions can be chosen later, or their rewards. I define a beneficiary's *utility* as the average of all rewards it received in the past. As a first step, in this chapter I allow an infinite number of actions to be chosen. This last assumption (although less realistic) greatly facilitates both the theoretical analysis and the algorithmic approaches.

This framework, borrowed from [178], is very similar to the repeated normal-form game framework from game theory,[2] except there is a single *decision maker* that chooses actions for the good of all beneficiaries. This means that this is not actually a multi-agent problem as configured. I am ultimately interested in the game-theoretic aspects of the multi-agent (that is, multiple decision maker) case, but to do so, I must first understand the single decision maker case, and as it turns out even this is nontrivial. As the literature on the single decision-maker, multiple-beneficiary case has been relatively limited, I begin there.

3.2 Infinite Sequences

An action may give high rewards to some beneficiaries and low rewards to others, so sticking to just one action might be unfair to some beneficiaries, and thus leximin-undesirable. However, if beneficiaries receive different rewards from different actions, I may be able to improve all beneficiaries' reward averages by performing a combination of actions.

One approach to doing combinations is to use a periodic, repeated sequence of actions. In previous literature [178], distributed algorithms for discovering such sequences found suboptimal ones. I will show optimal solutions, albeit with non-distributed algorithms.

Because beneficiaries' utilities change with every new action being performed, it is not obvious how to compare arbitrary (possibly non-periodic) sequences. Periodic sequences are thus convenient because the resulting utility profiles always converge and one can compare periodic sequences by comparing the utility profiles they converge to.

[2] The example assigning professors to classes is a variant of the game "Battle of the Sexes" (also known as "Bach or Stravinsky"), without the miscoordinated joint-actions.

Periodic sequences are intuitively appealing, but even if one can find the periodic sequence with the best limit, that can still be suboptimal. I will show at the end of this section that in some problem instances with irrational coefficients there might exist infinite non-periodic sequences that achieve, at the limit, leximin-superior utility profiles to any utility profile achievable by a periodic sequence. My algorithms in Chapters 4 and 5 are guaranteed to produce sequences converging to the optimal utility profile, and, whenever possible, those sequences are periodic.

In this chapter I (1) argue that it suffices to focus on a specific set of "well-behaved" sequences (periodic or non-periodic) and (2) identify a subclass of those sequences with the leximin-optimal limit-point. In the following sections I propose additional requirements to impose on this class of sequences and then provide algorithms that produce sequences satisfying these requirements.

I postpone the leximin-specific discussion until Section 3.5 and provide as much of the theory as possible in a social welfare measure independent format (Sections 3.3 and 3.4).

3.3 Formal Framework

Let there be a set $\mathbb{A}$ of n_a actions affecting a set $\mathbb{B}$ of n_b beneficiaries through the reward functions $R_b : \mathbb{A} \to \mathbb{R}, \forall b \in \mathbb{B}$; let $\mathbb{S}$ be the set of infinite sequences of actions from $\mathbb{A}$:

$$\mathbb{S} = \{S = \langle S_1, S_2, \dots \rangle | \forall i \in \mathbb{N} : S_i \in \mathbb{A}\}$$

where I use the standard notations: $\mathbb{R}$, $\mathbb{Q}$, and $\mathbb{N}$ for the set of real, rational, and natural (positive integer) numbers, respectively.

Let $\mathbb{U}$ be the set of all possible utility profiles (vectors) achievable from following any sequence in $\mathbb{S}$ for any finite number of time steps:

$$\mathbb{U} = \{U \in \mathbb{R}^{n_b} | \exists t \in \mathbb{N}, \exists S \in \mathbb{S}, \forall b \in \mathbb{B} : U_b = \frac{1}{t} \sum_{j=1}^{t} R_b(S_j)\}.$$

Note that although utility profiles $U \in \mathbb{U}$ must be reached in *finite* time, there is no bound on the amount of time it takes to reach them. Therefore by following an infinite sequence of actions S, one jumps from an element of $\mathbb{U}$ to the another *forever*. Let $U(S)$ be the sequence of utility profiles visited when one chooses the actions in the infinite sequence of actions S. Note that *all* terms of $U(S)$ are contained in $\mathbb{U}$.

To make things easier, I refer to the elements of $\mathbb{A}$ as $\{1 \dots n_a\}$ and the elements of $\mathbb{B}$ as $\{1 \dots n_b\}$.

3.4 Classes of Sequences and Utility Profiles

Let $\mathbb{S}' \subset \mathbb{S}$ be the set of all sequences S where the *proportions* of the actions in $\mathbb{A}$ converge. Formally:

$$\mathbb{S}' = \{S \in \mathbb{S} | \forall a \in \mathbb{A} : \exists \lim_{t \to \infty} \frac{1}{t} k_a(S_{1:t})\}$$

where $S_{1:t}$ is the subsequence of S consisting of the first t elements and k is the *count* function (so $k_a(S_{1:t}) = |\{1 \leq i \leq t | S_i = a\}|$ is equal to the number of times action a is used in the first t positions of sequence S). I denote with $F^{\infty}(S)$ the vector of action proportions (or fractions) S settles on (i.e. $F_a^{\infty}(S) = \lim_{t \to \infty} \frac{1}{t} k_a(S_{1:t})$). Note that $F_a^{\infty}(S) \geq 0$ ($\forall a \in \mathbb{A}$) and $\sum_{a=1}^{n_a} F_a^{\infty}(S) = lim_{t \to \infty} [\frac{1}{t} \sum_{a=1}^{n_a} k_a(S_{1:t})] = lim_{t \to \infty} \frac{t}{t} = 1$.

Let $U(S_{1:t})$ be the vector of utilities achieved after following the first t steps of sequence S (i.e. U's component for beneficiary b is $U_b(S_{1:t}) = \frac{1}{t} \sum_{i=1}^{t} R_b(S_i)$). Because the action proportions converge, the sequence of utility vectors $U(S) = \langle U(S_{1:1}), U(S_{1:2}), \dots \rangle$ also converges component by component; I make the following notation $U_b^{\infty}(S) = \lim_{t \to \infty} U_b(S_{1:t})$.

$$\forall S \in \mathbb{S}' : \quad U_b^{\infty}(S) = \sum_{a=1}^{n_a} F_a^{\infty}(S) R_b(a). \tag{3.1}$$

As a side note, I point out that there might exist sequences $S \in \mathbb{S} - \mathbb{S}'$ such that the sequence $U(S)$ converges (i.e. it is possible that $U^{\infty}(S)$ exists although $F^{\infty}(S)$ does not). This might happen for instance if one "clones" action a (i.e. adds a new action a' with identical rewards), then replaces some occurrences of action a in some sequence $S' \in \mathbb{S}'$ with a'.

Let $\mathbb{U}'$ be the set of utility vectors achievable by sequences in $\mathbb{S}'$ in any finite number of time steps. Let $\mathbb{U}''$ be the set of utility vectors achievable, at the limit, by sequences in $\mathbb{S}'$. Also, let $\mathbb{U}'''$ be the set of all limit points of sequences $U(S)$, regardless of whether $S \in \mathbb{S}'$ or $S \in \mathbb{S} - \mathbb{S}'$. Lastly, let $\mathbb{H}$ denote the set of all linear combinations of actions' rewards, i.e. the convex hull of the set $\{U | \exists a \in \mathbb{A}, \forall b \in \mathbb{B} : U_b = R_b(a)\}$. To restate:

$$\mathbb{U} = \{U | \exists S \in \mathbb{S}, \exists t \in \mathbb{N} : U = U(S_{1:t})\} \tag{3.2}$$

$$\mathbb{U}' = \{U | \exists S \in \mathbb{S}', \exists t \in \mathbb{N} : U = U(S_{1:t})\} \tag{3.3}$$

$$\mathbb{U}'' = \{U | \exists S \in \mathbb{S}' : U = U^{\infty}(S)\} \tag{3.4}$$

$$\mathbb{U}''' = \{U | \exists S \in \mathbb{S} : U = U^{\infty}(S)\} \tag{3.5}$$

$$\mathbb{H} = \{U | \exists w_1, \ldots, w_{n_a} \geq 0 : \left(\forall b \in \mathbb{B} : U_b = \sum_{a=1}^{n_a} R_b(a) w_a \right) \wedge \sum_{a=1}^{n_a} w_a = 1\}. \tag{3.6}$$

I will show that $\mathbb{U} = \mathbb{U}' \subseteq \mathbb{U}'' = \mathbb{U}''' = \mathbb{H}$, which will allow me to argue in favor of sequences in $\mathbb{S}'$ since they achieve (in finite time or at the limit) all utility profiles any other sequence in $\mathbb{S}$ achieves (in finite time or at the limit). Next, I will restrict my attention to a subset of $\mathbb{S}'$ consisting of sequences with optimal limit points (see Section 3.5 for leximin

and Section 4.3.4 for some other social welfare measures), and then restrict that further by imposing additional properties (Section 4.1).

Lemma 1. $\mathbb{U} = \mathbb{U}'$.

Proof. Obviously $\mathbb{U}' \subseteq \mathbb{U}$ since $\mathbb{S}' \subset \mathbb{S}$. Also, $\mathbb{U} \subseteq \mathbb{U}'$ because $\forall U \in \mathbb{U}$ there must exist $S \in \mathbb{S}$ and $\tau \in \mathbb{N}$ such that $U = U(S_{1:\tau})$, and based on S and τ it is trival to build a sequence $S' \in \mathbb{S}'$ that achieves U (e.g. $S'_t = S_{(t-1) \pmod{\tau}+1}$). The result follows. □

Lemma 2. $\mathbb{U} \subseteq \mathbb{U}''$

Proof. Let U be an arbitrary utility profile in $\mathbb{U}$; U is achieved by some sequence $S \in \mathbb{S}$ at time step $\tau \in \mathbb{N}$: $U = U(S_{1:\tau})$. Let $S' \in \mathbb{S}$ be the periodic sequence such that $S'_t = S_{(t-1) \pmod{\tau}+1}$. Because S' is periodic, $\lim_{t\to\infty} \frac{1}{t} k_a(S'_{1:t}) = \frac{1}{\tau} k_a(S_{1:\tau})\ \forall a \in \mathbb{A}$, so $S' \in \mathbb{S}'$. It follows that $U^\infty(S') = U$, so $U \in \mathbb{U}''$, proving that $\mathbb{U} \subseteq \mathbb{U}''$. □

Lemma 3. $\mathbb{U}'' = \mathbb{U}''' = \mathbb{H}$

Proof. Obviously, $\mathbb{U}'' \subseteq \mathbb{U}'''$ since $\mathbb{S}' \subset \mathbb{S}$.

Next I show that $\mathbb{U}''' \subseteq \mathbb{H}$. Let $S \in \mathbb{S}$ be an arbitrary sequence with the property that the sequence $U(S)$ converges. All terms of the sequence $U(S)$ belong to $\mathbb{H}$ (it is enough to pick $w_a = k_a(S_{1:\tau})/\tau$ to prove that $U(S_{1:t}) \in \mathbb{H}$). Since $\mathbb{H}$ contains all its limit points (any convex hull is closed), it follows that $U^\infty(S) \in \mathbb{H}$. Therefore $\mathbb{U}''' \subseteq \mathbb{H}$, since I proved $U^\infty(S) \in \mathbb{U}'''$ implies $U^\infty(S) \in \mathbb{H}$.

Note that the weights $w_1, \ldots, w_{n_a}$ in Equation 3.6 constitute the coordinates of an arbitrary point in the n_a-dimensional unit simplex; moreover $\forall S \in \mathbb{S}'$, $F^\infty(S)$ also belongs to the n_a-dimensional unit simplex. I will show in Theorem A.3 (Appendix A) that the algorithms I propose can produce sequences $S \in \mathbb{S}'$ with $F^\infty(S)$ equal to any point in the n_a-dimensional unit simplex. This is a constructive proof that $\mathbb{H} \subseteq \mathbb{U}''$.

So far I proved that $\mathbb{U}'' \subseteq \mathbb{U}''' \subseteq \mathbb{H} \subseteq \mathbb{U}''$, so $\mathbb{U}'' = \mathbb{U}''' = \mathbb{H}$. □

My purpose for $\mathbb{H}$ is three-fold. The first one is topological: in Lemma 3, I used the fact that $\mathbb{H}$ is closed to that show no sequence can converge to a point outside $\mathbb{H}$. The second (more important) reason is geometrical: $\mathbb{H}$ paints a very intuitive picture of what $\mathbb{U}''$ looks like. Lastly, the connection between $\mathbb{H}$ and the n_a-dimensional unit simplex (Equation 3.6) can be used to express the distinction between $\mathbb{U}$ and $\mathbb{U}'' - \mathbb{U}$, as seen in Lemma 4.

Lemma 4. For an arbitrary $U \in \mathbb{U}''$, $U \in \mathbb{U}$ if and only if there exists a point F in the n_a-dimensional unit simplex such that:

1. $\forall b \in \mathbb{B} : \sum_{a=1}^{n_a} R_b(a) F_a = U_b$, and

2. the coordinates of F are all rational (i.e. $\forall a \in \mathbb{A} : F_a \in \mathbb{Q}$).

Proof. Lemma 3 guarantees there exists a point F in the n_a-dimensional unit simplex satisfying the first condition for any $U \in \mathbb{U}'' = \mathbb{H}$, so the second condition is the only point of contention.

To prove the IF part, let $q_a \in \mathbb{N} \cup \{0\}$ and $p_a \in \mathbb{N}$ such that $F_a = \frac{q_a}{p_a}$ (since $F_a \in \mathbb{Q}$). Let P the *least common multiple* of the p values, and let $c_a = \frac{q_a}{p_a} \times P$. Since $c_a \in \mathbb{N} \cup \{0\}$, it follows that U can be achieved in exactly P time steps (e.g. use the first action c_1 times, then the second action c_2 times, etc.), so $U \in \mathbb{U}$.

The ONLY-IF part follows, since $\forall U \in \mathbb{U}$, $\exists S \in \mathbb{S}$ and $\exists \tau \in \mathbb{N}$ such that $U(S_{1:\tau}) = U$; it suffices to pick $F_a = \frac{k_a(S_{1:\tau})}{\tau}$ to guaranteed F is inside the n_a-dimensional unit simplex and both condition (1) and (2) are satisfied. □

Lemma 4 proves necessary and sufficient conditions for the points in $\mathbb{U}'' - \mathbb{U}$. Note that $\mathbb{U}'' - \mathbb{U} \neq \emptyset$ when $n_a > 1$.

The results in Lemma 1, Lemma 2 and Lemma 3 allow me to focus on the sequences in $\mathbb{S}'$ and compare them based on the utility profile they converge to.

3.5 Leximin Social Welfare

All the discussion so far was independent of one's choice of fairness measure (or more generally, social welfare). In this section I will present the specifics of using *leximin* to compare utility profiles.

Problem 1 (BASE PROBLEM)**.** Given a set $\mathbb{A}$ of n_a actions affecting a set $\mathbb{B}$ of n_b beneficiaries through the reward functions $R_b : \mathbb{A} \to \mathbb{R}, \forall b \in \mathbb{B}$, find the leximin optimal utility profile in $\mathbb{U}''$.

I use the n_a-dimensional unit simplex as a parameter space to search for $\max(\mathbb{U}'')$, the leximin optimal utility profile in $\mathbb{U}''$. Once reformulated as an optimization problem over a compact and convex set, the problem can be solved with the algorithm proposed in [138]. As a computational aside, the algorithm consists of solving $O(n_a^2)$ linear programs (LPs), but approximate results based on a floating-point fixed-length representation might be required to make sure the complexity of this algorithm does not dominate that of the algorithms I propose in this dissertation.

The algorithm produces $U^\star = \max(\mathbb{U}'')$, the unique [47] leximin-optimal utility vector, and $F^\star$, a point (not necessarily unique) inside the unit simplex such that:

$$\forall b \in \mathbb{B} : \sum_{a=1}^{n_a} R_b(a) \times F_a^\star = U_b^\star. \tag{3.7}$$

I emphasize that $U^\star$ is the single best utility profile any sequence can achieve in finite time or in the limit.

Example 3.1. Consider the professor-assignment example from the introduction: the first action corresponds to the assignment where the first professor teaches the easy class, the second action corresponds to the assignment where the second professor teaches the easy class, and the third action is the assignment where they teach the classes together:

		Beneficiaries	
		b_1	b_2
	a_1	10	0
Actions	a_2	0	10
	a_3	1	1

In this example $\mathbb{U}''' = \text{convex-hull}\{U_1, U_2, U_3\}$, where $U_1 = [10, 0]$, $U_2 = [0, 10]$, and $U_3 = [1, 1]$. $\mathbb{U} = \mathbb{U}''' \cap \mathbb{Q}^{n_b} = \text{convex-hull}\{U_1, U_2, U_3\} \cap \mathbb{Q}^2$. Furthermore, $U^\star = [5, 5]$, which can only be achieved through $F^\star = [0.5, 0.5, 0]$.

Example 3.2. Consider a variation of Example 3.1 where one of the professors actually gets a reward of $10\sqrt{2}$ from teaching the easy class:

		Beneficiaries	
		b_1	b_2
	a_1	10	0
Actions	a_2	0	$10\sqrt{2}$
	a_3	1	1

In this case $U^\star = [\frac{10\sqrt{2}}{1+\sqrt{2}}, \frac{10\sqrt{2}}{1+\sqrt{2}}]$ and there is a unique $F^\star = [\frac{\sqrt{2}}{1+\sqrt{2}}, \frac{1}{1+\sqrt{2}}, 0]$.

I revisit the existence of a periodic sequence for the optimal utility profile solution $U^\star$. I claim that having all rewards rational is a sufficient condition for $U^\star \in \mathbb{U}$, which guarantees $U^\star$ can be achieved over and over by a periodic sequence. Consider that $F^\star$ can be computed with a finite number of arithmetic operations (e.g. by using the Simplex algorithm to handle the linear programming calls in the algorithm computing $U^\star$ and $F^\star$), and since the input values (rewards) are rational, so must be the output values (i.e. $F^\star$). Since the $F^\star$ I get has all rational coefficients, the claim follows from Lemma 4. Irrational rewards could mean (but not necessarily), that there is no $F^\star$ with rational coefficients, in which case there is no optimal periodic sequence.

I make the observation that even when all components in $F^\star$ are rational, it may take a large number of time steps to achieve $U^\star$. If $F^\star_a = \frac{q_a}{p_a}$ (in reduced form), then the smallest period is P, the least common multiple of the p values. Small changes in $F^\star$ can produce very large changes in the period size and the actions' *multiplicities* (i.e. the number of times each action is used in one period). In Example 3.1 $F^\star = [0.5, 0.5, 0]$, so $P = 2$ (and the first two actions have a multiplicity of 1); however an $F^\star = [0.45, 0.55, 0]$ implies $P = 20$ (first action has multiplicity 9, the second 11); while an $F^\star = [0.49, 0.51, 0]$ leads to $P = 100$ (and multiplicities of 49 and 51 for the first two actions respectively).

Chapter 4: Finite Time Horizon

In this chapter I study the finitely-repeated game of unknown length. This is relevant for applications such as assigning classes to professors: professor b's average reward will oscillate around $U_b^\star$, and he prefers to retire when his average reward is above rather than bellow $U_b^\star$. Other examples include assigning shifts to nurses, and delivery routes to postal workers. In this chapter I argue for a solution that will keep all negative deviations from $U_b^\star$ small, so no beneficiary can lose too much. Intuitively, this should help professor b_1 (from Example 3.1) trust professor b_2 not to back out of the deal right after b_1 taught the hard class.

This model starts from the infinitely-repeated game model in Chapter 3 and formulates additional requirements to a solution's transient phase. In Section 4.1, I propose a formal problem, and prove that one cannot be expected to find optimal solutions in this case. I then propose a family of approximation algorithms (Section 4.2), and discuss their performance (both worst case and empirical results), and possible variations (Section 4.3). I also compare my algorithms against others from the literature (Section 4.4). In Section 4.5, I optimize the approach introduced in this chapter to a specific subclass of resource (task) allocation problems.

4.1 Formulating the Problem

Solving the BASE PROBLEM provides $U^\star$, an upper bound over the set of achievable utility profiles. If the game is guaranteed to last forever, one could be satisfied with the goal of finding a sequence that achieves $U^\star$ at the limit. But in more practical applications, one must consider the implications of having the process end after a finite number of steps, or

more generally, having the length of the sequence of actions drawn from some probability distribution.

Example 4.1. I continue the professor-assignment Example 3.1, where $U^\star = [5, 5]$ coincides with the optimal solution if the game lasts for an even number of steps: simply use actions 1 and 2 in equal proportions. However, action 3 is leximin-optimal if the game lasts for only one round. This shows that the optimal sequence of decisions depends on the duration of the game, and so if the duration is not known in advance, some sort of tradeoff might be required.

A Risk-Averse Approach Given that *leximin* is a *risk-averse* fairness concept ("no one is left behind"), it makes sense to focus on a *risk-averse* solution approach: minimize the largest amount a beneficiary risks losing due to the game ending prematurely.[1]

I define the *windfall* of beneficiary b at time step t while executing sequence S as the difference between the rewards accumulated by beneficiary b during the first t steps, and the amount he was entitled to, which is $t \times U_b^\star$.

$$\text{Windfall}_b(S, t) = \sum_{i=1}^{t} R_b(S_i) - t \times U_b^\star. \tag{4.1}$$

This difference is the benefit beneficiary b would get over what it was entitled to if the process stopped right after time step t. If the value is positive than it is a unearned gain (hence the term windfall); if the value is negative, then it is a loss.

[1] I note two other obvious approaches. First, assuming the agents were risk-neutral rather than risk-averse, I could choose an action stochastically using the values in $F^\star$ as probabilities. This approach produces leximin-optimal *expected rewards* (equal to $U^\star$). However, this provides a poor sort of guarantee usually referred to in the economics literature as *ex-ante* fairness, where past results are disregarded in making future decisions. For example, if beneficiary b loses to beneficiary b' a hundred times by sheer luck, the algorithm will not try to be kinder to b' in the future.

Second, I might try to deterministically find a sequence which leximin-optimizes the beneficiaries' *expected utilities* weighted by the probabilities in the distribution of game lengths. For instance in Example 3.1, the first approach would always randomly pick action 1 or action 2: but given a high enough probability of the game ending early, the second approach could select action 3. However, while I do not have an algorithm to suggest for the second approach, I expect that it would be increasingly expensive with longer action sequences.

I use the term *worst loss* (WL) of a sequence S to mean a lower bound on all windfall values, over all beneficiaries (i.e. $\text{WL} \leq \text{Windfall}_b(S,t)$, $\forall t \in \mathbb{N}$ and $\forall b \in \mathbb{B}$). Note that not every sequence has a constant WL.[2] When it exists, $\text{WL} \leq 0$, and $|\,\text{WL}\,|$ is an upper bound on the largest amount that any beneficiary can lose.

My proposed solution is particularly suitable to scenarios with very large game durations, because my approach is insensitive to the distribution of game durations. Furthermore, the existence of a lower-bound for all windfall values is enough to guarantee the utilities converge to $U^\star$ as $t \to \infty$ (see Theorem A.2 in Appendix A).

Note that the windfalls of all beneficiaries cannot be simultaneously non-negative, unless they are all zero. Otherwise one could use the sequence of actions used up to that point to build an infinite sequence with a utility vector leximin-superior to the optimal vector $U^\star$. Thus there must exist strictly negative windfalls during the implementation of any non-empty sequence. The only exception would be if there is some action a such that $\forall b : R_b(a) = U^\star_b$, but that case is trivial: simply play action a at each step, with 0 windfall for each beneficiary.

For convenience, I define for each action j a vector $X_j = [X_{j,1} \dots X_{j,n_b}]$, where $X_{j,b} = R_b(j) - U^\star_b$, the amount action j changes the windfall of beneficiary b. Here are the equivalents of Equation 3.7 and Equation 4.1 using the X notation:

$$\begin{aligned}\sum_{a=1}^{n_a} X_{a,b} \times F^\star_a &= \sum_{a=1}^{n_a} R_b(a) \times F^\star_a - U^\star_b \sum_{a=1}^{n_a} F^\star_a \\ &= U^\star_b - U^\star_b \\ &= 0 \end{aligned} \tag{4.2}$$

[2] I show two sequences with WL $= -\infty$. In Example 3.1 if $S = \langle 1, 1, \dots \rangle$, then $\text{Windfall}_{b_2}(S,t) = -5t$, so $lim_{t\to\infty}\text{Windfall}_{b_2}(S,t) = -\infty$. The other example is $S' = \langle 1,2,1,1,2,2,\dots,(1)^{2^j},(2)^{2^j},(1)^{2^{j+1}},(2)^{2^{j+1}},\dots\rangle$. Although $U(S')$ reaches $U^\star$ infinitely often, I can show that $\text{Windfall}_{b_2}(S, 3\times 2^j - 2) = -5(2^{j+1}-1)$, so $lim_{j\to\infty}\text{Windfall}_{b_2}(S, 3\times 2^j - 2) = -\infty$.

$$\text{Windfall}_b(S,t) = \sum_{i=1}^{t} [R_b(S_i) - U_b^\star]$$

$$= \sum_{i=1}^{t} X_{S_i,b} \tag{4.3}$$

$$= \sum_{a=1}^{n_a} k_a(S_{1:t}) X_{a,b}. \tag{4.4}$$

I also make these two simplifying notations:

$$X_{a,b}^{+} = \max(X_{a,b}, 0) \tag{4.5}$$

$$X_{a,b}^{-} = \min(X_{a,b}, 0) \tag{4.6}$$

Example 4.2. I illustrate the concept of windfall with the professor-assignment running-example (continued from Example 4.1). The X vectors are: $X_1 = [5, -5]$, $X_2 = [-5, 5]$, and $X_3 = [-4, -4]$. I compare the following periodic action sequences (see Figure 4.1):

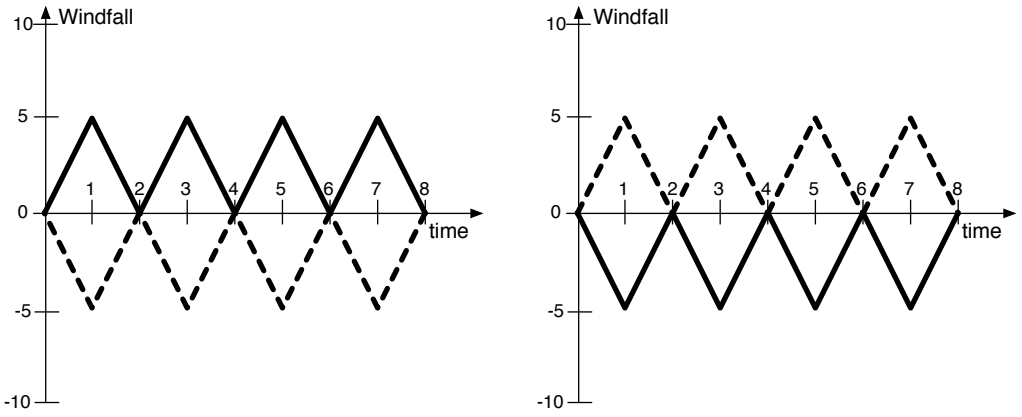
$$S^{(a)} = \langle 1, 2, 1, 2, \dots \rangle$$

$$S^{(b)} = \langle 2, 1, 2, 1, \dots \rangle$$

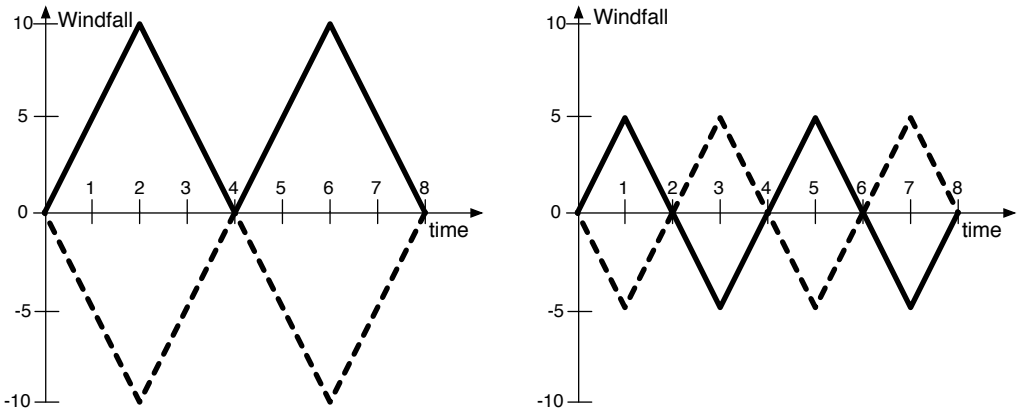
$$S^{(c)} = \langle 1, 1, 2, 2, 1, 1, 2, 2, \dots \rangle$$

$$S^{(d)} = \langle 1, 2, 2, 1, 1, 2, 2, 1, \dots \rangle$$

Sequence $S^{(a)}$ makes the first beneficiary's windfall equal to 5 on odd steps and 0 on even steps, while the second beneficiary's windfall is 0 on odd steps and -5 on even ones. Therefore the second beneficiary risks coming up 5 units short if the game stops after an odd number of steps and breaks even otherwise. Sequence $S^{(b)}$ has identical effects as $S^{(a)}$,



(a) $S^{(a)} = \langle (1,2)^* \rangle$ (b) $S^{(b)} = \langle (2,1)^* \rangle$



(c) $S^{(c)} = \langle (1,1,2,2)^* \rangle$ (d) $S^{(d)} = \langle (1,2,2,1)^* \rangle$

Figure 4.1: Windfalls as functions of time for the two beneficiaries in Example 3.1 (the solid line for the first and the dotted line for the second) as produced by various periodic sequences.

but to different beneficiaries, so the two are equally good. Using sequence $S^{(c)}$, the second beneficiary risks losing as much as 10 units, so I prefer $S^{(a)}$ (or $S^{(b)}$) to $S^{(c)}$.

Note that using sequence $S^{(a)}$ exposes the second beneficiary losses (negative windfall), while the first beneficiary is only exposed to gains (positive windfall). Although sequence $S^{(a)}$ is as fair as possible with respect to average utilities, one cannot help but notice a "second degree" unfairness; the sequence $S^{(d)}$ avoids this by alternating which beneficiary risks negative windfalls. This leads to the following paradox: although intuitively one

prefers $S^{(d)}$ to $S^{(a)}$ or $S^{(b)}$, the profile of worst losses for all beneficiaries during sequences $S^{(a)}$ or $S^{(b)}$ (i.e. -5 and 0) are leximin superior to the worst loss profile during $S^{(d)}$ (i.e. -5 and -5).

This example suggests that one might consider optimizing only the "min" of the worst loss profile, instead of leximin optimizing all beneficiaries' worst losses. However, before focusing on *infinite* sequences, I prove that finding *finite* sequences (of given length) with optimal WL is NP-hard (Theorem A.1, Appendix A):

Problem 2 (WORST LOSS OPTIMIZATION PROBLEM)**.** Given the setup in the BASE PROBLEM, and $\lambda \in \mathbb{N}$ such that $\lambda > |\mathbb{A}|$, find a sequence of actions S of length λ with an optimal *worst loss*.

4.2 Algorithms

As a consequence of Theorem A.1, the best one can hope for is approximation algorithms. As part of the motivation for my algorithms, I start by showing how the most obvious heuristic produces unacceptable results. Next, I present an alternate heuristic, and then introduce a family of algorithms based on it. In Section refsection:choosing:thetas I show a way to choose from this family the algorithm with the best WL bound for a given problem. Finally, I present an approximation ratio for the entire family of algorithms.

The most intuitive solution to optimizing worst losses(s) is to try to keep all windfalls as large as possible at all times, i.e. greedily choose the action that leximin optimizes windfalls during the next step (choose the action with the best immediate effects). I will refer to this strategy as **GW** (Greedy leximin-optimizing next-step Windfalls). One weakness of this approach is that it assumes the game ends at the next step. In Example 3.1 it will always choose action 3, leading to arbitrarily bad windfalls for both beneficiaries if the game lasts long enough. Note that $F_3^\star = 0$, meaning action 3 should not be played at all. One can easily extend the greedy heuristic to ignore the "unusable" actions, but I will show that it is not

enough to prevent windfalls from getting arbitrarily bad.

Example 4.3. I will now introduce a slightly less simplistic example, which I will use for illustration in the rest of the paper:

		Beneficiaries		
		b_1	b_2	b_3
Actions	a_1	-20	30	-20
	a_2	60	30	-40
	a_3	-56	-60	64

In this case $U^\star = [0, 0, 0]$ (so the rewards and the X values coincide) and there is a unique $F^\star = [\frac{4}{15}, \frac{6}{15}, \frac{5}{15}]$. For this problem **GW** will choose action a_1 repeatedly, leading to arbitrarily bad windfalls for the first and last beneficiaries.[3] The reason is that actions a_2 and a_3 hurt one of those beneficiaries more than action a_1 hurts any of them, and **GW** lacks the look-ahead to see the benefits of using actions a_2 and a_3.

The algorithms I propose are based on the following observation. All beneficiaries' windfalls are zero at time t if the number of times each action j was used up to time t are all proportional to the corresponding components of some $F^\star$ vector (i.e. $k_j(S_{1:t}) = F^\star_j \times t$, $\forall j$). Ideally, all $k_j(S_{1:t}) = F^\star_j \times t$ at all times, but since the k_j functions only take integer values, that is not always possible. It is intuitive that the windfalls at time t' cannot get very bad if the relative counts $k_j(S_{1:t'})/t'$ for how much each action was used up to time t' are close enough to the corresponding values in $F^\star$. There are many ways one can construct sequences S to keep the $k_j(S_{1:t})$ values close enough to the $F^\star_j \times t$ values; I previously introduced two such methods [10]. I include them for completeness, then I introduce a parameterized family that generalizes over both those methods.

[3] **GW** always chooses action a_1 in Example 4.3, so this method actually gets stuck at the suboptimal utility profile $U = [-20, 30, -20]$.

I first note that all my methods completely ignore the actions j which are not used in $F^\star$ (i.e. $F^\star_j = 0$). Let $n^\star_a$ be the number of actions used in $F^\star$; for simplicity, I rename the actions (reorder the dimensions of the simplex) such that all actions used in $F^\star$ come before the other ones: $\forall j \in \{1 \dots n^\star_a\} : F^\star_j > 0$ and $\forall j \in \{n^\star_a + 1 \dots n_a\} : F^\star_j = 0$.

Method 1 from [10] This method chooses an action that, so far, has been used the least relative to how often the action should have been used. Consequently, an action j can be chosen at time $t+1$ if it has the smallest ratio $\frac{k_j(D_{1:t})}{F^\star_j \times t}$, where D is the sequence of decisions produced by this method. Without affecting the decision process, one can eliminate t from the denominator. I formalize this method as follows:

$$D_t \in \left\{ j \leq n^\star_a \middle| \forall i \leq n^\star_a : \frac{k_j(D_{1:t-1})}{F^\star_j} \leq \frac{k_i(D_{1:t-1})}{F^\star_i} \right\} \tag{4.7}$$

where $k_j(\langle\rangle) = 0 \; \forall j \in \{1 \dots n^\star_a\}$, and $\langle\rangle$ is the empty sequence. Note that multiple actions could tie for the minimum.

Method 2 from [10] While the previous approach helps less-often chosen actions catch up to the others, this approach chooses actions that — if used — will get the least ahead of the others. The sequence of decisions D' for this method satisfies:

$$D'_t \in \left\{ j \leq n^\star_a \middle| \forall i \leq n^\star_a : \frac{k_j(D'_{1:t-1}) + 1}{F^\star_j} \leq \frac{k_i(D'_{1:t-1}) + 1}{F^\star_i} \right\}. \tag{4.8}$$

Generalized Method This method associates a constant θ_j to each action j. An action j can be chosen at time $t+1$ if it has the smallest score $\frac{k_j(D''_{1:t}) + \theta_j}{F^\star_j \times t}$, where D'' is the sequence of

decisions produced by this method.

$$D''_t \in \left\{ j \leq n^\star_a \middle| \forall i \leq n^\star_a : \frac{k_j(D''_{1:t-1}) + \theta_j}{F^\star_j} \leq \frac{k_i(D''_{1:t-1}) + \theta_i}{F^\star_i} \right\} \tag{4.9}$$

For convenience, let θ denote the collection of θ_j values for each action j. Note that this coincides with the first method when $\theta = [0, \ldots, 0]$, and it coincides with the second method when $\theta = [1, \ldots, 1]$. As a result, I will refer to the generalized method as $\mathbf{GF}^{\boldsymbol{\theta}}$, and I will refer to the two special cases as $\mathbf{GF}^{\mathbf{0}}$ and $\mathbf{GF}^{\mathbf{1}}$, respectively. I use the name **GF** because these methods *greedily* optimize actions' usage *frequencies* (relative counts) relative to some optimal configuration $F^\star$; this naming is consistent with **GW**, which greedily leximin-optimizes next-step windfalls.

An intuitive view on the effect of the constants in θ is that they allow one to offset the occurrences of an action relative to the occurrences of some other actions with no impact on $U^\star$ or $F^\star$.

So far I have shown how the $\mathbf{GF}^{\boldsymbol{\theta}}$ algorithm decides what action to execute at each time step, but not how to choose the θ to seed the $\mathbf{GF}^{\boldsymbol{\theta}}$ algorithm. Next, I will establish analytical bounds on WL as a function of θ, and then I will show how to pick θ using linear programs (Section 4.2.1) such as to optimize those bounds. Alternatively, I show faster ways to seed the algorithm, at the expense of WL (Section 4.2.1 and Section 4.3.1).

I use the following result to derive analytical bounds on WL for $\mathbf{GF}^{\boldsymbol{\theta}}$:

Theorem 4.1. For any arbitrary functions $C : \mathbb{N} \to \mathbb{R}$ and $\Delta_l, \Delta_h : \mathbb{A} \to \mathbb{R}$ satisfying:

$$\forall t \in \mathbb{N}, \forall a \in \{1 \ldots n^\star_a\} : \Delta_l(a) \leq k_a(D''_{1:t}) - C(t)F^\star_a \leq \Delta_h(a), \tag{4.10}$$

all $\text{Windfall}_b(D'', t)$ values are bounded as follows ($\forall t \in \mathbb{N}$ and $\forall b \in \mathbb{B}$):

$$\sum_{a=1}^{n_a^\star} \Delta_l(a) X_{a,b}^+ + \sum_{a=1}^{n_a^\star} \Delta_h(a) X_{a,b}^- \leq \text{Windfall}_b(D'', t) \leq \sum_{a=1}^{n_a^\star} \Delta_h(a) X_{a,b}^+ + \sum_{a=1}^{n_a^\star} \Delta_l(a) X_{a,b}^- . \quad (4.11)$$

Proof. I start with $\text{Windfall}_b(D'', t) = \sum_{i=1}^{n_a^\star} k_i(D''_{1:t}) \times X_{i,b}$ (Equation 4.4) and subtract $0 = \sum_{i=1}^{n_a^\star} F_i^\star \times X_{i,b}$ (Equation 4.2) multiplied by $C(t)$. I get:

$$\begin{aligned} \text{Windfall}_b(D'', t) &= \left(\sum_{i=1}^{n_a^\star} k_i(D''_{1:t}) \times X_{i,b} \right) - C(t) \times \left(\sum_{i=1}^{n_a^\star} F_i^\star \times X_{i,b} \right) \\ &= \sum_{i=1}^{n_a^\star} \left(k_i(D''_{1:t}) - F_i^\star \times C(t) \right) \times X_{i,b}. \end{aligned} \quad (4.12)$$

The theorem follows by replacing the weights of the X values in the previous equation with their bounds from Equation 4.10. Specifically, I lower bound a beneficiary's windfalls by using smallest weights (i.e. $\Delta_l(a)$) for positive X values and largest weights (i.e. $\Delta_h(a)$) for negative X values. The converse holds for the upper bound. □

Given a triplet of functions C, Δ_l and Δ_h satisfying Equation 4.10, one can compute the lower bounds for Equation 4.12 for all beneficiaries, and then use the smallest of these values as WL. Finding an optimal triplet of functions C, Δ_l and Δ_h (with respect to the WL it induces) is still an open problem; instead, I investigate a number of C functions. For each such C function I derive corresponding Δ_l and Δ_h functions (Appendix A.2), then compute a θ that optimizes WL based on the windfall lower bounds from Theorem 4.1. While iterating over the set of C functions, one keeps track of the largest WL so far and the θ used to achieve it.

Before deciding what C functions to investigate, I make the following two observations:

(1) $\lim_{t\to\infty} \frac{C(t)}{t} = 1$,[4] and (2) one cannot improve the WL bounds of Theorem 4.1 by simply translating the C function by some constant.[5] The most obvious choice is $C(t) = t$ (specially since the original motivation for my methods was to keep $\frac{k_a}{F^\star_a}$ as close to t as possible). I also investigate: $C(t) = \frac{k_a(D''_{1:t})}{F^\star_a}$, $C(t) = \min_a \frac{k_a(D''_{1:t})}{F^\star_a}$, $C(t) = \max_a \frac{k_a(D''_{1:t})}{F^\star_a}$, $C(t) = \min_a \frac{k_a(D''_{1:t})+\theta_a}{F^\star_a}$, and $C(t) = \max_a \frac{k_a(D''_{1:t})+\theta_a}{F^\star_a}$.

In order to get simpler closed-form Δ_l and Δ_h bounds I require that:

$$\forall j \in \mathbb{A} : \ \theta_j \in [0, 1]. \tag{4.13}$$

I will make it clear shortly that I only need these restrictions to ensure the windfall bounds I derive hold before all "usable" actions have been used at least once. Suppose one were to be able to guarantee that the game will not end for a large enough number of time steps that all actions (or some subset of interest) are used at least once under the $\mathbf{GF}^{\boldsymbol{\theta}}$ policy. In that case one could relax all (or a subset) of these restrictions, as the resulting violations of windfall bounds would be irrelevant, since the game cannot end when the WL bounds are violated. I leave the pursuit of this endeavor as future work; Equation 4.13 will hold implicitly for the rest of this dissertation.

Lemma 5. If Equation 4.13 holds, then:

$$\forall i, j \in \{1 \dots n^\star_a\} \text{ and } \forall t \in \mathbb{N} : \frac{k_i(D''_{1:t}) + \theta_i - 1}{F^\star_i} \leq \frac{k_j(D''_{1:t}) + \theta_j}{F^\star_j}.$$

Proof. If $k_i(D''_{1:t}) = 0$, then the inequality holds since the left hand side is non-positive (since

[4] It follows from Equation 4.10 that $\lim_{t\to\infty} \frac{k_a(D''_{1:t}) - C(t)F^\star_a}{t} = 0$ (since $\frac{k_a(D''_{1:t}) - C(t)F^\star_a}{t}$ is sandwiched between $\frac{\Delta_l(a)}{t}$ and $\frac{\Delta_h(a)}{t}$). By Theorem A.3, $\lim_{t\to\infty} \frac{k_a(D''_{1:t})}{t} = F^\star_a$. Therefore $\lim_{t\to\infty} \frac{C(t)}{t} = 1$.

[5] Let C' be the translation of C by an arbitrary constant $\delta \in \mathbb{R}$ ($\forall t$: $C'(t) = C(t) + \delta$), and let $\Delta'_l(a) = \Delta_l(a) - \delta F^\star_a$ and $\Delta'_h(a) = \Delta_h(a) - \delta F^\star_a$ (such that C', Δ'_l and Δ'_h satisfy Equation 4.10). The windfall bounds produced by Theorem 4.1 for C', Δ'_l and Δ'_h are identical to those for C, Δ_l and Δ_h, since $\delta \times \sum_{i=1}^{n^\star_a} F^\star_i X_{i,b} = 0$ (by Equation 4.2).

$\theta_i \leq 1$) and the right hand side is non-negative ($\theta_j \geq 0$). This derivation step is the whole extent Equation 4.13 is needed in my proofs.

If $k_i(D''_{1:t}) > 0$, let t' be the last time action i was used ($t' = \max_{\tau=1\ldots t} D''_\tau = i$). All functions $k_l(D''_{1:t})$ are monotonically non-decreasing with t because $\forall l \in \{1 \ldots n^\star_a\}$ and $\forall t \in \mathbb{N}$: $k_l(D''_{1:t}) = k_l(D''_{1:t-1}) + 1$ if $D''_t = l$ and $k_l(D''_{1:t}) = k_l(D''_{1:t-1})$ otherwise. Consequently, the following must hold:

$$
\begin{aligned}
\frac{k_i(D''_{1:t}) + \theta_i - 1}{F^\star_i} &= \frac{k_i(D''_{1:t'}) + \theta_i - 1}{F^\star_i} && \text{(since } i \text{ was last used at } t') \\
&= \frac{k_{D''_{t'}}(D''_{1:t'}) + \theta_{D''_{t'}} - 1}{F^\star_{D''_{t'}}} && \text{(because } D''_{t'} = i) \\
&= \frac{k_{D''_{t'}}(D''_{1:t'-1}) + \theta_{D''_{t'}}}{F^\star_{D''_{t'}}} \\
&\leq \frac{k_j(D''_{1:t'-1}) + \theta_j}{F^\star_j} && \text{(Lemma 4.9)} \\
&\leq \frac{k_j(D''_{1:t}) + \theta_j}{F^\star_j} && (k_j \text{ is monotonically non-decreasing).} \quad \square
\end{aligned}
$$

I now present several windfall lower bounds based on a number of $C(t)$ functions I investigated (see Appendix A.2 for the step by step derivations). Equation 4.14 (corresponding to $C(t) = \frac{k_{a'}(D''_{1:t})}{F^\star_{a'}}$) produces multiple windfall lower bounds, depending on the choice of $a' \in \{1 \ldots n^\star_a\}$.

$$
\text{Windfall}_b(D'', t) \geq -\sum_{a=1}^{n^\star_a} \theta_a X_{a,b} + \sum_{a=1}^{n^\star_a} X^-_{a,b} + \frac{1}{F^\star_{a'}} \sum_{a=1}^{n^\star_a} F^\star_a X^-_{a,b} + |X_{a',b}| \tag{4.14}
$$

$$
\text{Windfall}_b(D'', t) \geq -\sum_{a=1}^{n^\star_a} \theta_a X_{a,b} + \sum_{a=1}^{n^\star_a} X^-_{a,b} \tag{4.15}
$$

$$\text{Windfall}_b(D'', t) \geq -\sum_{a=1}^{n_a^\star} \theta_a X_{a,b} + \frac{F^\star_{a''}}{\min_{a''} F^\star_{a''}} \sum_{a=1}^{n_a^\star} F^\star_a X^-_{a,b} \tag{4.16}$$

$$\text{Windfall}_b(D'', t) \geq -\sum_{a=1}^{n_a^\star} \theta_a X^+_{a,b} + (\max_{a''} \frac{1-\theta_{a''}}{F^\star_{a''}}) \sum_{a=1}^{n_a^\star} F^\star_a X^-_{a,b}. \tag{4.17}$$

4.2.1 Choosing θ

In this section I discuss several ways to choose θ in connection with WL.

- The first approach is to optimize WL for each of the Equations 4.14, 4.15, 4.16, and 4.17, and keep the largest. To this end I solve $n_a^\star + 3$ linear programs, using the following formulation:

  $$\begin{aligned} &\text{Maximize } \text{WL} \text{ subject to:} \\ &\text{WL} \leq \text{WL}_b(\theta), && (\forall b \in \mathbb{B}) \\ &0 \leq \theta_a \leq 1, && (\forall a \in [1 \ldots n_a^\star]) \end{aligned}$$

  where $\text{WL}_b(\theta)$ is the right-hand side of either Equation 4.14, 4.15, 4.16, or 4.17. Note that these LPs have $O(n_a^\star)$ variables (i.e. $\theta_1, \ldots, \theta_{n_a^\star}$) and $O(n_b + n_a^\star)$ constraints.[6] Let $\text{WL}^{(\boldsymbol{\theta})}$ denote the worst-case loss bound one can guarantee for $\mathbf{GF}^{\boldsymbol{\theta}}$ for a given problem instance after solving these linear programs.

- Theoretically, one could obtain a better WL with a single mathematical program combining Equations 4.14, 4.15, 4.16, and 4.17. While this can be achieved with a mixed integer program formulation (following an idea from [24]), or a quadratic constraint program formulation [88], it is not clear that it can be achieved with a linear program.

[6] When implementing Equation 4.17, an extra variable and $n_a^\star$ extra constraints are needed to compute $\max_{a''} \frac{1-\theta_{a''}}{F^\star_{a''}}$.

- Conversely, one could trade performance (with respect to WL) for a speed-up of this preprocessing phase. One could solve only a subset of the $n_a^\star + 3$ linear programs.[7] Alternatively, one could use a stochastic algorithm to try different θ profiles and plug them into Equations 4.14, 4.15, 4.16, and 4.17 to compute WL (such that no linear programs need to be solved).

  The method I previously introduced in [10] considers only $\theta = [0, \ldots, 0]$ and $\theta = [1, \ldots, 1]$. It computes the two corresponding WL values ($\text{WL}^{(\mathbf{0})}$ and $\text{WL}^{(\mathbf{1})}$, respectively) based on Equation 4.15 alone, then uses θ with the largest worst-loss guarantee. Let $\mathbf{GF^{01}}$ refer to this method, and $\text{WL}^{(\mathbf{01})}$ be its WL bound:

$$\text{WL}^{(\mathbf{0})} = \min_{b \in \mathbb{B}} \sum_{i=1}^{n_a^\star} X_{i,b}^- \tag{4.18}$$

$$\text{WL}^{(\mathbf{1})} = \min_{b \in \mathbb{B}} \sum_{i=1}^{n_a^\star} -X_{i,b}^+ \tag{4.19}$$

$$\text{WL}^{(\mathbf{01})} = \max(\text{WL}^{(\mathbf{0})}, \text{WL}^{(\mathbf{1})}). \tag{4.20}$$

I will revisit $\mathbf{GF^{01}}$ in Section 4.4 and show that it produces good results when compared to the literature. It follows that even better WL could be guaranteed (through the other methods I described in this section) if one could afford more preprocessing time. In Section 4.3.3, I present an empirical comparison of $\mathbf{GF^{01}}$ and $\mathbf{GF^{\theta}}$ with respect to both analytical WL bounds and actual WL bounds (observed during simulations).

Furthermore, I will use Equation 4.20 as analytical WL bounds for all my algorithms, since the actual performance of $\mathbf{GF^{\theta}}$ depends on LP-decided θ, which is difficult to express

[7] Preliminary empirical tests with randomly generated problems suggest that the function $C(t) = \min_{a'} \frac{k_{a'}(D''_{1:t}) + \theta_{a'}}{F^\star_{a'}}$ (Equation 4.15) produces WL results better than or equal to the results of any of the other investigated functions. It is not yet clear whether any of the Equations 4.14, 4.16, and 4.17 can produce strictly better results than Equation 4.15 (see Appendix A.2.5). This empirical evidence suggests that if there is only time for solving a single LP, it should be the one based on Equation 4.15.

analytically. It follows that the approximation ratio I will compute based on Equation 4.20 holds for all my algorithms.

Approximation Ratio An approximation algorithm has an approximation ratio $\epsilon \geq 1$ if its performance is never worse than ϵ times the optimal result. That it, $\mathrm{WL}(P) \geq \epsilon\, \mathrm{WL}^{\mathrm{OPT}}(P)$, where $\mathrm{WL}^{\mathrm{OPT}(P)}$ is the optimal worst-loss for some problem P, and $\mathrm{WL}(P)$ be the worst-loss guaranteed by **GF**θ for problem P. The closer ϵ to 1, the better the algorithm.

Based on the fact that some action must be chosen first, it holds that $\mathrm{WL}^{\mathrm{OPT}} \leq \max_j \min_b X_{j,b}$. It follows that the approximation ratio of my algorithms is bounded by:

$$\epsilon \leq \frac{\mathrm{WL}^{(\mathbf{01})}}{\max_j \min_b X_{j,b}}. \tag{4.21}$$

See Appendix A.3 for a discussion on the tightness of this bound.

4.3 Discussion

In Section 4.2 I introduced a family of methods for choosing actions such that the windfalls are always lower-bounded; furthermore and I provided analytical bounds and an approximation ratio. In this section I illustrate the performance of these algorithms on two numerical examples, I present their computational complexity, and discuss several variations and improvements.

Example 4.4. Let me revisit Example 4.3, where $U^\star = [0, 0, 0]$, so $X_{a,b} = R_b(a)$, $\forall a \in \mathbb{A}$ and $\forall b \in \mathbb{B}$. Note that the worst loss produced by **GF^0** is $\mathrm{WL}^{(\mathbf{0})} = -76$, because $\sum_{i=1}^{n_a^\star} X^-_{i,b1} = -20 - 56 = -76$; $\sum_{i=1}^{n_a^\star} X^-_{i,b2} = -60$; and $\sum_{i=1}^{n_a^\star} X^-_{i,b3} = -20 - 40 = -60$. As for **GF^1**, $\mathrm{WL}^{(\mathbf{1})} = -64$, since $\sum_{i=1}^{n_a^\star} -X^+_{i,b1} = -60$; $\sum_{i=1}^{n_a^\star} -X^+_{i,b2} = -30 - 30 = -60$; and $\sum_{i=1}^{n_a^\star} -X^+_{i,b3} = -64$. In this case **GF^{01}** is essentially **GF^1**, and $\mathrm{WL}^{(\mathbf{01})} = \mathrm{WL}^{(\mathbf{1})} = -64$.

It turns out $\mathbf{GF}^{\boldsymbol{\theta}}$ is able to guarantee a worst loss of -61.92 to each beneficiary when $\theta = [124/375, 0, 2/15]$, so $\mathrm{WL}^{(\boldsymbol{\theta})} = -61.92$.

Example 4.5. In this numerical example $\mathbf{GF}^{\boldsymbol{\theta}}$ produces a more significant improvement over $\mathbf{GF}^{\mathbf{01}}$ than in Example 4.4.

		Beneficiaries		
		b_1	b_2	b_3
	a_1	-244	55	-14
Actions	a_2	-198	-152	101
	a_3	998	101	-129

I claim without proof that $U^\star = [0, 0, 0]$ (so $X_{a,b} = R_b(a)$, $\forall a \in \mathbb{A}$ and $\forall b \in \mathbb{B}$), and the unique $F^\star = [\frac{409}{805}, \frac{247}{805}, \frac{149}{805}]$. Furthermore, $\mathrm{WL}^{(\mathbf{01})} = -442$ ($\mathrm{WL}^{(\mathbf{0})} = -442$ and $\mathrm{WL}^{(\mathbf{1})} = -998$, both due to beneficiary b_1).

$\mathbf{GF}^{\boldsymbol{\theta}}$ is able to guarantee a worst loss of -159.13 to each beneficiary, so $\mathrm{WL}^{(\boldsymbol{\theta})} = -159.13$, which is only 36% of $\mathrm{WL}^{(\mathbf{01})}$. In other words, the best of $\mathbf{GF}^{\mathbf{0}}$ and $\mathbf{GF}^{\mathbf{1}}$ is 2.77 times worse than $\mathbf{GF}^{\boldsymbol{\theta}}$ in this case.

Complexity Aside the preprocessing phase, the computational complexity of $\mathbf{GF}^{\boldsymbol{\theta}}$ is $O(\lg n_a^\star)$ per time step. This is because one can use a heap to store actions' $\frac{k_a(D_{1:t-1})+\theta_a}{F_a^\star}$ scores (a single action's score changes at each time step).

4.3.1 Eliminating Unnecessary Actions

If the vector $F^\star$ is not unique, the particular choice of $F^\star$ influences both the time complexity (through $n_a^\star$), and the worst loss. For the simple $\mathbf{GF}^{\mathbf{01}}$ algorithm, each beneficiary's worst loss is bounded below by the sum of its negative X values (or the negative sum of its positive X values), so eliminating an action can only improve the worst-case loss (Equation 4.20).

Empirical evidence based on randomly generated problem instances suggests the same is true for $\mathbf{GF}^{\boldsymbol{\theta}}$.

Problem 3 (BEST $F^\star$ PROBLEM)**.** Given the setup in the BASE PROBLEM, the vector $U^\star$ and a value $r \in \mathbb{R}$, is there a valid $F^\star$ vector such that $\max(LB_1, LB_2) \geq r$?

I prove in Theorem A.4 (Appendix A) that Problem 3 is NP-complete.

Theorem 4.2. There must always exist $F^\star$ such that $n_a^\star \leq n_b$.

Proof. Carathéodory's theorem [60, 136] guarantees that any point $U \in \mathbb{H} \subset \mathbb{R}^{n_b}$ can be expressed as a linear combination of $n_b + 1$ vertices of $\mathbb{H}$. $U^\star$, in particular, is *on the boundary* of $\mathbb{H}$ (leximin-optimality implies Pareto-optimality), so $U^\star$ can be expressed as a linear combination of n_b vertices of $\mathbb{H}$ [136]. Equivalently, there exists a vector $F^\star$ with no more than n_b positive components, and the result follows. □

Based on this result, I can eliminate at least $\max(n_a - n_b, 0)$ actions in a preprocessing phase, thus reducing the per-step complexity to $O(\lg(\min(n_a, n_b)))$.

I sketch a simple algorithm for this task based on a particular constructive proof for Carathéodory's theorem (e.g. [60]). As long as $n_a^\star > n_b$, I can use the Gaussian elimination algorithm (e.g. [91]) to find a non-trivial solution $\alpha_2, \ldots, \alpha_{n_a^\star}$ to the system of n_b equations: $\forall b \in \mathbb{B} : (X_{i,b} - X_{1,b}) \times \alpha_i = 0$. Let $\alpha_1 = -\sum_{i=2}^{n_a^\star} \alpha_i$ and let

$$\zeta = \min_{\substack{1 \leq i \leq n_a^\star \\ \alpha_i > 0}} \frac{F_i^\star}{\alpha_i}.$$

This is well defined, since $\sum_{i=1}^{n_a^\star} \alpha_i = 0$ and not all $\alpha_2, \ldots, \alpha_{n_a^\star}$ are zero, so at least one $\alpha_i > 0$. I compute a new $F^\star$ vector by subtracting $\zeta \times \alpha_i$ from each $F_i^\star$. The new vector is a valid $F^\star$ vector (it produces $U^\star$, since $\sum_{i=1}^{n_a^\star} \alpha_i \times X_i = 0$) and it has at least one extra zero at position k where $\frac{F_k^\star}{\alpha_k} = \zeta$. This process is repeated until no non-trivial solution $\alpha_2, \ldots, \alpha_{n_a^\star}$ can be found.

The complexity of the Gaussian elimination is $O(n_a n_b^2)$, and there are at most $n_a - 1$ iterations, so the entire pre-process of eliminating actions can be done in $O(n_a^2 n_b^2)$.

In Section 4.5 I will present an alternative to this algorithm, customized to a particular class of resource (task) allocation problems with large numbers of actions (exponential in n_b), and whose rewards to the beneficiaries can be inferred from resources' (task') rewards to beneficiaries. The main merit of that algorithm is that it generates only a small number of actions (linear in n_b).

4.3.2 Breaking Ties with GW

All lower-bounds on windfall and worst loss presented in this chapter so far hold even when ties[8] are broken in the worst possible way. It may be possible to improve on those bounds by breaking ties in a productive way, but actually finding the best way to break ties is NP-hard (see the proof of Theorem A.1). I propose a greedy approach: simply use **GW** to break ties for **GF** (for a time complexity of $O(n_b \times n_a^\star)$ per time step).

A potentially frequent occurrence of ties is not necessarily a weakness of the **GF** methods. One can use the opportunity to pursue other goals while being guaranteed that the losses are held in check. For instance, one can try to address the $S^{(a)}$ versus $S^{(d)}$ paradox from Example 4.2. In this case, one could break ties to optimize beneficiaries' *cumulative windfalls* (weighted by the probability the game ends that time step).

4.3.3 Empirical Results

Equation 4.21 provides an upper bound on how far from the optimal WL my algorithms can ever get. Appendix A.3 contains a *tight example*, i.e. a family of arbitrary-sized problem instances where the analytical WL bounds are actually optimal. Given these worst case and best case scenarios, in this section I investigate the performance of the proposed algorithms on randomly generated problems (i.e. the empirical case scenario).

[8] A tie refers to a situation where two or more actions have a minimal $\frac{k_a(D_{1:t-1})+\theta_a}{F_a^\star}$ score.

Computing the ideal WL (which I have proved it is NP-Hard) may be prohibitively expensive even for modest sized problems, so comparing the performance of an algorithm against the optimum is not feasible. Instead, I compare the performance of my algorithms in terms of both analytical WL bounds and actual WL values observed after simulations. This experiment investigates the following two questions:

Q1 How close to the analytical WL bound (i.e. the predicted WL value) should one expect the WL value actually observed during simulation?

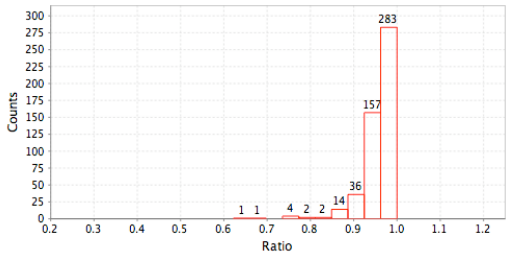
Q2 How much better are the results of $\mathbf{GF}^{\boldsymbol{\theta}}$ compared to the results of $\mathbf{GF}^{\mathbf{01}}$? This should be relevant to the practitioner trying to decide if the expected improvement in WL is worth solving the linear programs described in Section 4.2.1.

The experiment consists of 500 randomly generated test problems, each with ten beneficiaries, five actions, and rewards uniformly-chosen with replacement from the set $\{0, 1, \ldots, 20\}$. For each test problem, I compute $U^\star$ and an $F^\star$ vector (see Section 3.5), and then the analytical worst case bounds for $\mathbf{GF}^{\mathbf{01}}$ (i.e. $\mathrm{WL}^{(\mathbf{01})}$) and $\mathbf{GF}^{\boldsymbol{\theta}}$ (i.e. $\mathrm{WL}^{(\boldsymbol{\theta})}$). Next, I simulate $\mathbf{GF}^{\mathbf{01}}$ and $\mathbf{GF}^{\boldsymbol{\theta}}$, and record the observed WL during simulations. In both cases I use **GW** to break ties (see Section 4.3.2).

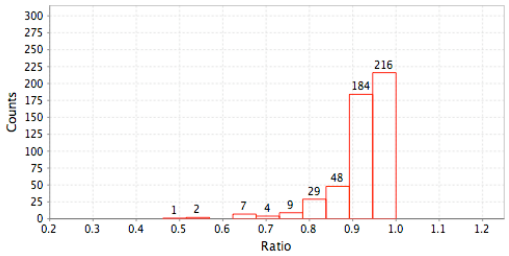
Statistic	Figure	Confidence Interval
observed WL for $\mathbf{GF}^{\boldsymbol{\theta}}$ / analytical WL for $\mathbf{GF}^{\boldsymbol{\theta}}$	4.2(a)	$[0.96661, 0.96662]$
observed WL for $\mathbf{GF}^{\mathbf{01}}$ / analytical WL for $\mathbf{GF}^{\mathbf{01}}$	4.2(b)	$[0.94026, 0.94032]$
analytical WL for $\mathbf{GF}^{\boldsymbol{\theta}}$ / analytical WL for $\mathbf{GF}^{\mathbf{01}}$	4.2(c)	$[0.76820, 0.76847]$
observed WL for $\mathbf{GF}^{\boldsymbol{\theta}}$ / observed WL for $\mathbf{GF}^{\mathbf{01}}$	4.2(d)	$[0.79999, 0.80000]$

Table 4.1: Confidence intervals for the median [184] for the statistics in Figure 4.2. There is a 99% confidence level for the results in the entire table.

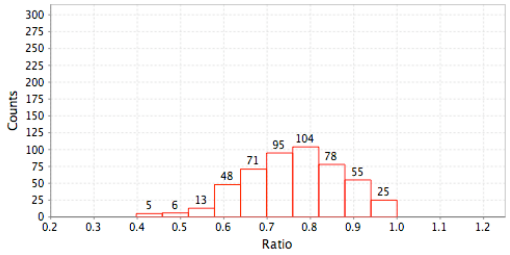
The results are presented in Figure 4.2, and Table 4.3.3. There is a 99% confidence level over all four confidence intervals (i.e. a 99.75% confidence level for each, using the Bonferroni correction). I am now in position to provide some answers to the two questions posed in this section:



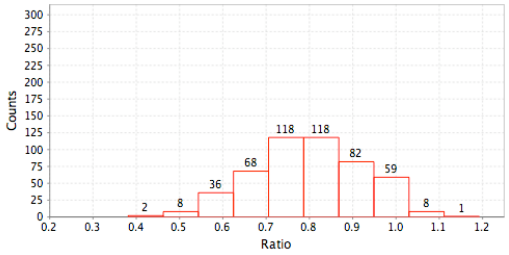
(a) Histogram of ratios of observed and analytical WL for $\mathbf{GF}^{\boldsymbol{\theta}}$.



(b) Histogram of ratios of observed and analytical WL for $\mathbf{GF}^{\mathbf{01}}$.



(c) Histogram of ratios of analytical WL bounds for $\mathbf{GF}^{\boldsymbol{\theta}}$ and $\mathbf{GF}^{\mathbf{01}}$ (i.e. $\mathrm{WL}^{(\boldsymbol{\theta})}$ / $\mathrm{WL}^{(\mathbf{01})}$).



(d) Histogram of ratios of observed WL for $\mathbf{GF}^{\boldsymbol{\theta}}$ and $\mathbf{GF}^{\mathbf{01}}$.

Figure 4.2: Empirical WL results for $\mathbf{GF}^{\boldsymbol{\theta}}$ and $\mathbf{GF}^{\mathbf{01}}$ on 500 randomly generated problems.

Q1 Figures 4.2(a) and Figure 4.2(b) show that the analytical bounds for both $\mathbf{GF}^{\boldsymbol{\theta}}$ and $\mathbf{GF}^{\mathbf{01}}$ are good approximations of what the method can achieve. In other words, one can expect these algorithms to deliver a WL close to the WL they guaranteed beforehand. More concretely, the confidence intervals for the medians in Table 4.3.3 imply that there is a 50% chance that the observed WL is within 4% of the analytical WL bound for $\mathbf{GF}^{\boldsymbol{\theta}}$, and within 6% for $\mathbf{GF}^{\mathbf{01}}$.

Q2 Figures 4.2(c) and Figure 4.2(d) (and the last two rows of Table 4.3.3) show that $\mathbf{GF}^{\boldsymbol{\theta}}$ is able to improve on both the analytical and the observed WL bounds of $\mathbf{GF}^{\mathbf{01}}$:

Analytical WL Bounds Since $\mathbf{GF}^{\mathbf{0}}$ and $\mathbf{GF}^{\mathbf{1}}$ are particular cases of $\mathbf{GF}^{\boldsymbol{\theta}}$, it was already clear that the analytical bounds of $\mathbf{GF}^{\boldsymbol{\theta}}$ must be better than those of $\mathbf{GF}^{\mathbf{01}}$. The confidence interval for the median (Table 4.3.3) shows that there is at least a 50% chance the analytical bound of $\mathbf{GF}^{\boldsymbol{\theta}}$ constitutes an improvement of at least 25% over

the analytical bound of $\mathbf{GF^{01}}$.

Observed WL Bounds The confidence interval for the median in the last row of Table 4.3.3 shows that there is at least a 50% chance the observed WL bound of $\mathbf{GF}^{\boldsymbol{\theta}}$ constitutes an improvement of 20% over the observed WL bound of $\mathbf{GF^{01}}$.

Figure 4.2(d) shows that there were a small number of tests (15 out of 500, to be exact) where the observed WL bound of $\mathbf{GF}^{\boldsymbol{\theta}}$ was actually inferior to that of $\mathbf{GF^{01}}$. I offer the following intuition, based on the discussion in Section 4.3.2 about breaking ties (i.e. when $\mathbf{GF}^{\boldsymbol{\theta}}$ is indifferent between two or more actions). First, note that the analytical WL bound for a particular problem must hold even if one breaks all ties in the **GF** algorithms in an adversarial way. Second, note that the observed WL bound is based on breaking ties in a productive way using **GW**. Finally, the choice of θ indirectly affects the ties. Intuitively, $\mathbf{GF}^{\boldsymbol{\theta}}$ is able to provide analytical WL bounds that are superior to those of $\mathbf{GF^{01}}$ by taking some of the ties off the table (so they cannot be used in an adversarial way). But those ties are used in a productive way by **GW** during simulations, so $\mathbf{GF^{01}}$ could theoretically do better than $\mathbf{GF}^{\boldsymbol{\theta}}$ simply by allowing more ties. However, this phenomena occurred in only 3% of the 500 problems in this experiment, and the performance of $\mathbf{GF}^{\boldsymbol{\theta}}$ was overall superior to that of $\mathbf{GF^{01}}$.

To summarize, (1) the analytical bounds of both $\mathbf{GF}^{\boldsymbol{\theta}}$ and $\mathbf{GF^{01}}$ are expected to be quite tight, and (2) $\mathbf{GF}^{\boldsymbol{\theta}}$ is expected to improve significantly on $\mathbf{GF^{01}}$. Note that these observations are based on numerical results for problem instances with five actions and ten beneficiaries; I leave as future work deriving dependencies between results of this kind and problem size.

4.3.4 Alternatives to Leximin

I discussed my algorithms in the context of leximin, but they can be used with any other social welfare measure. This is important since leximin's absolute priority to the worst-off might be too drastic for some applications [167]. Among the milder alternatives, I note *ordered weighted averaging* (OWA) [35, 185] and *prioritarianism* [135]; in both cases one can tune the tradeoff between leximin (fairness) and utilitarianism (efficiency). The algorithms presented in this dissertation require the vectors $U^\star$ and/or $F^\star$ as input. For *ordered weighted averaging* measures with positive, monotonically decreasing weights (which is the case for fairness), one can obtain $U^\star$ and $F^\star$ by solving a single LP (of quadratic size) [120]. When such an algorithm is missing for one's social welfare measure of choice, one could use hill climbing, simulated annealing, evolutionary algorithms, etc. If an algorithm provides the $U^\star$ vector but not $F^\star$, one can remedy the situation by solving a single LP.

4.4 Comparison with Literature

In this section it is the simplified setup $\mathbf{GF^{01}}$ (i.e. best of $\mathbf{GF^0}$ and $\mathbf{GF^1}$) that I compare against algorithms from the literature, and not $\mathbf{GF^\theta}$. This is because the WL analytical bound in Equation 4.20 is strong enough to hold against all other algorithms.

4.4.1 Compact Vector Summation Problem

The Worst Loss Optimization Problem is actually a variant of the Compact Vector Summation Problem [56, 159, 160]: given a finite set of vectors $X_1, \ldots, X_n \in \mathbb{R}^m$ such that $\sum_{i=1}^{|V|} X_i = 0$, one must find a permutation π of $\{1, 2, \ldots, n\}$ that minimizes $\max_{1 \leq k \leq n} \| \sum_{i=1}^{k} X_{\pi_i} \|$. In the earliest such work $\| \cdot \|$ was the Euclidian norm, so the problem consisted of ordering the vectors such that the path resulting from adding vectors one by one stays inside a minimum-radius m-dimensional circle centered at the origin. Later research has focused on results general enough to accommodate arbitrary norms (intuitively

a norm is a function associating a "size" value to every vector).

I believe that Sevast'janov's algorithm [159] is the most relevant algorithm to compare with my own for two reasons. First, it has the best guarantee and time complexity I am aware of in the COMPACT VECTOR SUMMATION PROBLEM literature. Second, it accommodates "asymmetric norms." This is particularly relevant because the function I try to optimize can be rearranged as an asymmetric norm but not a norm.[9] I will show that, compared to Sevast'janov's algorithm, my approach is faster and its guarantees are never worse (although they are sometimes significantly better).

Guarantee Comparison Sevast'janov's algorithm guarantees that no loss will be worse than $-M(n_b - 1 + \frac{1}{n_b})$, where $M = max_{1 \leq k \leq n} \|X_i\|$ (i.e. M is the largest absolute value of any negative component of any X vector). I claim that $\text{WL}^{(\mathbf{0})} \geq -M(n_a^\star - 1)$. This follows from Equation 4.18, replacing every negative $X_{i,b}$ with $-M$, and noticing that for any given beneficiary b at most $n_a^\star - 1$ of b's X values can be negative.[10] Therefore my worst loss, $\text{WL} \geq \text{WL}^0 \geq -M(n_a^\star - 1) = -M(\min(n_a, n_b) - 1) > -M(n_b - 1 + \frac{1}{n_b})$, the guarantee of Sevast'janov's algorithm.

Let S be a finite sequence of p actions such that $\text{Windfall}_b(S, p) = 0$, $\forall b \in \mathbb{B}$, and let S' be the reverse of S. $\text{Windfall}_b(S', t) = \sum_{i=1}^{t} X_{S'_i,b} = \sum_{i=p-t+1}^{p} X_{S_i,b} = \sum_{i=1}^{p} X_{S_i,b} - \sum_{i=1}^{p-t} X_{S_i,b} = -\sum_{i=1}^{p-t} X_{S_i,b} = -\text{Windfall}_b(S, p-t)$, $\forall t \in \{1 \dots p\}$. In other words, if one plays the sequence S in reverse, then each beneficiary experiences the same windfalls as when S is played, except with opposite signs (and in reverse order).[11] Therefore one can apply Sevast'janov's algorithm to the set of vectors $X' = -X$, then use its output in

[9] Maximizing the worst loss is equivalent to minimizing the largest absolute value of any negative coordinate of any partial sum of X vectors. This function is an asymmetric norm, but not a norm, since it satisfies the triangle inequality ($\|y + z\| \leq \|y\| + \|z\|$) and positive definiteness ($\|y\| = 0 \Rightarrow y = 0$), but it only satisfies the scalability condition ($\|ky\| = |k| \times \|y\|$) for *positive scaling* factors [21, 154].

[10] From Equation 4.2 it follows that for any beneficiary b, if there exists $i \in \{1 \dots n_a^\star\}$ such that $X_{i,b} < 0$ then there must exist $j \in \{1 \dots n_a^\star\}$ such that $X_{j,b} > 0$.

[11] Interestingly, whenever my methods produce periodic sequences (i.e. for rational $F^\star$ values) the reverse of any period produced by $\mathbf{GF^0}$ can be produced by $\mathbf{GF^1}$ and vice-versa.

reversed order. In this case no windfall can be worse than $-M'(n_b - 1 + \frac{1}{n_b})$, where M' is the largest positive component of any X vector). Using an argument similar to that in the previous paragraph, I prove that $\text{WL} \geq \text{WL}^{(\mathbf{1})} \geq -M'(n_a^\star - 1) = -M'(\min(n_a, n_b) - 1) > -M'(n_b - 1 + \frac{1}{n_b})$.

Complexity Comparison Sevast'janov's algorithm has a complexity of $O(n^2m^2)$, because it picks a vector $n - m$ times, and each such operation has a complexity of $O(km^2)$, where k is the number of alternatives ($k = n, \ldots, n - m$). An iteration in Sevast'janov's algorithm has the same complexity as an iteration in my preprocessing phase (they are both based on Gaussian elimination). However, the number of iterations in Sevast'janov's algorithm is at least the number of iterations in my preprocessing phase (each action has at least multiplicity 1). Therefore the time complexity of my preprocessing phase from Section 4.3.1 cannot be larger than that of Sevast'janov's algorithm. More importantly, even if that algorithm were extended to benefit from my preprocessing phase and to explicitly deal with multiplicities (i.e. $k = n_a^\star$), its complexity would still be $O(n_a^\star n_b^2)$ per time step, which is higher than the complexity of my algorithms (even when breaking ties with **GW**).

Sevast'janov's algorithm actually produces the output sequence in reversed order, so one needs to store an entire period on a stack. This was not an issue in the COMPACT VECTOR SUMMATION PROBLEM, where all vectors have multiplicity 1. But the memory requirements could be very large in my context, because a period can be arbitrarily long.

In summary, by eliminating unnecessary actions and only keeping track of multiplicities, I am able to offer worst case guarantees that are never worse than (and sometimes much better than[12]) Sevast'janov's.

[12]My results are superior whenever there are X values different from M and M'. In Example 4.4 my methods guarantee a worst loss of -76 and -64 respectively, while Sevast'janov's algorithm guarantees a worst loss of approximately -112 (-128 using the $X' = -X$ formulation), since $M = -60$ and $M' = -64$. Moreover, the -112 bound is achieved only if one is willing to deal with reversing the action sequence; in this case the period length is 15 ($F^\star = [\frac{4}{15}, \frac{6}{15}, \frac{5}{15}]$), but in general it can be arbitrarily long.

4.4.2 CHAIRMAN ASSIGNMENT PROBLEM

The algorithms I proposed are based on the heuristic that actions' usage should always be close to their ideal usage (i.e. $k_a(D''_{1:t})$ should be close to $tF^\star_a$ at all times, $\forall a \in \mathbb{A}$). This heuristic is closely related to the CHAIRMAN ASSIGNMENT PROBLEM from the scheduling literature [93,155,169]: $n^\star_a$ states ($n^\star_a \geq 2$) form a union and, ideally, must choose a chairman every year such that the number of chairmen assigned by each state j is always proportional to that state's weight in the union $F^\star_j$. This is impossible (ideal proportions cannot be integers at every time step), so the goal of the problem is to minimize the maximum absolute-value "discrepancy" (over all states and time steps) between the number of chairmen a state was entitled to assign and the number of chairmen it actually did.

The states and their weights correspond to social actions and their $F^\star$ values. The goal of the CHAIRMAN ASSIGNMENT PROBLEM, however, is only a stepping stone in my problem.[13] The WORST LOSS OPTIMIZATION PROBLEM has another level: beneficiaries windfall values are bounded by actions' discrepancies multiplied by the actions' X values, and it is the worst of the windfall values that must be maximized. All algorithms proposed for the CHAIRMAN ASSIGNMENT PROBLEM can be used to solve the WORST LOSS OPTIMIZATION PROBLEM, but they are oblivious to the X values (just like $\mathbf{GF^0}$ and $\mathbf{GF^1}$), and this puts them at a disadvantage.

The discrepancy δ of a sequence S is an upper bound on $|k_a(S_{1:t}) - tF^\star_a|, \forall t, \forall a$. It follows from Therorem 4.1 (with $C(t) = t$, $\Delta_h(a) = \delta$ and $\Delta_l(a) = -\delta$) that:

$$\text{Windfall}_b(S,t) \geq -\delta \sum_{a=1}^{n^\star_a} |X_{a,b}|. \tag{4.22}$$

This allows me to map any discrepancy bound δ for the CHAIRMAN ASSIGNMENT PROBLEM

[13] Baruah et al. [16] investigate non-stationary versions of this problem, which may be relevant to the $\mathbf{GF^\theta}$ extensions I describe in the future-work section of this dissertation (Section 7.1).

into a WL bound for the WORST LOSS OPTIMIZATION PROBLEM. See [169] for best discrepancy bound (to the best of my knowledge) for the CHAIRMAN ASSIGNMENT PROBLEM:[14]

$$\delta(n_a^\star) = 1 - \frac{1}{2(n_a^\star - 1)} \tag{4.23}$$

Note that $\delta(n_a^\star)$ is monotonically increasing between $\frac{1}{2}$ and 1 ($\delta(2) = \frac{1}{2}$ and $\lim_{n_a^\star \to \infty} \delta(t) = 1$). Therefore the smaller the problem, the better the loss bound in Equation 4.22. For $n_a^\star = 2$, Equation 4.22 guarantees each beneficiary a loss bound equal to the average of the loss bound guaranteed by $\mathbf{GF^0}$ (Equations 4.18) and $\mathbf{GF^1}$ (Equation 4.19). It follows that one can always do better simply by picking between $\mathbf{GF^0}$ and $\mathbf{GF^1}$ the one with the best bound (Equation 4.20).

The bound in Equation 4.23 is achieved by Tijdeman's algorithm [169]. At each time step t, it determines a set T_t of candidate actions that are underused by more than some threshold:

$$T_t = \{a \in \mathbb{A} | t F_a^\star - k_a(S_{1:t-1}) \geq \frac{1}{2n_a^\star - 2}\}, \tag{4.24}$$

then it chooses from T_t an action with the smallest "score:"

$$S_t = \text{argmin}_{a \in T_t} \left(\frac{k_a(S_{1:t-1}) + 1 - \frac{1}{2n_a^\star - 2}}{F_a^\star} - t \right)$$

or, equivalently:

$$S_t \in \left\{ a \in T_t \middle| \forall a' \in T_t : \frac{k_a(S_{1:t-1}) + \frac{2n_a^\star - 3}{2n_a^\star - 2}}{F_a^\star} \leq \frac{k_{a'}(S_{1:t-1}) + \frac{2n_a^\star - 3}{2n_a^\star - 2}}{F_{a'}^\star} \right\}. \tag{4.25}$$

[14] Better bounds were obtained in [155] for a specific subclass of problem instances.

Tijdeman's paper does not provide implementation details (or complexity bounds), so I provide my own. Note that once an action is included in T_t, it will be included in subsequent sets $T_{t+1}, T_{t+2}, \ldots$, until it is used. Also, when an action a is used at time t (i.e. $S_t = a$), one can compute the next time t' it becomes a candidate again (i.e. $a \in T_{t'}$): $t' = \lceil \frac{k_a(S_{1:t}) + \frac{1}{2n_a^\star - 2}}{F_a^\star} \rceil$.

My implementation of Tijdeman's algorithm uses two heaps, one for elements in T_t, one for elements not in T_t. The first heap uses score values (Equation 4.25) as keys, while the second heap uses time step values as keys. A key τ in the second heap is the time step when that action should be added to the first heap. At each time step, it takes $\log |T_t|$ to extract the best score action from the first heap, and $\log(n_a^\star - |T_t|)$ to insert that action into the heap of elements not in T_{t+1}. Therefore Tijdeman's algorithm (at least in this implementation) takes $O(\log n_a^\star)$ per step, just like my algorithms.

Note that Equation 4.25 is a particular case of Equation 4.9, with $\theta_a = \frac{2n_a^\star - 3}{2n_a^\star - 2}$ $(\forall a)$. Under this formulation, Tijdeman's algorithm is rather similar to my algorithms. I provide some intuition for why I get better WL bounds. The CHAIRMAN ASSIGNMENT PROBLEM treats actions' deviations from their ideal usage (both positive and negative) uniformly, while my solution uses a weighted sum of those deviations. Based on those weights (i.e. the X values), $\mathbf{GF}^{\boldsymbol{\theta}}$ is able to focus on reducing specific discrepancies rather than trying to reduce all of them. $\mathbf{GF}^{\mathbf{0}}$ and $\mathbf{GF}^{\mathbf{1}}$ are not using the X values directly, but choosing the most suitable of the two for the problem at hand proves powerful enough to outperform Tijdeman's algorithm.

I also mention the Virtual-Time Round-Robin (VTRR) [115] and Virtual-Time Fair Queuing (VTFQ) [98] algorithms. Although they do not provide any bounds on discrepancy (they call it *service time error*), they are able to achieve $O(1)$ computation complexity per time step (while my algorithms require $O(\log n_a^\star)$). These algorithms accomplish this by either moving one step forwards in a fixed list of actions (sorted decreasingly by $F^\star$ values) or by moving back to the beginning of the list.

To summarize, VTRR and VTFQ are faster than my algorithms, but cannot produce better bounds.[15] Furthermore, they cannot take advantage of breaking ties with **GW** (see Section 4.3.2 and Section 4.3.3), because of their fixed ordering of actions.

4.5 An Application to Resource and Task Allocation

The framework used in this dissertation uses actions that affect all beneficiaries to different degrees. In the context of resource (task) allocation, an action is an assignment of resources (tasks) to beneficiaries, and the number of possible assignments can be exponential in the number of beneficiaries. Therefore, the algorithms proposed in this chapter take as input an exponential (in n_b) number of assignments, yet they need at most n_b assignments (see Section 4.3.1). It makes sense to try to generate only the right assignments (i.e. only a linear rather than an exponential number of assignments).

In this section I focus on a special subclass of resource allocation problems, where:

- There are no correlations between resources (tasks). Formally, the reward a beneficiary b gets from being assigned a subset of resources (tasks) is equal to the sum of rewards b would get from getting each of the resources (tasks) separately. This assumption was used in the literature under the terminology *additive utility functions* or *linear utility functions* [7, 17, 32, 68].

- All resources must be allocated. This is known in the resource allocation literature as *free disposal*: all resources have positive rewards, since in the worst case a beneficiary can choose not to use it (i.e. there are no costs associated with using a resource). For tasks this is a most reasonable assumption: all tasks need to be accomplished, so all tasks must be allocated.

- All assignments are feasible (i.e. no constraints).

[15] The empirical results of a preliminary study (similar to that in Section 4.3.3) indicates that VTRR performs worse than $\mathbf{GF}^{\boldsymbol{\theta}}$ with respect to observed WL. I leave a more thorough comparison of $\mathbf{GF}^{\boldsymbol{\theta}}$ and VTRR (and also VTFQ) as future work.

This section introduces an alternative to the preprocessing phase in Section 4.3.1 for this class of resource (task) allocation problems.

Notations The results in this section hold both for resources and tasks; I will use the term *item* to refer to either resources or tasks. Let $\mathbb{I} = \{i_1, \ldots, i_{n_i}\}$ be a set of n_i items, and let $R_b(i)$ be the reward beneficiary b gets from item i. As stated before, $\forall I' \subseteq I$: $R_b(I') = \sum_{i \in I'} R_b(i)$. Note the problems in this class have $O(n_i \times n_b)$-sized inputs.

Hardness Result. Theorem A.1 proves that Problem 2 if NP-hard. It is trivial to show that this is still the case (i.e. optimizing WL is NP-hard) even for the subclass studied in this section: the proof is identical to the reduction from PARTITION PROBLEM used in [17] (for a slightly different problem).

Computing $U^\star$ In Chapter 3 I computed $U^\star$ by searching the space of F profiles, where F_a is the proportion of times action a is used. In this section I use a similar concept: a matrix $M \in \mathbb{R}^{n_i \times n_b}$, where $M_{i,b}$ is the proportion of times item i is assigned to beneficiary b. Note that matrix M must be row-stochastic ($\forall i \in \mathbb{I}$: $\sum_b M_{i,b} = 1$) since each item must be allocated at all times. Therefore one should use the same algorithm to search $\mathbb{M} = \{M \in \mathbb{R}^{n_i \times n_b} | \forall i \in \mathbb{I} : \sum_b M_{i,b} = 1\}$ rather than the set of F profiles. Let $M^\star$ be a matrix associated with $U^\star$. Intuitively, $M^\star$ is the equivalent of $F^\star$; formally $\sum_{i \in \mathbb{I}} M^\star_{i,b} = U^\star_b$.

Generating $O(n_b)$ Actions The following algorithm uses $M^\star$ to generate a number of actions linear in n_b:

Algorithm 1. Preprocessing Algorithm for Item Allocation.

1: $\mathbb{A} \leftarrow \emptyset$;

2: $F^\star \leftarrow \emptyset$;

3: $s \leftarrow 1$; ▷ Invariant ($\forall i \in \mathbb{I}$): $\sum_b M^\star_{i,b} = s$

4: **while** $s > 0$ **do**

5: $a = [0, \ldots, 0]$ ▷ I build action a from scratch

6: $f \leftarrow \infty$; ▷ f will hold $F^\star_a$

7: **for** $i \in \mathbb{I}$ **do**

8: $b \leftarrow$ arbitrary beneficiary such that $M^\star_{i,b} > 0$; ▷ There exists at least one such beneficiary, since $\sum_b M^\star_{i,b} = s > 0$

9: allocation$[i] \leftarrow b$

10: $a_b \leftarrow a_b + R_b(i)$; ▷ Give item i to beneficiary b

11: $f \leftarrow \min(f, M^\star_{i,b})$; ▷ Invariant: $f > 0$

12: $\mathbb{A} \leftarrow \mathbb{A} \cup \{a\}$; ▷ Add action a to action set $\mathbb{A}$

13: $F^\star \leftarrow F^\star \cup f$; ▷ Set its $F^\star$ value: $F^\star_a = f$

14: **for** $i \in \mathbb{I}$ **do** ▷ update the $M^\star$ matrix

15: $b \leftarrow$ allocation$[i]$;

16: $M^\star_{i,b} \leftarrow M^\star_{i,b} - f$;

17: $s \leftarrow s - f$;

At each iteration the algorithm creates a new action and "extracts" it from $M^\star_{i,b}$.

Example 4.6. Consider the following small example with 2 items and 2 beneficiaries:

Rewards		Beneficiaries	
		b_1	b_2
Items	i_1	1	2
	i_2	3	1

$U^\star = [2.25, 2.25]$, resulting from the following $M^\star$ matrix:

$M^\star$		Beneficiaries	
		b_1	b_2
Items	i_1	0	1
	i_2	3/4	1/4

After the first iteration, allocation$= [b_2, b_1]$, $a = [3, 2]$, $f = \frac{3}{4}$, $s = \frac{1}{4}$, and:

$M^\star$		Beneficiaries	
		b_1	b_2
Items	i_1	0	1/4
	i_2	0	1/4

After the second iteration, allocation$= [b_2, b_2]$, $a = [0, 3]$, $f = \frac{1}{4}$ and $s = 0$, and the algorithm terminates.

Intuitively, $M^\star_{i,b}$ is the fraction of time b should be assigned item i in addition to whatever b receives from the actions already in $\mathbb{A}$. An action is created by assigning each item to a beneficiary that is entitled to it under $M^\star$ (i.e. $M^\star_{i,b} > 0$). The value f is a valid $F^\star_a$ value since all beneficiaries are *still* entitled to the assigned items at least a fraction f of the time.

Example 4.7. This example has 3 items and 4 beneficiaries:

Rewards		Beneficiaries			
		b_1	b_2	b_3	b_4
	i_1	1	2	1	2
Items	i_2	3	1	2	1
	i_3	2	1	4	0

$U^\star = [\frac{48}{31}, \frac{48}{31}, \frac{48}{31}, \frac{48}{31}]$, achievable with the following $M^\star$ (although not the only one):

$M^\star$		Beneficiaries			
		b_1	b_2	b_3	b_4
	i_1	0	7/31	0	24/31
Items	i_2	16/31	15/31	0	0
	i_2	0	19/31	12/31	0

Allocations			Rewards				$F^\star$
i_1	i_2	i_3	b_1	b_2	b_3	b_4	
b_2	b_1	b_2	3	3	0	0	7/31
b_4	b_1	b_2	3	1	0	2	9/31
b_4	b_2	b_2	0	2	0	2	3/31
b_4	b_2	b_3	0	1	4	2	12/31

Proof of Correctness Note that all $M^\star$ rows sum to 1 initially and at each iteration the same value f is subtracted from each row. Therefore all rows have the same sum s. As long as there are positive entries (on all rows), a positive value is chosen from each row, and f is set to the smallest of these values. It follows that $f > 0$ and line 16 guarantees $M^\star_{i,b} \geq 0$. Therefore, the following invariants hold at the end of each while-loop iteration:

$$M^\star_{i,b} \geq 0,$$

$$f > 0,$$

$$\sum_b M^\star_{i,b} = s,$$

$$s + \sum_a F^\star_a = 1.$$

At each iteration the algorithm creates a new item allocation, where all items are allocated. The algorithm continues until $M^\star$ contains only zeros, and at least one entry in $M^\star$ becomes zero at each iteration (line 16). It follows that the algorithm finishes in a finite number of steps, and at each iteration a valid item allocation was produced.

Time Complexity The number of while-loop iterations is upper-bounded by the number of non-zero entries in the original $M^\star$ matrix. If follows that there are at most $n_i n_b$ iterations,[16] so $|\mathbb{A}| \leq n_i n_b$. The time complexity for this algorithm is $O(n_i^2 n_b)$ (by using a linked lists to hold the positive elements on each row of $M^\star$).

At this point one can use the algorithm in Section 4.3.1 to trim $\mathbb{A}$ to at most n_b actions in polynomial time ($O((n_i n_b)^2 n_b^2) = O(n_i^2 n_b^4)$).

[16] There are actually at most $n_i n_b - n_i + 1$ iterations, as the last iteration makes n_i zeros.

Chapter 5: Stateful Domains

My work in Chapter 4 assumed that any action can be chosen at any time step, and it will always yield the same rewards to the beneficiaries. In this chapter I generalize my previous results to stateful domains, where actions have as side-effect changing the state of the world. Past decisions affect the set of actions available in the future and/or the rewards those actions yield to the beneficiaries.

The layout of the chapter is as follows: first I provide motivation for considering stateful domains (Section 5.1), then I discuss the framework in Section 5.2, emphasizing the implications and conditions for extending the formulation of Problem 2 (the stateless WORST LOSS OPTIMIZATION PROBLEM) to a stateful problem. I argue for a multi-objective optimization formulation, since the two goals of improving the long-term utility profile and improving the worst-case loss might be conflicting. Next, I generalize the worst-case loss bounds of $\mathbf{GF}^{\theta}$ to the stateful problem (Section 5.3), and incorporate that into an evolutionary computation based algorithm for building Pareto-fronts (Section 5.4). I validate this approach in Section 5.5.

5.1 Motivation

I revisit the static problem in Example 3.1, where each professor always receives the same reward from a particular assignment of classes. It is conceivable the reward a professor gets when teaching some class is higher if he also taught that class the previous semester than if he taught a different class (in the first case it is easier to prepare for the class). Alternatively, teaching the advanced class right after the introductory class might mean a better reward, since the students took the introductory class with the same professor and they are used to

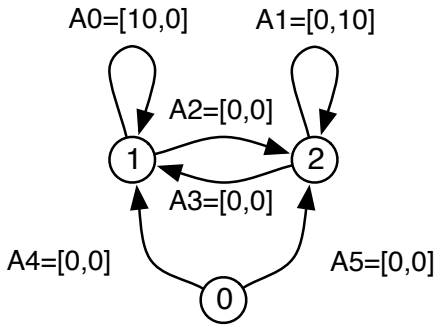


Figure 5.1: State graph for Example 5.1.

his style. In this example all actions are always available, but their rewards depend on the choice of action at the previous step.

Example 5.1. In Example 3.1 both professors got a reward of zero from teaching the hard class and a reward of ten from teaching the easy class. In this example professors get a reward of ten from teaching the easy class *only* if they also taught the easy class in the previous semester. Professors still get a reward of zero from teaching the hard class, and co-teaching the classes (action a_3 in Example 3.1) is no longer an option. I modeled this example using the state-graph Figure 5.1. In state 1 the first professor (beneficiary b_1) taught the easy class last time, and in state 2 it is the second professor (beneficiary b_2) that taught the easy class last time. State 0 is the initial state.

Consider now the problem of assigning shifts to hospital nurses [29]. A nurse's sequence of shifts is restricted through a number of rules such as "no more than X shifts in a week," or "no more than Y night shifts in a week" [30], or "at least one weekend off every two weeks" [105]. Millar and Kiragu [105] create a graph of nurse shifts and use it to generate "cycling schedules."[1] However, nurses may have different X and Y values written in their contracts, so these "cycling schedules" are not always applicable. Additionally, there are restrictions on the set of nurses scheduled on each shift: "at least Z registered nurses and W

[1] "A cyclic schedule consists of a set of work patterns which is rotated among a group of workers over a set scheduling horizon. At the end of the scheduling horizon each worker would have completed each pattern exactly once" [105].

certified nursing assistants per shift." In this problem domain the set of available actions (full assignments of shifts to nurses) changes from one time step to the next.

5.2 Framework

Based on the examples in Section 5.1, I will focus my study of long-term fairness in stateful domains to the following framework:

- There is a finite set of states;
- In addition to giving rewards to beneficiaries, actions have as side-effect the transition of the system from one state to another with probability 1 (i.e. a purely deterministic transition model).
- I assume all states are reachable from the initial state (otherwise they should be removed during a preprocessing phase).
- Since the controller does not determine when the process stops,[2] there are no "terminal" states (like in [133, 134]), so I require that at least one action is available from every state.

The state graph is actually a multi-graph, since it may contain *parallel edges* (i.e. multiple actions with the same source state and the same destination state, but with different reward profiles) and *self-loops* (i.e. actions where the source state is the same as the destination state).

The approach used to solve the stochastic time horizon model in Chapter 4 converges to $U^\star$ (the best possible utility profile), while insuring that no beneficiary falls behind too much at any time. This approach was motivated by the time horizon being stochastic or altogether unknown. In the stateless case, any profile achievable in finite time is also achievable at the limit (Lemma 2), but not necessarily the other way around. Since $U^\star$ might

[2] It is conceivable that a beneficiary (professor, nurse) would wait for a large positive windfall to quit/retire; I do not consider this aspect in my framework.

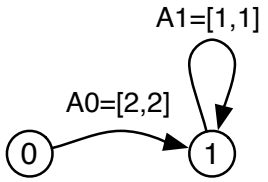


Figure 5.2: Simple example where all utility profiles achievable in finite time when starting from state 0 (i.e. $U_t = [1 + \frac{1}{t}, 1 + \frac{1}{t}], \forall t \in \mathbb{N}$) are strictly better than $[1, 1]$, the only profile achievable at the limit.

be achievable only at the limit, the longer the process lasts, the better the performance of this approach.

I extend this approach to stateful domains, i.e. I choose an utility profile to converge to, and bound the beneficiaries' losses with respect to that utility profile. There are a number of differences with respect to the utility profiles U and action frequency profiles F that have to be considered when generalizing from one state to multiple states:

1. Not every utility profile achievable in finite time is achievable at the limit. Furthermore, there exist utility profiles in the first category that are strictly (leximin) superior to any profile in the second category (e.g. Figure 5.2).[3] However, a profile that can only be achieved in τ steps is of little interest if the game can never end in exactly τ steps. Therefore, **I focus on convergent sequences of actions and use their limit-point utility profiles to compare them** (just like in the stateless case).

   Note that an infinite sequence of actions might traverse several strongly-connected components (SCCs), but eventually must settle in some SCC. This is because the sequence of actions traces a path through the graph, and once the path leaves a SCC, it cannot return to it. It follows that limit-point utility profiles are associated with SCCs, and one's path through the graph to reach one's final SCC is not important.

2. The leximin-optimal limit-point utility profile in each SCC is unique, but since there

[3] Unlike in the stateless case, *it is* possible for all beneficiaries' windfall values to be simultaneously positive.

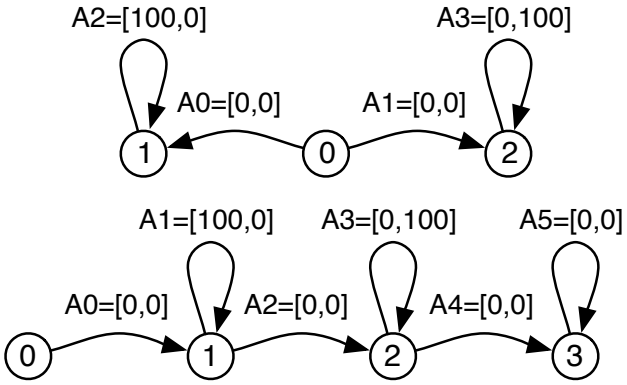


Figure 5.3: Examples with two different leximin-optimal limit-point utility profiles: $[100, 0]$ and $[0, 100]$ (in both cases the process starts in state 0).

can be multiple SCCs, the graph might contain more than one leximin-optimal limit-point utility profile (see Figure 5.3 for two examples). Choosing between the different optimal limit points might cause envy between beneficiaries.[4] This issues could be attenuated by choosing randomly between the optimal limit-point utility profiles.

It follows from these observations, that **one should find the optimal limit-point utility profile in each SCC, and choose the best one, breaking ties randomly.**

3. The system performs a walk in the graph, so the number of times a state was entered cannot differ from the number of times that state is left by more than ± 1 (-1 for the current state and $+1$ for the initial state). If follows that, **at the limit, the sum of usage frequencies for actions entering a state must be equal to the sum of usage frequencies for the actions leaving the state.**

4. There are leximin-optimal limit-point utility profiles for which no finite WL is possible.

   Example 5.2. Consider Example 5.1, where one must assign an easy class and a hard class to two professors; the twist is that the professors get a reward from teaching the easy class only if they also taught it in the previous semester. Figure 5.4 shows the SCC for the state-graph in Figure 5.1 (obtained by removing state 0 and actions

[4] Two leximin-optimal utility profiles $U \neq U'$ must be permutations of each other, so $\exists b, b' \in \mathbb{B}$ such that $U_b < U'_b = U_{b'}$. If U is chosen over U', then beneficiary b envies b'.

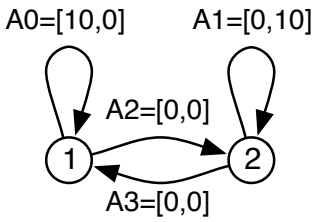


Figure 5.4: Example of leximin-optimal limit-point utility profile (i.e. $[5, 5]$) for which no finite WL is possible. An example of sequence that achieves the utility profile $[5, 5]$ at the limit is the sequence that uses action $A0$ α times in a row, then $A2$ once, then $A1$ α times, then $A3$, then $A0$ $\alpha + 1$ times, and so on.

$A4$ and $A5$). If the professors teach the same class for α time steps (semesters) before switching, then the limit-point utility profile is $U^{(\alpha)} = [5 - \frac{5}{\alpha+1}, 5 - \frac{5}{\alpha+1}]$, and the action frequency profile is $F^{(\alpha)} = [\frac{\alpha}{2\alpha+2}, \frac{\alpha}{2\alpha+2}, \frac{1}{2\alpha+2}, \frac{1}{2\alpha+2}]$. Note that the larger the value of α, the better the limit-point utility profile $U^{(\alpha)}$; intuitively, the less often the professors switch, the more they can take advantage of the reward they get from teaching the easy class in successive semesters. At the limit, $\lim_{\alpha\to\infty} U^{(\alpha)} = [5, 5]$ and $\lim_{\alpha\to\infty} F^{(\alpha)} = [\frac{1}{2}, \frac{1}{2}, 0, 0]$. Intuitively, this means the actions that make the transition between the two states should be used with frequency zero as time goes by. The less often the professors switch, the more one has to wait for its turn to teach the easy class, and the worse its WL.

This example shows that **it may be impossible to converge to the leximin-optimal limit-point utility profile while guaranteeing finite worst-case losses to all beneficiaries**. Arguably, a tradeoff between optimizing the long-term (i.e. at the limit) utility profile and WL might be preferable, so I will treat this as a **multi-objective optimization problem**.

5.2.1 Notation

In the remaining of this chapter I use the following notations:

- n_s is the number of steps;

- $\sigma(\cdot)$ is the function mapping finite length action sequences to the states one ends up in when one follows those sequences. Formally, if one follows the actions in subsequence $S_{1:\tau}$, then one ends up in state $\sigma(S_{1:\tau})$. Function σ is defined over the set of *valid* sequences, i.e. $\forall t \leq \tau$: S_t is an action at state $\sigma(S_{1:t-1})$, where $\sigma(\langle\rangle)$ is the initial state.
- $n_a(s)$ denotes the number of actions available in state s;
- $s.i$ refers to action i, available in state s;
- $s \xrightarrow{s.i} s'$ denotes an action i which is available in state s and, when executed, causes the system to transition into state s'. Alternatively, action i is an edge in the state graph, from state s to state s'.
- When there is no danger of confusion, I use $\sum_{s \xrightarrow{s.i} s'}$ as short hand for $\sum_s \sum_{s \xrightarrow{s.i} s'}$ to iterate over all actions entering state s'.
- $k_s(S_{1:\tau}) = \sum_{s.i=1}^{n_a(s)} k_{s.i}(S_{1:\tau})$ represents the number of times one leaves state s while following subsequence S_τ. Note that $\sum_s k_s(S_{1:\tau}) = \tau$.
- I call **siblings** two actions available in the same state.
- For a given action frequency profile F, $F_s = \sum_{s.i=1}^{n_a(s)} F_{s.i}$.
- The graph **induced** by an action frequency profile F is obtained from the original graph by removing all actions $s.i$ with $F_{s.i} = 0$ and all states s with $F_s = 0$. Intuitively, I remove the actions (states) that don't have to be used (visited) with positive frequency at the limit.
- $n_a^\star(s) = |\{s.i | F_{s.i} > 0\}|$ denotes the number of "usable" actions available in state s (usable according to action frequency profile F).
- $n_s^\star = |\{s' | F_{s'} > 0\}|$ denotes the number of states in the F-induced graph, and $n_a^\star = \sum_{s=1}^{n_s^\star} n_a^\star(s)$ denotes the number of edges.

5.3 $\mathbf{GF}^{\theta}$

In Chapter 4 I introduced the $\mathbf{GF}^{\theta}$ family of heuristics, which are used to repeatedly choose actions such that (1) the proportions the actions are used in the long run match a given action frequency profile F; and (2) a good WL is guaranteed. In this section I show that $\mathbf{GF}^{\theta}$ can trivially be applied to stateful domains, provided the action frequency profile F meets certain requirements. It is not trivial, however, to generalize the WL bounds to account for different state graph topologies; I derive these bounds in Section 5.3.4.

In the single-state setup in Chapter 4, $\mathbf{GF}^{\theta}$ computes a score for each action and chooses one of the actions with the lowest score. This paradigm extends easily to multiple states: one computes the scores of all actions *available in the current state*, then choose one of the actions with the lowest score.

5.3.1 Requirements for Long-term Action Frequency Profiles

I present some necessary conditions that a long-term action frequency profile F must satisfy in order for $\mathbf{GF}^{\theta}$ to be able to use, at the limit, all actions in the proportions prescribed by F (i.e. $\forall s.i$: $\lim_{\tau\to\infty} \frac{1}{\tau} k_{s.i}(D''_{1:\tau}) = F_{s.i}$).

Since $\sum_s k_s(S_{1:t}) = t$ and $k_{s.i}(S_{1:t}) \geq 0$ ($\forall s.i$), any valid F profile must satisfy:

Condition 1. $\sum_s F_s = 1$ and $\forall s.i$: $F_{s.i} \geq 0$.

I argued in Section 5.2 that a valid sequence of actions S enters a state s' roughly the same number of times it leaves state s' (± 1). Formally:

$$\left| \sum_{s \xrightarrow{s.i} s'} k_{s.i}(S_{1:\tau}) - \sum_{s'.j} k_{s'.j}(S_{1:\tau}) \right| \leq 1. \tag{5.1}$$

It follows that

$$\lim_{\tau\to\infty}\frac{1}{\tau}\sum_{s\xrightarrow{s.i}s'}k_{s.i}(S_{1:\tau})=\lim_{\tau\to\infty}\frac{1}{\tau}\sum_{s'.j}k_{s'.j}(S_{1:\tau}) \tag{5.2}$$

which leads to the second necessary condition:

Condition 2. $\forall s'$: $\sum_{s\xrightarrow{s.i}s'}F_{s.i}=\sum_{s'.j}F_{s'.j}$.

Another condition results from the fact that $\mathbf{GF}^{\boldsymbol{\theta}}$ will not use an actions whose F value is zero. In the example from Figure 5.4, $\mathbf{GF}^{\boldsymbol{\theta}}$ will not leave the initial state when presented with the action frequency profile $F=[\frac{1}{2},\frac{1}{2},0,0]$. More formally, the graph induced by $F=[\frac{1}{2},\frac{1}{2},0,0]$ consists of two strongly connected components, and $\mathbf{GF}^{\boldsymbol{\theta}}$ is stuck in one of them (and will not get to the other one).

Intuitively, $\mathbf{GF}^{\boldsymbol{\theta}}$ has to be able to reach all the nodes in the F-induced graph. However, for any two states s,s' in the graph induced by an F satisfying Condition 2, s is reachable from s' *if and only if* s' is reachable from s.[5] It follows that a valid F must satisfy to following condition:

Condition 3. The F-induced graph is strongly connected.

5.3.2 Preliminaries

Lemma 6. $\forall s.i,s.j$: $\frac{k_{s.i}(D''_{1:t})+\theta_{s.i}-1}{F_{s.i}}\leq\frac{k_{s.j}(D''_{1:t})+\theta_{s.j}}{F_{s.j}}$.

Proof. The result follows from Lemma 5 (from Chapter 4) applied to state s. Intuitively, an action's score can't fall *too much* behind the score of its siblings, or it would have been used

[5]Let F be profile satisfying Condition 2. I will now prove that if state s is reachable from state s' in the F-induced graph, then s' is reachable from s. I suffices to note that for any cut in the F-induced graph, the sum of F values for actions crossing the cut is always zero. Let V be the set of states in the F-induced graph. I start with the partition $(V',V-V')$, where $V'=\{s\}$. Since s' is reachable from s, there are edges crossing over the cut from V' to $V-V'$, so there must be edges crossing over the other way. I move into V' all nodes with edges crossing the cut into V', (intuitively, V' will contain states that can reach s). As long as $s'\in V-V'$, there are always edges crossing from V' to $V-V'$, so there must be edges crossing the cut the other way. There are only a finite number of states, so s' will become part of V' in a finite number of steps. Therefore s' can reach s.

more in the past. □

Lemma 7. $\forall s.i$: $-\frac{\theta_{s.i}}{F_{s.i}} + \frac{(\sum_{s.j}\theta_{s.j})-(n_a^\star(s)-1)}{F_s} \leq \frac{k_{s.i}(D''_{1:t})}{F_{s.i}} - \frac{k_s(D''_{1:t})}{F_s} \leq \frac{1-\theta_{s.i}}{F_{s.i}} + \frac{(\sum_{s.j}\theta_{s.j})-1}{F_s}$.

Proof. The results follows from summing the results of Lemma 6 applied to each sibling of $s.i$; the derivation uses the same steps as the proof of Theorem A.3. □

Lemma 6 connects the k values (i.e. usage counts) of sibling actions. Lemma 7 connects the k value of an action with the k value of that state where the action is available.

Next, I will connect the k value of different states. Specifically, I use Lemma 7 on each action entering state s', and tie them together using Equation 5.1 (connecting the k value of state s' with the k values of the actions entering s'):

$$-\frac{n_a^\star(s)-1}{F_s} \leq \frac{k_{s.i}(D''_{1:t})}{F_{s.i}} - \frac{k_s(D''_{1:t})}{F_s} + \frac{\theta_{s.i}}{F_{s.i}} - \frac{\sum_{s.j}\theta_{s.j}}{F_s} \leq \frac{1}{F_{s.i}} - \frac{1}{F_s}$$

$$F_{s.i}\frac{1-n_a^\star(s)}{F_s} \leq k_{s.i}(D''_{1:t}) - F_{s.i}\frac{k_s(D''_{1:t})}{F_s} + \theta_{s.i} - F_{s.i}\frac{\sum_{s.j}\theta_{s.j}}{F_s} \leq 1 - \frac{F_{s.i}}{F_s}$$

$$\sum_{s\xrightarrow{s.i}s'}\left[F_{s.i}\frac{1-n_a^\star(s)}{F_s}\right] \leq \sum_{s\xrightarrow{s.i}s'} k_{s.i}(D''_{1:t}) - \sum_{s\xrightarrow{s.i}s'}\left[F_{s.i}\frac{k_s(D''_{1:t})}{F_s} - \theta_{s.i} + \frac{F_{s.i}}{F_s}\sum_{s.j}\theta_{s.j}\right] \leq \sum_{s\xrightarrow{s.i}s'}\left[1-\frac{F_{s.i}}{F_s}\right]$$

$$\sum_{s\xrightarrow{s.i}s'}\left[F_{s.i}\frac{1-n_a^\star(s)}{F_s}\right] - 1 \leq k_{s'}(D''_{1:t}) - \sum_{s\xrightarrow{s.i}s'}\left[k_s(D''_{1:t})\frac{F_{s.i}}{F_s} - \theta_{s.i} + \frac{F_{s.i}}{F_s}\sum_{s.j}\theta_{s.j}\right] \leq \sum_{s\xrightarrow{s.i}s'}\left[1-\frac{F_{s.i}}{F_s}\right]+1$$

I write the previous double inequality (for all states in the F-induced graph) in matrix form:

$$\bar{l} - T\bar{\theta} \leq (I - P^\top)\bar{k} \leq \bar{u} - T\bar{\theta} \tag{5.3}$$

where

- $P \in \mathbb{R}^{n_s^\star \times n_s^\star}$, $P_{s,s'} = \sum_{s\xrightarrow{s.i}s'} \frac{F_{s.i}}{F_s}$;

- $\overline{l} \in \mathbb{R}^{n_s^\star}$, $\overline{l}_{s'} = -1 - \sum_{s \xrightarrow{s.i} s'} (n_a^\star(s) - 1) \frac{F_{s.i}}{F_s}$;
- $\overline{u} \in \mathbb{R}^{n_s^\star}$, $\overline{u}_{s'} = 1 + \sum_{s \xrightarrow{s.i} s'} [1 - \frac{F_{s.i}}{F_s}]$;
- $\overline{k}$ is a column vector of $n_s^\star$ variables, $\overline{k}_{s'} = k_{s'}(D''_{1:t})$;
- $\overline{\theta}$ is a column vector of size $n_a^\star$, containing $\theta_{s.i}$ variables;
- $T \in \mathbb{R}^{n_s^\star \times n_a^\star}$, $T_{s',s'' \xrightarrow{s''.i} s'''} = \mathbf{1}(s' = s''') - \sum_{s'' \xrightarrow{s''.i} s'} \frac{F_{s''.i}}{F_{s''}} = \mathbf{1}(s' = s''') - P_{s'',s'}$, where $\mathbf{1}(\text{true}) = 1$ and $\mathbf{1}(\text{false}) = 0$.

Let $Q = I - P^\top$ and let $Q_{[-s]}$ be the principal submatrix of Q obtained by deleting the s^{th} row and column. Let $\overline{l}_{[-s]}$, $\overline{u}_{[-s]}$ and $\overline{k}_{[-s]}$ be the column vectors obtained by deleting the s^{th} entry from $\overline{l}$, $\overline{u}$ and $\overline{k}$ respectively. Finally, $T_{[-s]}$ is the matrix obtained by deleting the s^{th} row from T. Let $\overline{b}_s$ be the s^{th} column of $I - P^\top$, without the s^{th} entry, and let k_s be the s^{th} entry of $\overline{k}$, that is $k_s(D''_{1:t})$. I remove the s^{th} inequality from Equation 5.3 and re-write the rest using these new notations:

$$\overline{l}_{[-s]} - T_{[-s]}\overline{\theta} \leq Q_{[-s]}\overline{k}_{[-s]} + \overline{b}_{[-s]}k_s \leq \overline{u}_{[-s]} - T_{[-s]}\overline{\theta} \tag{5.4}$$

The matrix P is row stochastic (the entries of every row are nonnegative and sum to 1). Condition 3 (F induces a strongly connected graph), implies P is irreducible. The spectral radius[6] of matrices P and $P^\top$ is 1 since a row stochastic matrix has spectral radius 1 [74, fact 1a, page 9-15], and a matrix and its transpose have the same eigenvalues [74, fact 10, page 4-7].

Matrix Q is an M_0-matrix,[7] since $P^\top$ is non-negative and has spectral radius 1 [74, page 35-12]. Q is also singular (all its columns sum to zero, so its rows are not linearly

[6]The spectral radius of a matrix A is defined as $\rho(A) = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } A\}$.

[7]A matrix A is an M_0-matrix if there exists a scalar $a \geq \rho(A)$ and a nonnegative matrix P such that $A = aI - P$. If $a > \rho(A)$, then the matrix A is said to be an M-matrix.

independent). It follows from [74, fact 10d, page 9-20] that all $Q_{[-s]}$ matrices are M-matrices, and therefore inverse-nonnegative [74, fact 1d, page 9-18].

Because $Q_{[-s]}^{-1}$ exists and it is nonnegative, I claim that the following system of inequalities results from Equation 5.4:

$$Q_{[-s]}^{-1} l_{[-s]} - Q_{[-s]}^{-1} T_{[-s]} \overline{\theta} \leq \overline{k}_{[-s]} + Q_{[-s]}^{-1} \overline{b}_{[-s]} k_s \leq Q_{[-s]}^{-1} \overline{u}_{[-s]} - Q_{[-s]}^{-1} T_{[-s]} \overline{\theta} \tag{5.5}$$

This is because each inequality (row) in Equation 5.5 is obtained as a sum of inequalities in Equation 5.3 multiplied by nonnegative values (from some row of $Q_{[-s]}^{-1}$), thus preserving the inequalities.

I make the following notations: $\overline{l}'_{[-s]} = Q_{[-s]}^{-1} \overline{l}_{[-s]}$, $\overline{u}'_{[-s]} = Q_{[-s]}^{-1} \overline{u}_{[-s]}$, $\overline{b}'_{[-s]} = Q_{[-s]}^{-1} \overline{b}_{[-s]}$, and $T'_{[-s]} = Q_{[-s]}^{-1} T_{[-s]}$.

$$\overline{l}'_{[-s]} - T'_{[-s]} \overline{\theta} \leq \overline{k}_{[-s]} + \overline{b}'_{[-s]} k_s \leq \overline{u}'_{[-s]} - T'_{[-s]} \overline{\theta} \tag{5.6}$$

This is an important result. First, it allows me to prove that in the long run $\mathbf{GF}^{\boldsymbol{\theta}}$ uses all actions in the proportions specified in F (Section 5.3.3). Second, it connects the k value of each state with the k value of state s, which allows me to establish bounds for WL (Section 5.3.4).

5.3.3 Convergence Guarantee

By Lemma 7, all action $s.i$ available at state s are used in the appropriate relative proportions in the long run (i.e. $\lim_{t\to\infty} \frac{k_{s.i}(D''_{1:t})}{k_s(D''_{1:t})} = \frac{F_{s.i}}{F_s}$). In order to show that in the long run all actions are used in the correct proportions (i.e. $\lim_{t\to\infty} \frac{k_{s.i}(D''_{1:t})}{t} = F_{s.i}$), I also have to show that all states are visited in the correct proportions in the long run (i.e. $\lim_{t\to\infty} \frac{k_s(D''_{1:t})}{t} = F_s$).

Note that if a state s is visited an infinite number of times, then all states s' that usable

actions $s.i$ point to will also be visited an infinite number of times. This means that all states reachable from s along edges with positive F values are also visited an infinite number of times. Moreover, the pigeon hole principle guarantees that an infinite sequence of actions visits at least one state (out of the finite set of states) an infinite number of times. So using $\mathbf{GF}\theta$ starting from a state of the F-induced graph guarantees that at least one of the states in the F-induced graph will be used an infinite number of times. Therefore, if the F-induced graph is strongly connected, then all the states in it are visited an infinite number of times.

Since Equation 5.6 is satisfied for any state s in the F-induced graph, it follows that $\frac{k_s(D''_{1:t})}{k_{s'}(D''_{1:t}))}$ converges as $t \to \infty$ for any two states s and s' in the F-induced graph.

Because the "sum of limits is the limit of sums[8]," and $\sum_s k_s(D''_{1:t}) = t$, it follows that $\frac{k_s(D''_{1:t}))}{t}$ also converges. Let $\alpha_s = lim_{t\to\infty}\frac{k_s(Dt)}{t}$.

It follows from the conservation of visits in each state (Equation 5.2) that the row vector $\overline{\alpha}^\top$ must satisfy the system of equations $\overline{\alpha}^\top = \overline{\alpha}^\top P$. Based on results from the Markov chain literature (see [20]), if d is the period of P (or, equivalently, the graph induced by F), then the system of linear equations $\overline{\alpha}^\top = \overline{\alpha}^\top P^d$ has a unique solution (up to a multiplicative constant). By noting that any solution to the system $\overline{\alpha}^\top = \overline{\alpha}^\top P$ also satisfies $\overline{\alpha}^\top = \overline{\alpha}^\top P^d$, it follows that the former system cannot have more than one solution (again, up to a multiplicative constant). Since $\alpha_s = F_s$ $(\forall s)$ is a solution of $\overline{\alpha}^\top = \overline{\alpha}^\top P$ and the multiplicative constant is tied by the condition $||\overline{\alpha}^\top||_1 = 1$, it follows that:

$$\forall s, t\colon \lim_{t\to\infty} \frac{k_s(Dt)}{t} = F_s. \tag{5.7}$$

[8]Given two convergent sequences a_n and b_n, such that $\lim_{n\to\infty} a_n = a$ and $\lim_{n\to\infty} b_n = b$, it follows that $\lim_{n\to\infty} a_n + b_n = a + b$.

It now follows directly from Lemma 7 that:

$$\forall s,t\colon \lim_{t\to\infty}\frac{k_{s.i}(Dt)}{k_s(D''_{1:t})}=\frac{F_{s.i}}{F_s}. \tag{5.8}$$

By multiplying Equations 5.7 and 5.8, it follows that:

$$\forall s,t\colon \lim_{t\to\infty}\frac{k_{s.i}(Dt)}{t}=F_{s.i}. \tag{5.9}$$

which proves that $\mathbf{GF}^{\theta}$ usage of actions converges to the F specifications.

5.3.4 WL Bounds for $\mathbf{GF}^{\theta}$

In Chapter 4 I use Theorem 4.1 to produce WL bounds in the single-state case; the result holds verbatim in the present setup. I restate the claim of the theorem using the notations in this chapter: if $\Delta_l(s.a)\le k_{s.a}(D''_{1:t})-C(t)F_{s.a}\le\Delta_h(s.a)$ holds $\forall t\in\mathbb{N},\forall s.a\in\{1\dots n^{\star}_a\}$, then $\text{Windfall}_b(D'',t)\ge\sum_s\sum_{s.a}\Delta_l(s.a)X^{+}_{s.a,b}+\sum_s\sum_{s.a}\Delta_h(s.a)X^{-}_{s.a,b}$.

I investigate the following $C(t)$ functions: $C(t)=\frac{k_s(D''_{1:t})}{F_s}$, $C(t)=\min_s\frac{k_s(D''_{1:t})}{F_s}$, $C(t)=\max_s\frac{k_s(D''_{1:t})}{F_s}$, $C(t)=\frac{k_{s.i}(D''_{1:t})}{F_{s.i}}$, $C(t)=\min_{s.i}\frac{k_{s.i}(D''_{1:t})}{F_{s.i}}F_{s.i}$, $C(t)=\max_{s.i}\frac{k_{s.i}(D''_{1:t})}{F_{s.i}}F_{s.i}$, $C(t)=\min_{s.i}\frac{k_{s.i}(D''_{1:t})+\theta_{s.i}}{F_{s.i}}F_{s.i}$, and $C(t)=\max_{s.i}\frac{k_{s.i}(D''_{1:t})+\theta_{s.i}}{F_{s.i}}F_{s.i}$. See the discussion in Section A.2.3 for not including $C(t)=\frac{k_{s.i}(D''_{1:t})+\theta_{s.j}}{F_{s.i}}$. I will show how the corresponding Δ_h and Δ_l functions are computed using Equation 5.6 and Lemma 7.

It follows directly from Equation 5.8 that the component of $\overline{b}_{[-s]}$ corresponding to s' must be equal to $-\frac{F_{s'}}{F_s}$. Therefore, Equation 5.6 consists of inequalities of the following form:

$$\forall s,s'\neq s,t\colon (\overline{l}'_{[-s]})_{s'}-T'_{[-s]}\overline{\theta}\le k_{s'}(D''_{1:t})-k_s(D''_{1:t})\frac{F_{s'}}{F_s}\le(\overline{u}'_{[-s]})_{s'}-T'_{[-s]}\overline{\theta} \tag{5.10}$$

Let $\bar{l}''_{[-s]}$ and $\overline{u}''_{[-s]}$ be column vectors obtained by inserting a zero on the s^{th} entry $\bar{l}'_{[-s]}$ and $\overline{u}'_{[-s]}$ respectively. Additionally, let $T''_{[-s]}$ be the matrix obtained by inserting a row of zeros as the s^{th}row of $T'_{[-s]}$. These extensions correspond to the trivial inequality $0 \leq k_{s'}(D''_{1:t}) - k_{s'}(D''_{1:t}) \leq 0$. It follows that:

$$\forall s, s', t\text{: } (\bar{l}''_{[-s]})_{s'} - (T''_{[-s]})_{s'}\overline{\theta} \leq k_{s'}(D''_{1:t}) - k_s(D''_{1:t})\frac{F_{s'}}{F_s} \leq (\overline{u}''_{[-s]})_{s'} - (T''_{[-s]})_{s'}\overline{\theta}. \tag{5.11}$$

Equation 5.11 connects the k value of any state in the F-induced graph to the k value of any other state in the F-induced graph. Combining this equation with Lemma 7 allows the k values of all action in the F-induced graph to be bounded by functions of k values of any other state or action in the F-induced graph. Therefore, it is straight forward to derive Δ_l and Δ_h functions for simple $C(t)$ involving $k_s(D''_{1:t})$ and $k_{s.i}(D''_{1:t})$ (two examples will follow shortly). The Δ_l and Δ_h functions are then plugged into Theorem 4.1, and the bounds on worst-case losses follow.

$\boldsymbol{C(t) = \frac{k_s(D''_{1:t})}{F_s}}$ I use Lemma 7 applied to $k_{s'.i}(D''_{1:t})$ to substitute $k_{s'}(D''_{1:t})$ out of Equation 5.11. The Δ_l and Δ_h functions follow:

$$\begin{cases} \Delta_l(s'.i) & = \frac{F_{s'.i}}{F_{s'}}\left[(\bar{l}''_{[-s]})_{s'} - (T''_{[-s]})_{s'}\overline{\theta} + \sum_{s'.j}\theta_{s'.j} - n^{\star}_a(s) + 1\right] - \theta_{s'.i} \\ \Delta_h(s'.i) & = \frac{F_{s'.i}}{F_{s'}}\left[(\overline{u}''_{[-s]})_{s'} - (T''_{[-s]})_{s'}\overline{\theta} + \sum_{s'.j}\theta_{s'.j} - 1\right] + 1 - \theta_{s'.i}. \end{cases} \tag{5.12}$$

$\boldsymbol{C(t) = \min_s \frac{k_s(D''_{1:t})}{F_s}}$ It follows from Equation 5.11 that:

$$0 \leq \frac{k_{s'}(D''_{1:t})}{F_{s'}} - \min_s \frac{k_s(D''_{1:t})}{F_s} \leq \frac{1}{F_{s'}} \max_s \left[(\overline{u}''_{[-s]})_{s'} - (T''_{[-s]})_{s'} \overline{\theta} \right]$$

$$0 \leq k_{s'}(D''_{1:t}) - F_{s'} \min_s \frac{k_s(D''_{1:t})}{F_s} \leq \max_s \left[(\overline{u}''_{[-s]})_{s'} - (T''_{[-s]})_{s'} \overline{\theta} \right].$$

I use the same substitution step as in the previous example, and obtain the following:

$$\begin{cases} \Delta_l(s'.i) & = \frac{F_{s'.i}}{F_{s'}} \left[\sum_{s'j} \theta_{s'.j} - n^\star_a(s') + 1 \right] - \theta_{s'.i} \\ \Delta_h(s'.i) & = \frac{F_{s'.i}}{F_{s'}} \left\{ \max_s \left[(\overline{u}''_{[-s]})_{s'} - (T''_{[-s]})_{s'} \overline{\theta} \right] + \sum_{s'j} \theta_{s'.j} - 1 \right\} + 1 - \theta_{s'.i}. \end{cases} \tag{5.13}$$

5.3.5 Choosing θ

Note that the Δ_h and Δ_h functions for $C(t) = \frac{k_s(D''_{1:t})}{F_s}$ and $C(t) = \frac{k_{s.i}(D''_{1:t})}{F_{s.i}}$ are linear functions of the θ variables. Thus, the following LP formulation can be used for these $C(t)$ functions:

Maximize WL subject to:

$$\text{WL} \leq \sum_s \sum_{s.i} \Delta_l(s.i) X^+_{s.i,b} + \sum_s \sum_{s.i} \Delta_h(s.i) X^-_{s.i,b} \qquad (\forall b \in \mathbb{B})$$

$$0 \leq \theta_a \leq 1. \qquad (\forall a \in [1 \ldots n^\star_a])$$

where the corresponding functions (linear in θ variables) are substituted for $\Delta_l(s.i)$ and $\Delta_h(s.i)$ (e.g. the right hand sides of the Equations 5.12 for $C = \frac{k_s(D''_{1:t})}{F_s}$).

Note that for $C(t) = \min_s \frac{k_s(D''_{1:t})}{F_s}$, Δ_h is not a linear function (Equation 5.13), so it cannot

be used directly in a linear program. However, Δ_h can be rewritten as:

$$\Delta_h(s'.i) = \frac{F_{s'.i}}{F_{s'}} \left\{ \max_s \left[(\overline{u}''_{[-s]})_{s'} - (T''_{[-s]})_{s'} \overline{\theta} \right] + \sum_{s'j} \theta_{s'.j} - 1 \right\} + 1 - \theta_{s'.i}$$

$$= \frac{F_{s'.i}}{F_{s'}} \left\{ M_{s'} + \sum_{s'j} \theta_{s'.j} - 1 \right\} + 1 - \theta_{s'.i} \tag{5.14}$$

where $M_{s'}$ is a new variable:

$$M_{s'} = \max_s (\overline{u}''_{[-s]})_{s'} - (T''_{[-s]})_{s'} \overline{\theta}. \tag{5.15}$$

$M_{s'}$ can be expressed with $n^\star_a$ linear constraints of the form:

$$M_{s'} \geq (\overline{u}''_{[-s]})_{s'} - (T''_{[-s]})_{s'}. \tag{5.16}$$

I introduce the following LP formulation for $C(t) = \min_{s.i} \frac{k_{s.i}(D''_{1:t})}{F_{s.i}}$, using additional variables M_s for each state s in the F-induced graph:

Maximize WL subject to:

$$\text{WL} \leq \sum_s \sum_{s.i} \Delta_l(s.i) X^+_{s.i,b} + \sum_s \sum_{s.i} \Delta_h(s.i, M_s) X^-_{s.i,b} \qquad (\forall b \in \mathbb{B})$$

$$M_s \leq M_s(\overline{\theta}) \qquad (\forall s)$$

$$0 \leq \theta_a \leq 1. \qquad (\forall a \in [1 \ldots n^\star_a])$$

The notation $M_s(\overline{\theta})$ refers to the right hand side of Equation 5.16.

Similar formulations can be written for $C(t) = \max_s \frac{k_s(D''_{1:t})}{F_s}$, $C(t) = \min_{s.i} \frac{k_{s.i}(D''_{1:t})}{F_{s.i}}$,

$C(t) = \max_{s.i} \frac{k_{s.i}(D''_{1:t})}{F_{s.i}}$, $C(t) = \min_{s.i} \frac{k_{s.i}(D''_{1:t})+\theta_{s.i}}{F_{s.i}}$, and $C(t) = \max_{s.i} \frac{k_{s.i}(D''_{1:t}+\theta_{s.i})}{F_{s.i}}$.

Reducing the Size of the LPs Note that θ values are irrelevant for edges that do not have siblings in the F-induced graph. The θ variables for such edges can be eliminated from the LPs.

If $X_{s.i,b} < 0 \; \forall b$, then $\Delta_l(s.i)$ and its defining constraints can be eliminated. Similarly, if $X_{s.i,b} > 0 \; \forall b$, then $\Delta_h(s.i)$ and its defining constraints can be eliminated.

Initial State If a sequence of actions converges to some action frequency profile F, that is enough to determine the at-the-limit utility profile U. In particular, the choice of initial state is irrelevant. This is not true, however, for WL.

The WL bounds derived in this section hold as long as **the initial state is in the F-induced graph**. This is not an unreasonable assumption for task-division applications where one can choose the initial state. When assigning classes to professors, there is no restriction on the assignment that can be used in the first semester.

When **the initial state is not in the F-induced graph**, one can look for a finite path (in the original graph) from the initial state to some state in the F-induced graph. Such a path doesn't have to be a simple: some actions can be used several times. One can compute an lower bound on the loss of some beneficiary b by walking backwards from the F-induced graph to the initial state and adding that beneficiary's X values to the WL_b computed for the F-induced graph. Finding the best way (with respect to WL) to reach the F-induced graph from the initial state (in a number of steps not exceeding some parameter λ) is NP-hard; one can trivially prove this with a reduction very similar to [133]. I leave further investigating this issue as future work.

5.4 Multi-objective Optimization

I have shown in Section 5.2 that optimizing the limit-point utility profile may lead to unbounded losses, and I have argued for a multi-objective optimization approach. The two objectives are the long-term (i.e. at the limit) utility profile and WL. In this section I introduce a methodology for finding a set of F profiles corresponding to different utility-versus-WR tradeoff points. Rather than imposing an a priori set of weights on the two objectives, the proposed methodology would produce a large number of tradeoff points for the decisions maker to choose from.

Scalarizations There is a significant body of work in the multi-objective optimization literature on "scalarizations" [37, 48]. The approach consists of solving multiple single-objective optimization problems, where the objective is a combination of the original objectives, and the weights are varied between problems to discover different tradeoff points. Alternatively, one would optimize one of the objectives while setting bounds for the remaining objectives. Finding the best long-term utility profile with WL no worse than a given threshold value seems particularly appealing for the problem at hand.

In Section 5.3 I presented a linear program-based method for computing WL bounds for a given *valid*[9] long-term action frequency profile F. In these LP formulations the components of F are constants, and the components of $\overline{\theta}$ are variables. A scalarization approach would have the components of both F and $\overline{\theta}$ be variables, leading to a quadratic constrained mathematical program. A work-around would be to eliminate the θ variables by using $\mathbf{GF^0}$ (or $\mathbf{GF^1}$) instead of $\mathbf{GF^\theta}$.

Although inverting matrix $Q_{[-s]}$ and enforcing Conditions 2 can easily be achieved with linear constraints, I could only enforce Condition 3 (i.e. the strong connectivity of the F-induced graph) using quadratic constraints, and not linear one. Additionally, $X_{s.i,b}$ are variables (since U_b depends on all F variables and $X_{s.i,b}$ depends on U_b), so computing WL

[9] An F profile is valid if it satisfies Conditions 1, 2 and 3.

also requires quadratic constraints. Furthermore, the $X_{s.i,b}$ variables introduce a different type of non-linearity (since $X_{s.i,a}$ is multiplied by Δ_l when $X_{s.i,a} < 0$ and it is multiplied by Δ_h), and I could only deal with it by using mixed-integer constraints.

One cannot easily solve such a mathematical program, since *quadratic constraint programming* (QCP) is NP-Hard [66, p. 245] (and *integer quadratic constraint programming* (IQCP) is actually intractable). Therefore the scalarization-based methods [48] (and also Lagrangian methods) seem intractable, since, in spite my best efforts, each scalarization is an QCP (IQCP).

5.4.1 The Proposed Approach

Based on this evidence of the hardness of the problem, I propose an evolutionary computation[10] based approach to search the space of F profile. The specifics of the approach are listed bellow:

Building the Pareto-front There could be an infinity of Pareto-undominated solutions, and one has to pick a finite subset that is representative of the entire Pareto-front. Therefore, one has two goals when building a Pareto-front: (1) finding solutions that as as close as possible to the "true" Pareto-front, and (2) keeping a "well-spread" set of solutions (also referred to as "crowd-control"). In my experiments I use the SPEA-II paradigm [195], which seems to perform quite well in the literature. SPEA-II uses a fixed-sized population $\mathcal{P}$ (to store newly generated individuals) and a fixed-sized "archive" $\mathcal{A}$ (to maintain the

[10] Evolutionary Computation [39] is concerned with the study of evolutionary algorithms, which are optimization algorithms based on the Darwinian principle of natural selection. Evolutionary algorithms start with randomly generated solutions, then repeatedly generate new solutions from existing solutions. Candidate solutions are typically referred to as *individuals*, and are obtained from applying *genetic operators* to *parents* extracted from a *population* of solutions. In order to escape from local optima, the *selection* of parents is typically a stochastic process biased towards preferring individuals with better *fitness* (i.e. better, more fit individuals).

Evolutionary algorithms are very versatile function optimizers, as they require very little domain information (e.g. functions do not have to be differentiable or continuous). This makes evolutionary algorithms suitable for black-box function optimization, which means the evolutionary approach proposed here can be used for any social welfare function.

Pareto-front). The original specifications call for $\mathcal{A}$ to be initially empty, but in the particular implementation I used [99] both $\mathcal{P}$ and $\mathcal{A}$ are initialized with randomly-generated individuals (the different is likely to be of little consequence). At subsequent generations $\mathcal{P}$ is filled with new individuals generated by genetic operators from parents chosen *exclusively* from $\mathcal{A}$. Next, the content of $\mathcal{A}$ is replaced by individuals from $\mathcal{P} \cup \mathcal{A}$ that are not Pareto-dominated by other individuals in $\mathcal{P} \cup \mathcal{A}$. If there are fewer Pareto-undominated individuals than $|\mathcal{A}|$, the archive is padded with Paredo-dominated individuals. However, if there are more Pareto-undominated individuals than $|\mathcal{A}|$, SPEA-II uses the following crowd-control mechanism to retain exactly $|\mathcal{A}|$ individuals. An individual's crowdedness (called "density" in SPEA-II) is assessed by computing the distance to the k^{th} closest individual ($k = \sqrt{|\mathcal{P}| + |\mathcal{A}|}$). Note that the distance is measured in the fitness space (i.e. in the objectives space).

Objectives In my experiments I used two objectives: $min_b U$ and WL. I used maximin rather than leximin because finding a meaningful distance measure would be non-trivial for leximin, which "does not introduce directly any scalar measure" [117]. However, note that in this experimental setup one can replace $min_b U$ with any other *scalar* social welfare measure. The same goes for aggregating the worst losses of all beneficiaries: e.g. $OWA(\mathrm{WL}_1, \ldots, \mathrm{WL}_{n_b})$ instead of $\mathrm{WL} = \min_b \mathrm{WL}_b$.

In order to compute WL for a given F, I compute a WL lower bound (using Theorem 4.1 and the LP formulations described in Section 5.3.5) for each of the following $C(t)$ functions, and keep the best one: $C(t) = \frac{k_s(D''_{1:t})}{F_s}$,$C(t) = \min_s \frac{k_s(D''_{1:t})}{F_s}$, $C(t) = \max_s \frac{k_s(D''_{1:t})}{F_s}$, $C(t) = \frac{k_{s.i}(D''_{1:t})}{F_{s.i}}$, $C(t) = \min_{s.i} \frac{k_{s.i}(D''_{1:t})}{F_{s.i}} F_{s.i}$, $C(t) = \max_{s.i} \frac{k_{s.i}(D''_{1:t})}{F_{s.i}} F_{s.i}$, $C(t) = \min_{s.i} \frac{k_{s.i}(D''_{1:t}) + \theta_{s.i}}{F_{s.i}} F_{s.i}$, and $C(t) = \max_{s.i} \frac{k_{s.i}(D''_{1:t}) + \theta_{s.i}}{F_{s.i}} F_{s.i}$.

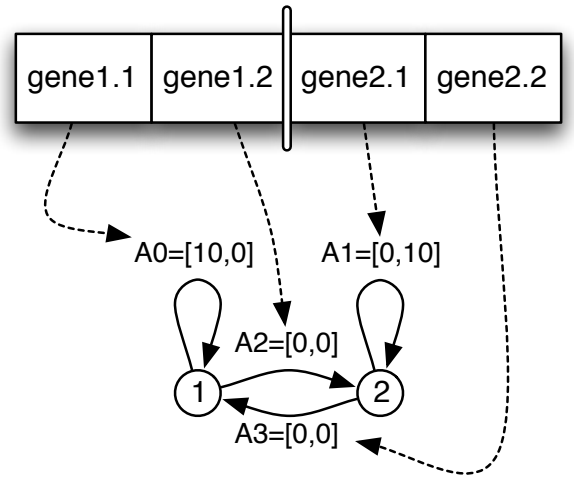


Figure 5.5: Genome corresponding to the SCC in Figure 5.4. The genome consists of two chunks, and the double line shows where the first one ends and the second begins.

Representation In evolutionary computation, an individual's *genotype* refers to the representation of the genetic material (computer encoding) it receives from its parents; and individual's *phenotype* refers to the individual's functionality, and it is the base for the individual's fitness evaluation. In this case the phenotype is a strongly-connected F-induced graph.

The simplest **genotype** representation in this problem is a collection of values (genes), one for each action $s.i$ (i.e. one gene for each component of F). Algorithmically, one can see the genotype as the adjacency matrix representation of the original graph, with weights dictated by the F profile. A more traditional way (in evolutionary computation, specifically genetic algorithms) to view the same representation is as a fixed-length array of genes. I refer to each group of genes corresponding to sibling actions as a *chunk*. The genome consists of chunks of genes.

Example 5.3. I continue the professors and classes running example by discussing the genome (see Figure 5.5) for the SCC introduced in Example 5.2. The genome consists of two *chunks* (corresponding to the two nodes in the SCC), and each chunk consists of two genes, corresponding to the two out-going edges each node has in the SCC.

Enforcing constraints on the genotype space is rather difficult in evolutionary computation, and it requires special procedures. Alternatives include penalizing the infeasible individual, attempting a local repair, or using such a representation (and genetic operators) that all generated individuals are guaranteed to be feasible.

Rather than enforcing Conditions 1, 2 and 3 on the genotype, I incorporate them into the procedure decoding the genotypes into phenotypes. Intuitively, *the gene values in each chunk specify the relative proportions of sibling actions in F*.

- First, I normalize the values in each chunk ($LF_{s.i} = \frac{gene_{s.i}}{\sum_{s.i} gene_{s.i}}$). These value are well defined provided no chunk consists of only zeros. To this end, I "repair" each such a chunk by setting a random gene in it to 1; also, I ensure all gene values are non-negative.

- Second, I solve the following linear program in order to find F such that $F_{s.i} = LF_{s.i} \times F_s$ for all states s in the F-induced graph:

$$\sum_{s \xrightarrow{s.i} s'} \alpha_s LF_{s.i} = \alpha_{s'} \qquad (\forall s')$$

$$\sum_s \alpha_s = 1$$

$$\alpha_s \geq 0. \qquad (\forall s)$$

  This LP solves the system of equations $\overline{\alpha}^\top = \overline{\alpha}^\top P$ from Section 5.3.3 ($P_{s,s'} = \sum_{s \xrightarrow{s.i} s'} \frac{F_{s.i}}{F_s} = \sum_{s \xrightarrow{s.i} s'} LF_{s.i}$), for which I showed a solution always exists. Note that the resulting F (defined as $F_{s.i} = \alpha_s LF_{s.i}$) satisfies both Condition 1 and Condition 2.

- In order to enforce Condition 3, one could try to force an individual to map into an F profile that induces a strongly connected graph, or repair it by adding and removing edges. My approach is to extract each strongly connected component in the F-induced

graph and evaluate it as a separate individual.[11] In other words, a genotype can have multiple phenotypes. Also, multiple genotypes can share a phenotype. Imagine a genotype consisting of 2 strongly connected components (i.e. phenotypes); by changing a value in one of the genes, one changes at most one phenotype; the new genotype is different from the old one, yet they share one of the phenotypes. In my approach, duplicate phenotypes are always removed from the population.

Operators An individual is generated by one of the following six genetic operators:

1. **Gaussian Mutation** Each gene is mutated with probability p_m. The gene value *zero* receives special treatment, since it has a special effect on the phenotype (if $gene_{s.i} = 0$, then $F_{s.i} = 0$, which essentially removes edge $s.i$ from the F-induced graph). With probability p_z a non-zero gene value will turn into a zero, and with probability p_{nz} probability a zero gene value will be replaced by a non-zero value. A non-zero value is obtained by adding Gaussian noise (with standard deviation σ_m) to the old value. If the new value is outside the range $[0, 1]$, then another attempt is made. This step is attempted up to 100 times, after which the gene is set to zero.

2. **Reset Mutation**: This operator makes a child from scratch, without inheriting anything from the parent. Also, the first generation of individuals is created using this genetic operator.

   Each gene is set to zero with probability $p_z/(p_z + p_{nz})$, and to $-log(u_{[0,1]})$ otherwise. The expression $p_z/(p_z + p_{nz})$ was chosen because it represents the stationary probability for a gene to have value zero if Gaussian Mutation were to be applied to it over and over again. The notation $u_{[0,1]}$ refers to a random value chosen uniformly from

[11] The decision of investigating each strongly connected component in the F-induced graph is motivated by my focusing on the case where the decision maker is allowed to choose the initial state (see discussion on page 97, at the end of Section 5.3.5).

If the initial state is fixed (i.e. specified in the problem's input), then one would only consider the SCC of the F-induced graph containing the initial state. If the initial state is not part of the F-induced graph, one could assign the individual the lowest possible WL value, or attempt to "repair" it: e.g. by adding action (i.e. replacing zeros with positive values) until the initial state becomes part of a loop.

the $[0,1]$ interval, and setting coordinates to $-log(u_{[0,1]})$ (followed by normalization) is one of the ways to pick uniformly distributed points in a unit simplex [42]. This is relevant because the gene values in each chunk will be normalized, so the set of values in each chunk represents a point in a unit simplex. By using $-log(u_{[0,1]})$ values, one makes a step towards generating uniformly distributed individuals.

Crossover operators receive two parent genotypes and produce two children genotypes. The next two crossover operators generate the children by mix-and-matching genetic material from the parents. Both crossovers are *chunk-based*, which means a child inherits all genes in a chunks from the same parent (note that such crossovers become simple cloning operators when there is a single state).

3. **Chunk-based Uniform Crossover** With this operator, a child is created by inheriting each chunk with 50-50 probability from either parent. Formally:

$$u_s = u_{[0,1]}$$

$$child_1.gene_{s.i} = \begin{cases} parent_1.gene_{s.i} & \text{if } u_s \geq 0.5 \\ parent_2.gene_{s.i} & \text{otherwise} \end{cases}$$

$$child_2.gene_{s.i} = \begin{cases} parent_2.gene_{s.i} & \text{if } u_s \geq 0.5 \\ parent_1.gene_{s.i} & \text{otherwise.} \end{cases}$$

4. **Topology-based Crossover** Rather than choosing uniformly which chunks to inherit from each parent, this chunk-based crossover makes the decision using the topology of the graph. Specifically, it builds a subset of states (chunks) V, starting from a random state and iteratively adding random states that can be reached directly from any of states already in V. The size of V is a random value from a binomial distribution:

$P(|V| = i) = \binom{n}{i} p^i (1-p)^{n-i}$, with n equal to the number of states and $p = 0.5$.

The state partition generated by this V decides which chunks a child gets from one parent and which chunks it gets from the other parent.

$$child_1.gene_{s.i} = \begin{cases} parent_1.gene_{s.i} & \text{if } s \in V \\ parent_2.gene_{s.i} & \text{otherwise} \end{cases}$$

$$child_2.gene_{s.i} = \begin{cases} parent_2.gene_{s.i} & \text{if } s \in V \\ parent_1.gene_{s.i} & \text{otherwise.} \end{cases}$$

Unlike the Chunk-based Uniform Crossover, this operator ensures that children inherit entire subgraphs from at least one of their parents, which, intuitively, should make this operator less destructive. The binomial distribution was used such that the distribution of the number of chunks a child gets from one parent is the same under both crossovers; so the two operators can be compared on equal footing.

The two crossover operators so far create children by distributing the gene values of the two parents to the two children. The next two operators are *blending* crossovers: a child's gene value is a function of the corresponding gene values of both parents.

5. **SBX** stands for Simulated Binary Crossover [40]. It uses a different *blending* coefficient

$\beta_{s.i}$ for each action $s.i$ (based on $u_{s.i} = u_{[0,1)}$):

$$\beta_{s.i} = \begin{cases} (2u_{s.i})^{\frac{1}{n_c+1}} & \text{if } u_{s.i} \leq 0.5 \\ \frac{1}{2(1-u_{s.i})}^{\frac{1}{n_c+1}} & \text{otherwise} \end{cases} \tag{5.17}$$

$$child_1.gene_{s.i} = \frac{1}{2}\left[(1+\beta_{s.i})parent_1.gene_{s.i} + (1-\beta_{s.i})parent_2.gene_{s.i}\right]$$

$$child_2.gene_{s.i} = \frac{1}{2}\left[(1-\beta_{s.i})parent_1.gene_{s.i} + (1+\beta_{s.i})parent_2.gene_{s.i}\right]$$

These steps are repeated until the $gene_{s.i}$ values for both children are inside the $0 \ldots 1$ range. After 100 unsuccessful tries, my approach uses blending coefficient $\beta_{s.i} = u_{[-1,1]}$, which makes the children's gene values linear combinations of the parent gene values (which are normalized, and thus guaranteed to be inside the $0 \ldots 1$ range).

6. **Modified SBX** I introduce a modified version of the SBX operator, which uses the same blending coefficient β for all genes. In order to motivate this choice, imagine two parent genotypes with no gene equal to zero; the parents' phenotypes are F and F', and the corresponding utility profiles are U and U'. Each child produced by this operator would have a phenotype $F'' = \frac{1+\beta}{2}F + \frac{1-\beta}{2}F'$, and an at-the-limit utility profile $U'' = \frac{1+\beta}{2}U + \frac{1-\beta}{2}U'$. Assuming neither of U and U' Pareto-dominates the other and $-1 \leq \beta \leq 1$, then $U''_b \geq \min(U_b, U'_b)$, so all beneficiaries will prefer U'' to one of U and U' (some of them will prefer U'' to U, some to U'). Intuitively, this line of reasoning generalizes to situations when the set of genes with value zero is the same for the two parent genotypes. In a nutshell, this operator has to improve the social welfare measure by combining F profiles that give preferential treatment to different beneficiaries.

$$child_1.gene_{s.i} = \frac{1}{2}\left[(1+\beta)parent_1.gene_{s.i} + (1-\beta)parent_2.gene_{s.i}\right]$$

$$child_2.gene_{s.i} = \frac{1}{2}\left[(1-\beta)parent_1.gene_{s.i} + (1+\beta)parent_2.gene_{s.i}\right]$$

The value β comes from the same distribution as before (Equation 5.17), but in order to ensure that all children's genes are in the $0 \ldots 1$ range, β must be in the interval $[LB'_\beta, UB'_\beta]$, where:

$$LB_\beta = \max\left\{ \frac{parent_2.gene_i}{parent_2.gene_i - parent_1.gene_i} \middle| parent_2.gene_i < parent_1.gene_i \right\}$$

$$UB_\beta = \min\left\{ \frac{parent_2.gene_i}{parent_2.gene_i - parent_1.gene_i} \middle| parent_2.gene_i > parent_1.gene_i \right\}$$

$$LB'_\beta = \max\{1 - UB_\beta, LB_\beta\}$$

$$UB'_\beta = \min\{UB_\beta, 1 - LB_\beta\}.$$

If $\beta \notin [LB'_\beta, UB'_\beta]$ after 100 tries, then $\beta = u_{[-1,1]}$.

Operators are repeatedly selected at random, and several probability distributions over these operators are considered.

Statistics Many measures have been proposed in the literature to gauge the performance of multi-objective optimization algorithms. Some measures (e.g. *generational distance*, *error ratio*[12], and the *Chi-Square-like deviation* [40, 67]) require knowledge of the true Pareto-front, which is not readily available for the problem at hand.

Another widely used measure is the *hypervolume* ([40, 195]), which measures both closeness to the true Pareto-front and how well spread the produced front it. Hypervolume

[12]Error ratio refers to the fraction of points in the produced front that are not on the true (ideal) Pareto-front.

measures the surface under the *attainment surface*.[13] One shortcoming of this measure is that it is sensitive to the scales of the objectives (i.e. it favors an objective that is orders of magnitude smaller than the other). Moreover, the true Pareto-front might not be bounded, so one cannot easily fix this just by scaling the objectives.

The most suitable measure is *set-cover* [40, 196]: it does not require the true Pareto-front, and it is insensitive to the objectives having different scales.

Set-cover is a pairwise comparison measure: it compares two multi-objective methods by assessing the percentage of solutions produced by one method that are weakly Pareto-dominated (and thus can be discarded) by solutions produced by the other method. Let M' and M'' be the Pareto-fronts generated by two methods. Formally, the set-cover $\mathcal{C}$ is defined as:

$$\mathcal{C}(M', M'') = \frac{|\{a'' \in M'' | \exists a' \in M' : a' \text{ weakly Pareto-dominates } a''\}|}{|M''|} \tag{5.18}$$

Note that $\mathcal{C}(M', M'') \in [0, 1]$, and that the higher the value of $\mathcal{C}(M', M'')$, the more solutions in M'' are made obsolete by M''. Also, note that $\mathcal{C}(M', M'')$ and $\mathcal{C}(M'', M')$ do not necessarily sum to one, so both should be computed.

5.5 Validating the Proposed Approach

5.5.1 Research Questions

The main question in this chapter is "**Q**: Can the proposed approach successfully find Pareto-fronts well spread and close to the true front?" However, it is difficult to investigate it since the true Pareto-fronts are available only for the simplest of problems. In lieu of addressing this question, I investigate a weaker version of it: "**Q1**: Is the proposed approach superior to random search?" This is an acceptable proof-of-concept methodology in evolutionary

[13] For two objectives, attainment surface is the curve connecting the points in the produced Pareto-front. Note that this curve consists exclusively of vertical and horizontal line segments.

computation.

The proposed approach is based on a customized representation (rather than a universal encoding, such as binary strings), and so the genetic operators used to manipulate it are not yet thoroughly understood. Therefore, a secondary research question refers to whether the genetic operators presented in Section 5.4.1 are a good choice in the context of this problem and representation. Out of the six genetic operators introduced in Section 5.4.1, the four crossover operator have similar functions, so it makes sense to investigate "**Q2** Is any of the crossover operators better than the others?"

I order to address these research questions, I investigate six versions of the proposed approach corresponding to six different combinations of genetic operators. The notation 'a-b-c-d-e-f' refers to the un-normalized proportions in which the six operators are used in a particular setup.

Random-search (or 0-1-0-0-0-0): all individuals are created using Reset-Mutation;

5-1-1-1-1-1: when a genetic operator is picked, there is a 50% probability of using Gaussian-Mutation and 10% probability for each of the other five genetic operators;

1-0-1-0-0-0: Gaussian-Mutation and Chunk-based Uniform Crossover are used with probability 50%;

1-0-0-1-0-0: Gaussian-Mutation and Topology-based Crossover are both used with probability 50%;

1-0-1-0-0-0: Gaussian-Mutation and SBX are both used with probability 50%;

1-0-0-1-0-0: Gaussian-Mutation and Modified SBX are both used with probability 50%.

5.5.2 Experimental Setup

I tested the six treatments one three (increasingly harder) problem instances Example B.1 (3 states, 6 actions, 3 beneficiaries), Example B.2 (5 states, 25 actions, 6 beneficiaries),

and Example B.3 (5 states, 45 actions, 4 beneficiaries). The input data is presented in Appendix B.1. The following parameter values were used:

- $|\mathcal{A}| = |\mathcal{P}| = 25$;
- Run length: 400 generations for the first two test problems (Example B.1 and Example B.2), and 500 generations for the last one (Example B.3);
- Mutation parameters: $p_m = 1/L$ (where L is the length of the genome, i.e. the number of genes in all the chunks), $p_z = 10\%$, $p_{nz} = 75\%$, and $\sigma_m = 0.125$;
- SBX: $n_c = 3$.
- Operators select parents from the archive using Tournament-Selection of size 2 (i.e. each parent is the best of two randomly selected individuals).

Each treatment was run 90 times on each test problem. In this experiment I followed the example of [196]: there were 90 randomly generated pairs $\mathcal{P}$ and $\mathcal{A}$ for each test problem, and each treatment was ran on each of those $\mathcal{P}$ and $\mathcal{A}$ pairs. That it, every time $\mathcal{C}(M', M'')$ was computed (Equation 5.18), M' and M'' were obtained from the same initial $\mathcal{P}$ and $\mathcal{A}$.

I point out that there is no established methodology in the literature for comparing two algorithms in a statistically-significant manner using the set-cover measure. I discuss two possible criteria:

- If $\mathcal{C}(M', M'') = 1$ and $\mathcal{C}(M'', M') = 0$ for every pair (M', M'') of Pareto-fronts produced by algorithms A' and A'', then clearly algorithm A' outperforms A''. However, this criterion is too strong:
  - If a state graph contains self loops, then there is a simple way to find at least one point on the true Pareto-front.[14] It follows that any two algorithms A' and A''

[14] For each self loop $s.i$, there is a valid F profile such that $F_{s.i} = 1$, and zero everywhere else. The F-induced graph consists of state s and action $s.i$, and it corresponds to repeatedly executing action $s.i$, which means $U_b = R_b(s.i)$ and WL $= 0$. One can iterate over all self loops and select the one with the best social welfare

that manage to find this trivial point are indistinguishable by this criterion (since $\mathcal{C}(M', M') > 0$ and $\mathcal{C}(M'', M') > 0$).

  - If $\mathcal{C}(M', M'') = 0.99$ and $\mathcal{C}(M'', M') = 0.01$ for a few (M', M'') pairs and $\mathcal{C}(M', M'') = 1$ and $\mathcal{C}(M'', M') = 0$ for all other pairs, then algorithm A' is *intuitively* superior to algorithm A''. I argue that even if $\mathcal{C}(M', M'') = 0.99$ and $\mathcal{C}(M'', M') = 0.01$ for all pairs, algorithm A' would still be *intuitively* superior to algorithm A''.

- Another criteria for preferring algorithm A' over algorithm A'' might be $\mathbf{E}[\mathcal{C}(M', M'')] > \mathbf{E}[\mathcal{C}(M'', M')]$ (e.g. [65]). Theoretically, one can make statistically-significant statements based on t-Tests or confidence intervals, but this is not the case here, because many of the $\mathcal{C}$ sample distributions I obtained are not normally distributed.

I use the following approach to produce statistically-significant results: First, I use a non-parametric approach, by going for the median rather than the mean [184]. Second, I point out that rather than comparing the distributions of $\mathcal{C}(M'', M')$ and $\mathcal{C}(M', M'')$ values, one should compare the distributions of $\Delta\mathcal{C}(M'', M')$ values, defined as:

$$\Delta\mathcal{C}(M', M'') = \mathcal{C}(M', M'') - \mathcal{C}(M'', M') \tag{5.19}$$

since the Pareto-fronts M' and M'' were obtained with the same initial population.

5.5.3 Empirical Results

The results for the pair-wise treatment comparisons using the $\mathcal{C}$ measure are presented in Figure 5.6. The box plots for $\Delta\mathcal{C}$ are presented in Figure 5.7. Note that unlike $\mathcal{C}$, $\Delta\mathcal{C}$ is

component (the one with the largest $\min_b U_b$ in this case). The tradeoff point corresponding to this loop is guaranteed to be on the true Pareto-front: (1) it has the best social welfare out of all tradeoff points with WL $= 0$, and (2) all other F-induced graphs (consisting of 2 or more edges) have WL < 0.

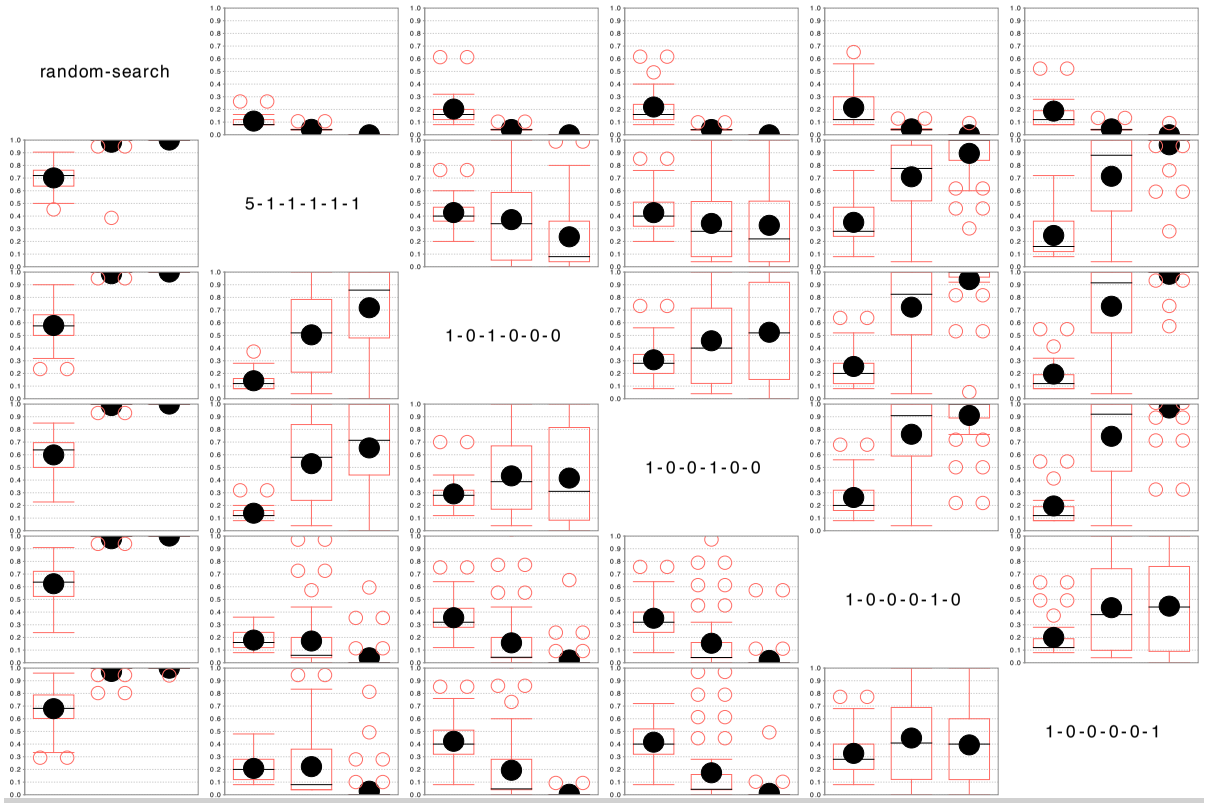


Figure 5.6: Box plots of the $\mathcal{C}$ (Equation 5.18) results for pairwise-comparison of six treatments, using three different test problems. Each of the entries in the 6-by-6 tableau contains the results for the comparison of two treatments corresponding to the row and column of the entry. Intuitively, the larger the values at entry (r, c), the more treatment (r, r) is able to improve on the solutions found by treatment (c, c). The three box plots in each entry refer to the 3 problems: Example B.1, Example B.2, and Example B.3 respectively. Each box plot is a distribution of set-cover values over 90 runs. A box plot shows the average (filled circle) the quartiles Q1, Q3 (the box), the median (the horizontal line inside the box), the min and max (the horizontal lines outside the box), and the outliers (the empty circles).

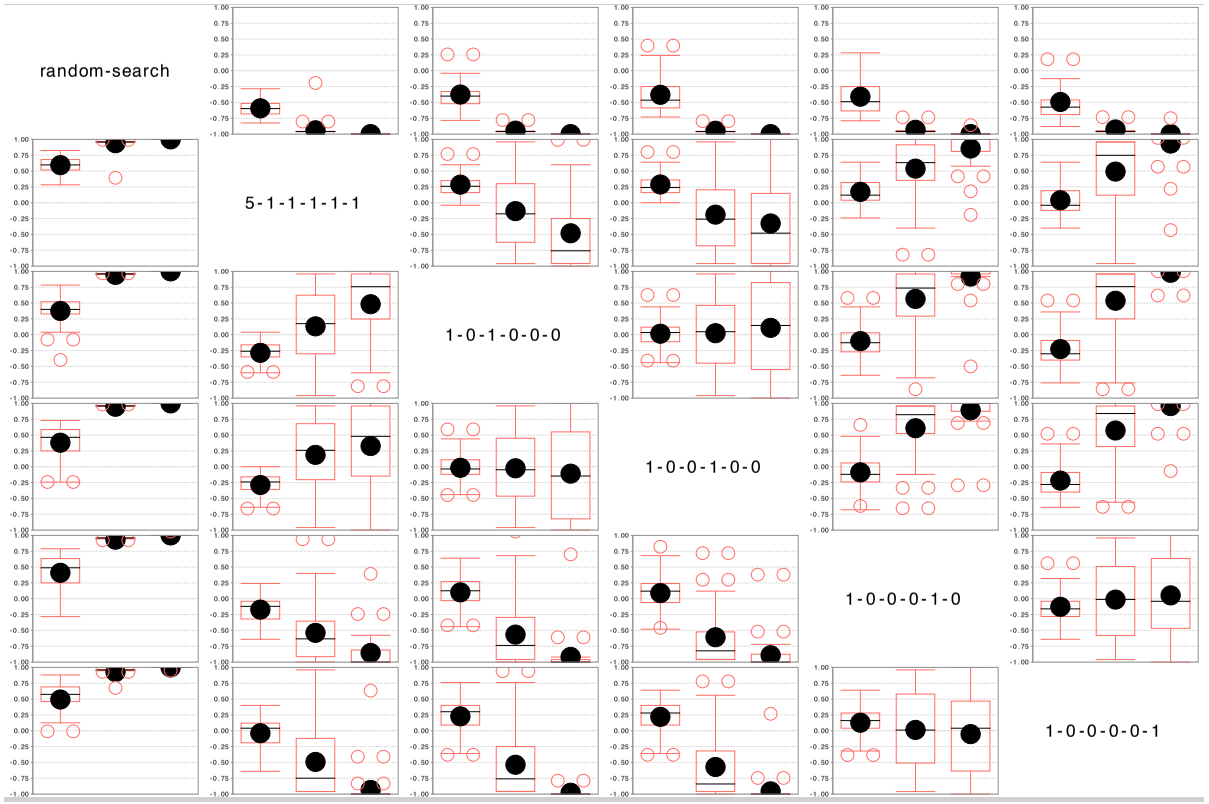


Figure 5.7: Box plots of the $\Delta\mathcal{C}$ (Equation 5.19) results for pairwise-comparison of six treatments, using three different test problems. Each of the entries in the 6-by-6 tableau contains the results for the comparison of two treatments corresponding to the row and column of the entry. Positive values at entry (r, c) mean treatment (r, r) is able to improve on the solutions found by treatment (c, c) more than treatment (c, c) is able to improve on the solutions found by treatment (r, r).

anti-symmetric, so the plots above the first diagonal are up-side-down versions of the plots bellow the diagonal.

A visual analysis shows that all treatments outperform Random-Search. First, the $\mathcal{C}(\cdot, \text{Random-Search})$ values (i.e. the first row of Figure 5.6) are rather small, indicating that few of the solutions produced by other treatments are weakly Pareto-dominated by the solutions produced by Random-Search. Also, the $\mathcal{C}(\text{Random-Search}, \cdot)$ values are rather large, which indicates that most solutions found by Random-Search are weakly Pareto-dominated by the treatments. These results are stronger for the last two (larger and more difficult) problems: all medians for $\mathcal{C}(\text{Random-Search}, \cdot)$ are 1, which means that in more than half the runs Random-Search was unable to find a single solution that the other treatments did not weakly Pareto-dominate.

Figure 5.7 paints a similar picture: both mean and median for all plots on the first row are negative. For 5-1-1-1-1-1 in particular, *all* $\Delta\mathcal{C}(\text{Random-Search}, \text{5-1-1-1-1-1})$ values are negative, meaning that $\mathcal{C}(\text{5-1-1-1-1-1}, \text{Random-Search}) > \mathcal{C}(\text{Random-Search}, \text{5-1-1-1-1-1})$ in all 90 runs.

I will now prove that the claim "all treatments outperform Random-Search with respect to $\Delta\mathcal{C}$" is statistically significant. In Table 5.1 I present the confidence-intervals for the medians of $\Delta\mathcal{C}(\text{Random-Search}, \cdot)$ at 99% confidence level ($\alpha = 0.05$ for the entire table, and $\frac{0.01}{15}$ per confidence interval, by the Bonferroni correction). Since all confidence intervals are bellow zero, it follows that, with 99% confidence, all treatments outperform Random-Search with respect to $\Delta\mathcal{C}$.

Based on these results, I argue that the approach I propose is better than random search at producing a Pareto-front of long-term utility vs. worst-case loss tradeoff points. This concludes the answer to question **Q1**.

In the rest of the section I will disregard Random-Search and focus on the remaining five treatments. Table 5.2 contains the statistically-significant comparisons between all

Treatment	Example B.1	Example B.2	Example B.3
5-1-1-1-1-1	$[-0.647272, -0.545]$	$[-0.96, -0.958333]$	$[-1.0, -1.0]$
1-0-1-0-0-0	$[-0.49, -0.36]$	$[-0.96, -0.958333]$	$[-1.0, -1.0]$
1-0-0-1-0-0	$[-0.521818, -0.36]$	$[-0.96, -0.96]$	$[-1.0, -1.0]$
1-0-0-0-1-0	$[-0.586666, -0.38]$	$[-0.96, -0.956521]$	$[-1.0, -1.0]$
1-0-0-0-0-1	$[-0.641904, -0.506666]$	$[-0.96, -0.958333]$	$[-1.0, -1.0]$

Table 5.1: Confidence intervals for the median for $\Delta\mathcal{C}$ statistic comparing Random-Search to all other treatments. The confidence level is 99% for the entire table.

5-1-1-1-1	$>$	$=$	$<$	$>$	$=$	$=$	$>$	$>$	$>$	$=$	$>$	$>$
$<$	$=$	$>$	1-0-1-0-0-0			$=$	$=$	$=$	$<$	$>$	$>$	$<$	$>$	$>$
$<$	$=$	$=$	$=$	$=$	$=$	1-0-0-1-0-0			$<$	$>$	$>$	$<$	$>$	$>$
$<$	$<$	$<$	$>$	$<$	$<$	$>$	$<$	$<$	1-0-0-0-1-0			$<$	$=$	$=$
$=$	$<$	$<$	$>$	$<$	$<$	$>$	$<$	$<$	$>$	$=$	$=$	1-0-0-0-0-1		

Table 5.2: Statistically significant differences (based on confidence intervals for the median for $\Delta\mathcal{C}$) for each of the 3 problems. A '+' (plus sign) means the row treatment is statistically better than the column treatment at that problem, i.e. the confidence interval for $\Delta\mathcal{C}(\text{row}, \text{column})$ is above zero. Conversely, a '-' (minus sign) means the row treatment is statistically worse than the column treatment; and a '=' (equal sign) means the differences are not statistically significant. The confidence level is 99% for the entire table. The layout of the table is similar to that in Table 5.7. The confidence level is 99% for the entire table.

variations of my approach on each of the three problems. The overall confidence level is 99%, with a significance level of $\frac{0.01}{30}$ per comparison using the Bonferroni correction.

Note that the harder the problem, the worse the performance of the two SBX treatments (1-0-0-0-1-0 and 1-0-0-0-1-0) when compared against 5-1-1-1-1-1 and the Chunk-based Crossovers (1-0-1-0-0-0 and 1-0-0-1-0-0). The SBX treatments manage to perform relatively well on the first (easy problem), but are clearly outperformed on the harder problems. It seems that in spite of its theoretical properties [40], the Chunk-based Crossovers are more suitable in this context.

Now I focus on the remaining three treatments: 5-1-1-1-1-1 and the Chunk-based Crossovers (1-0-1-0-0-0 and 1-0-0-1-0-0). 5-1-1-1-1-1 has a good overall performance, but it is outperformed by the first Chunk-based Crossover (1-0-1-0-0-0) on the hardest problem. The performance of the Chunk-based Crossovers is fairly similar: good results for the harder

problems, but outperformed by the other there methods on the easy problem.

The answer to question **Q2** is that although the Chunk-based Crossovers outperform the two versions of SBX on the harder problems, more effort is needed to identify the best combination of genetic operators for this problem domain. This includes both more parameter configurations, and more test problems.

In conclusion, I have provided proof-of-concept that the approach proposed here is able to build Pareto-fronts of long-term utility vs. worst-case loss tradeoff points, but additional research is needed to fine-tune its performance.

Chapter 6: Controller Hierarchy

Multi-agent systems have a number of advantages over centralized decision making: some problems require inherently decentralized (flying robots operating under radio silence should be autonomous); some problems are so large that a centralized control would be unable to keep up with the real-time requirements (e.g. urban traffic control at multiple intersections). Multi-agent systems are more fault-tolerant, as the decision is distributed. Because of all these properties, multi-agent systems are harder to implement and optimize than centralized approaches [166]. Designing agents able to coordinate with others with only local information and limited communication, and able to gracefully handle interacting with an increasing (or decreasing) number of agents around them is often non-trivial.

In this chapter I use a specific multi-agent paradigm, the agent hierarchy. This structure is widely used in real-life military organizations, governments and enterprises. The optimal shape of an hierarchy (with respect to its information-processing agility) was studied extensively [140, 174, 175, 191, 192]. Agent hierarchies are intuitively appealing for multi-agent system for several reasons. First, the hierarchy can be used to restrict the number of agents an agent interacts with, so it should help a multi-agent system scale up to large numbers of agents. Second, an agent hierarchy naturally fits into the hierarchical problem decomposition paradigm, with direct applications to problems such as hierarchical task decomposition [43, 100, 104, 137, 157, 193] and allocation [1, 2].

6.1 Motivation

Example 6.1. I revisit the professors and classes running example used throughout this dissertation. In this version of the example, the department offers AI classes (which can

only be taught be AI professors), networking and operating systems classes (which can only be taught by OS professors), and general computer science classes (which everyone can teach). Each semester, the department head distributes the general classes to the two groups (the groups of AI and OS professors, respectively). Each group has a person in charge of assigning professors to classes (both group-specific and general classes). Therefore, this example has 2 levels of controllers (in addition to the bottom level of beneficiaries).

In this chapter I investigate a repeated task division and assignment problem. This was motivated by the multi-robot patrolling problem domain [4, 5, 36, 49, 166] introduced in Section 1.1. Specific applications may include robots patrolling a museum, or unmanned aired vehicles patrolling the borders. I assume there are multiple ways the robots can ensure there are no intruders, i.e. there are multiple ways the main task ("no intruders") can be decomposed into atomic tasks (patrolling routes for each robot). It is critical that no robot runs out of power (or there will be security breaches), so the goal is to alternate various task decompositions (patrolling route assignments) such that robots go through their batteries at roughly the same rate. More specifically, one should try to maximize the energy reserves of the robot lowest on power. In the general framework of this dissertation, each robot is a beneficiary, and each time it patrols one of the routes it receives a reward equal to the amount of recharged energy (if at all) minus the consumed energy.[1]

The goal in this domain is to "rotate" different task assignments (i.e. route assignments to robots) over and over such that no robot runs out of power.

Using a controller hierarchy makes sense in this problem: if the UAVs need to patrol several separate areas, one only needs to decide which UAV goes to which area; subsequent decisions regarding assigning UAVs to routes over each area can be made in parallel by separate controllers. By having separate controllers in charge of physically-localized

[1]Different robots may use different amounts of energy to patrol the same route: some UAVs may fly higher than others (and still take the same quality pictures); robots have wheels, or tracks, or legs, so different robots may be better than others on flat surfaces, or lawns, or stairs. With respect to recharging, one could assume the UAVs have solar power panels and some of the museum patrol routes include charging stations.

subproblems should provide better response to unforeseen local events (that can be dealt with locally). One should make sure, however, that by distributing the decision-making one does not sacrifice too much performance.

6.2 Hierarchical Task Division Problem

This problem is centered around a hierarchy of controllers acting on behalf of a fixed group of beneficiaries. The hierarchy has controllers on the interior nodes and beneficiaries at the leaves. The controller at the top gets a task that it decomposes and assigns the pieces (subtasks) to its direct subordinates. The subordinates further decompose their tasks, and so on; at the leaves the beneficiaries execute the tasks they were assigned. Beneficiaries receive rewards (costs) based on the tasks assigned to them. Just as in Chapters 3, 4 and 5, the rewards are static and deterministic. As a first step towards stochastic rewards, I assume the rewards are initially unknown to the controllers. The goal is to have the controllers discover and achieve $U^\star$ at the limit (and a small WL) with minimum communication/coordination.

6.2.1 Terminology

- A **request** to a controller is a task (from a predefined set) that it can receive from its direct superior (parent);
- An **action** is each of the ways a controller can decompose a request (task) and assign the subtasks as requests to its direct subordinates (children);
- An **outcome** is a reward profile (i.e. vector) observed by a controller for all beneficiaries in its subtree. When an outcome does not refer to a specific controller, it is an outcome observed by the controller at the top of the hierarchy (i.e. it is a **global outcome**, consisting of rewards for *all* beneficiaries).

Therefore, controllers receive requests, perform actions on those requests, and observe outcomes. The notation $r.a.o$ refers to an output o observed as a result of playing action a

when requests r was received. A controller observes the outcome after all its subordinates performed their actions. The outcomes that different controllers observe are different-sized pieces of the same global outcome. Controllers try to optimize the social welfare (leximin in this work) for *their* beneficiaries (they only "see" the part of a global outcome corresponding to their beneficiaries). So the controllers' goals might be misaligned due to them receiving different information.

6.2.2 Discussion

In order to make the problem more tractable, I assume that the hierarchy is roughly balanced and each task can only be decomposed in a small number of ways. More precisely, I assume the depth of the hierarchy be logarithmic in the number of leaves and that each task can only be decomposed in a constant number of ways, so that each controller can only receive a number of tasks linear in the number of beneficiaries. It follows that controllers near the bottom of the hierarchy get a linear number of requests, a small number of actions and a few outcomes for each request-action pair. Controllers near the top receive few requests, have a small number of actions, but an exponential number of outcomes per request-action pair. Therefore controllers at the bottom have small problems, but lack the "big picture," while the controller at the top has the big picture, which maps into an exponentially large problem.

An outcome observed by a controller depends on its action, but also on the actions of its superiors and its subordinates. The utility profile (average of outcomes) achieved by a controller for its beneficiaries depends on the relative frequency of possible requests the controller receives from its superior(s), and on how its subordinates handle their requests. In order for a controller to optimize utilities for his beneficiaries, a controller would have to influence the actions of both its superiors and subordinates.

Assuming controllers know all the possible outcomes for their beneficiaries, then controllers higher in the hierarchy "know better" than their subordinates. That is, it would be

socially counterproductive for a controller to influence its superiors.

In this work I do not allow controllers to make suggestions/comments on other controllers' actions. Influence goes only from top to bottom: subordinates need to adapt to the request frequencies their superiors impose on them.

The problem is further complicated by controllers having to discover all outcomes. A controller has to adapt to its superior's request frequencies, but the superior may change those frequencies on the fly when new outcomes are discovered. As the controller adapts to this "moving target," its request frequencies change, so its subordinates need to adapt as well. Controllers may be mislead by the information they get from their superiors because (1) their superiors are still adapting to their superiors, and (2) because it takes time to build an accurate estimate of the request frequencies.

I intend the results/insights for this problem to constitute stepping stones for a more general problem where the rewards are stochastic. A more general problem still is having the probability distributions over rewards be non-stationary. Although I leave these stochastic-reward problems as future work, some of the choices I make in this chapter (e.g. Section 6.5.2) are influenced by this long-term research goal. That is, I try not to take too much advantage of the structure of the simplified problem that would make the approach completely inapplicable to the general problem. Specifically, when the rewards are deterministic, controllers could spend some time in the beginning visiting all outcomes,[2] and then never explore again. Only one of my experiments takes advantage of this approach (Section 6.6.1).

6.2.3 Experimental Framework

In my experiments all controllers have at most two subordinates (controllers or beneficiaries) and at most two ways to divide each task. The number of beneficiaries is not restricted to exact powers of 2, and I impose the following balancing constraint at each node: the total

[2]While seeing all outcomes in the beginning (and their order) is irrelevant to $U^\star$, it is not so for WL.

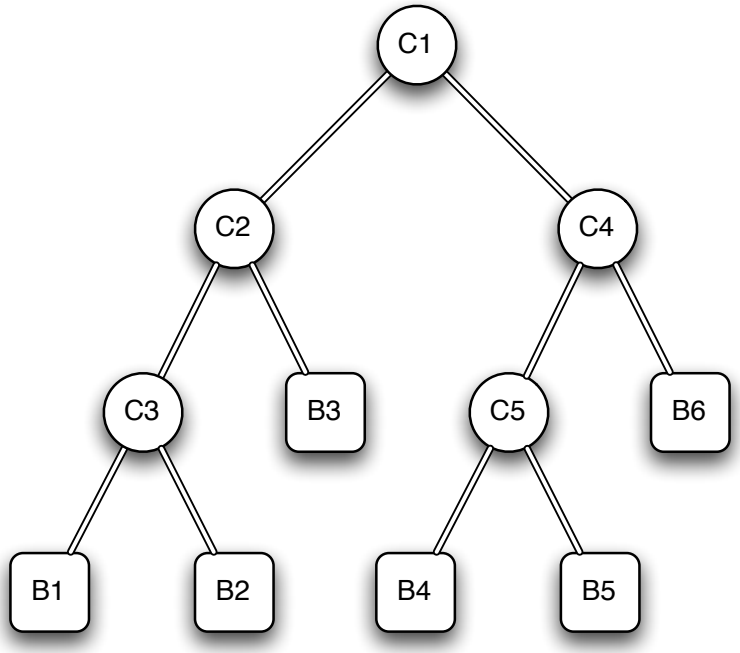


Figure 6.1: Hierarchy shape used in test problems involving six beneficiaries. Controllers are depicted with circles and beneficiaries are depicted using squares.

number of beneficiaries (leaves) in the left and right subtrees are as close as possible (see Figure 6.1 for the hierarchy with six beneficiaries). If there are n_b beneficiaries, then there are at most $n_b - 1$ controllers, and at most 2^{n_b-1} global outcomes.[3]

In order to keep the size of the problem input polynomial in n_b, I assume the following simplification: there are n_b atomic tasks and only 1-to-1 assignments of tasks to beneficiaries are considered (so the problem input consists of n_b^2 reward values). As for motivation, imagine the top controller's goal is to ensure the cleaning of a certain portion of beach, or a segment of highway. He divides the length of highway into two continuous pieces and assigns a piece to each of its subordinates, whose job is now to oversee the cleaning of their highway segments. Different beneficiaries (soldiers, volunteers) derive different rewards/costs from cleaning different atomic pieces (tasks): maybe they live closers to one segment then another, or they like the scenery more on some segments, etc.

[3]There are at most 2^{n_b-1} global outcomes because there are $n_b - 1$ controllers, and each has 2 actions (regardless of the request it receives).

To summarize, the hierarchy consists of n_b beneficiaries located in the leaves and at most $n_b - 1$ controllers located in the inner nodes. The top controller receives a unique request (tasks) repeatedly, and each controller has at most 2 actions for each possible request. It follows that a controller residing at depth d can receive at most 2^d different requests, and the height of the tree (max depth) is $\Theta(\log n_b)$. It follows that controllers receive $O(n_b)$ different requests.

Controllers associate outcomes with each request-action pair. The number of outcomes for a particular request-action pair is exponential in the number of subordinate controllers (in that controller's sub-tree). Although there are at most 2^{n_b-1} different outcomes, $U^\star$ can be achieved with only n_b outcomes (see discussion on Carathéodory's theorem in Section 4.3.1).

6.3 Algorithm

All controllers execute the same algorithm, which I break into three parts. The first part is called *ChooseAction*, and it is executed during the top-down phase: when a controller receives a request from its superior, and it needs to choose an action for that request. The action changes the requests for the controller's direct subordinates, which in turn execute *ChooseAction*, and so on. When all controllers have receive their requests and executed an action, all beneficiaries will have received atomic tasks. This is the beginning of the bottom-up phase, where rewards are reported back up the hierarchy. When a controller receives the outcome (consisting of rewards for all beneficiaries in its subtree), it executes *AddOutcome* to add the new outcome to its memory, then it executes *UpdateTarget* to update its local policy. The controller uses the updated policy to choose an action at the next step.

Each controller keeps track of how many times it received each request r (k_r) and how many times it used each action ($k_{r.a}$); additionally, it keeps a collection of outcomes seen (indexed by request and action, $r.a$) and uses these outcomes to compute $LU^\star$ and $LF^\star$

(which denote a controller's local $U^\star$ and local $F^\star$). Let actions(r) be the set of actions available for request r.

Algorithm 2. Controller-hierarchy Algorithm

procedure CHOOSEACTION(Request r, ExplorationSchedule e, TimeStep t)
  if $e[t] =$ explore **then** ▷ if this is an exploration step
    $a \leftarrow$ random(actions(r)) ▷ choose an action randomly
  else ▷ $e[t] =$ exploit, i.e. this is an exploitation step
    $a \leftarrow$ random$(\{a \in \text{actions}(r) | \frac{k_{r.a}}{LF^\star_{r.a}} = \min_{a' \in \text{actions}(r)} \frac{k_{r.a'}}{LF^\star_{r.a'}}\})$ ▷ use **GF^0**
  $k_{r.a} \leftarrow k_{r.a} + 1$
  return a

procedure ADDOUTCOME(Outcome r.a.o)
  for $r.a.o' \in$ outcomes$(r.a)$ **do**
    if $r.a.o'$ weakly Pareto-dominates $r.a.o$ **then**
      return
    else if $r.a.o$ Pareto-dominates $r.a.o'$ **then**
      outcomes$(r.a) \leftarrow$ outcomes$(r.a) \setminus \{r.a.o'\}$ ▷ remove $r.a.o'$ from outcomes$(r.a)$

  outcomes$(r.a) \leftarrow$ outcomes$(r.a) \cup \{r.a.o\}$ ▷ add outcome $r.a.o$ to outcomes$(r.a)$
  if $|$outcomes$(r.a)| >$ maxOutcomes **then**
    TRIMTOSIZE(outcomes$(r.a)$, maxOutcomes)

procedure UPDATETARGET ▷ Update local $U^\star$ and local $F^\star$
  $n_{lb} \leftarrow$ number of local beneficiaries
  $(LU^\star, LF^\star) \leftarrow$ Maximize SOCIALWELFARE$(U_1, \ldots, U_{n_{lb}})$

such that:

$\sum_i \sum_{r.a} \sum_{r.a.o} F_{r.a.o} R_{o,b} = U_b \ (\forall b = 1 \dots n_{lb})$, ▷ (C1)

$\sum_i \sum_{r.a} \sum_{r.a.o} F_{r.a.o} = 1$, ▷ (C2)

$\frac{1}{t} \sum_{r.a} k_{r.a} - w \leq \sum_{r.a} \sum_{r.a.o} F_{r.a.o} \leq \frac{1}{t} \sum_{r.a} k_{r.a} + w \ (\forall r \in \text{requests})$. ▷ (C3)

▷ To solve this mathematical program, use [119, 138] for leximin, [120] for OWA.

for $i \in$ requests **do**

  for $a \in \text{actions}(r)$ **do**

    $LF^{\star}_{r.a} = \sum_{r.a.o} LF^{\star}_{r.a.o}$

6.3.1 Choosing an Action

The algorithm takes as parameter an "exploration schedule," which can be an infinite sequence of boolean random variables, or, more generally an infinite sequence of elements from the set {explore, exploit}. The value "exploit" means the controller chooses the action according to its $LF^{\star}$, while "explore" makes the controller choose an action randomly.[4] This behavior mimics the concept of "ϵ-greedy exploration" from machine learning [157, 165]: a learner executes the action prescribed by its current policy (exploits it) with probability $1 - \epsilon$ and chooses a random action (explores) with probability ϵ; most often ϵ decreases with time.

The controllers can have different exploration schedules, or they can share one (in which case they all decide to explore at the same time). In the latter case one could assume the agents implement the same pseudo-random number generator algorithm and prime it using the same seed.

6.3.2 Updating Outcome Memory

After all controllers chose their actions, each controller sees the local "outcome," i.e. the reward profile for the beneficiaries in its subtree. Each controller adds its local outcome to

[4] An alternative exploration behavior can be to choose the least used action.

the collection of outcomes it observed for the current request-action pair (duplicates and Pareto-dominated outcomes are discarded).

This step has two parameters associated with it: the maximum number of outcomes allowed per request-action pair (maxOutcomes), and the heuristic used to trim the collection back to size when the maximum number is exceeded (TrimToSize).

6.3.3 Updating local $F^\star$

Each controller uses the updated collection of observed outcomes and the histogram of observed requests to update $LU^\star$ and $LF^\star$ (*UpdateTarget*). The constraints of type C1 and C2 define a utility profile as a linear combination of all different (Pareto-undominated) outcomes seen so far.

The purpose of the constraints of type C3 is to have controllers use the historical frequency of each of the requests (i.e. $\frac{1}{t}\sum_{r.a} k_{r.a}$) as the expected frequency in the future (i.e. local $\sum_{r.a}\sum_{r.a.o} F_{r.a.o}$). Rather than using an equality constraint (the frequency of a request in the future should be *equal* to the frequency of that request in the past), I use the parameter w to give controllers some "wiggle room," (the frequency of a request in the future should be *close* to the frequency of that request in the past). A controller treats this wiggle room optimistically: they set the expected frequencies of requests to whatever values (within w of the historical frequencies) optimize its local $U^\star$.

The following examples should provide some intuition on how wiggle room could be useful. First, the historical frequency of a request changes at each time step, whether the controller's superiors intend this or not. Imagine the controller receives a request every other step forever; the historical frequency is $\frac{1}{1}$ at step 1, then $\frac{1}{2}$ at step 2, then $\frac{2}{3}$, then $\frac{1}{2}$ again, then $\frac{3}{5}$. The historical frequency is an exact measure of the future frequency (i.e. $\frac{1}{2}$) only half the time. Secondly, the historical frequency might be off if a request was overused due to exploration. And if a request was overused, then some other request must have been underused. Using the wiggle room might allow a controller to "smooth over" the noise in

all these cases.

Intuitively, if the wiggle room is maximally loose ($w = 1$), then controllers see no correlation between past and future and they never adapt to their superiors' directions. Constant w values smaller than 1 still allow controllers to disobey their superiors. I address this issue by using wiggle room bounds that collapse onto each other as time goes by: $w = \frac{\delta}{t}$ (where δ is another parameter of the algorithm).

The effect of the wiggle room is two-fold. On one hand it allows controllers to ignore bad requests (due to exploration or their superiors not yet seeing all outcomes of their actions). If a request was received twice, it will take 200 steps for the controller with $w = \frac{1}{t}$ to finally ignore the request, but only 50 when $w = \frac{4}{t}$. On the other hand, requests that a controller would ignore based on local information might be beneficial to the controller's superiors. A larger wiggle room allows controllers to be stubbornly disobedient of their superiors, leading to worst losses. In a nutshell, larger wiggle rooms lets controllers ignore bad requests sooner, but also allows them to ignore good requests for longer periods of time.

6.4 Research Question

The goal, as presented in Section 6.2, is to have the controllers discover and achieve $U^\star$ at the limit (with a small WL, too) with minimum communication/coordination. I'm now in position to refine that goal: *have controllers (using only polynomial amount of memory and computation time per time step) achieve $U^\star$ at the limit when the only allowed communication is passing tasks top-down, and outcomes (rewards for all local beneficiaries) bottom-up*. Controllers do not inform their subordinates of their intended $LF^\star$, or that a request was received due to an exploration decision.

6.5 Experimental setup

I simulate the hierarchy of controllers for a number of time steps, which constitutes a run. Unlike other papers in the literature where controllers can only use information produced by other actions at the previous time step [174, 175, 190], here a time step consists of the controllers decomposing the original task all the way down to atomic tasks for beneficiaries, and having all controllers observe the rewards for the beneficiaries in their substrees.

6.5.1 Evaluating Success

I need a way to decide whether a run has finished successfully or not. I split the run into two parts: training (exploration is allowed) and testing (exploration is turned off).

The method I use requires the utility profile (vector of averaged rewards) during testing must be equal to $U^\star$.[5] For this method to work, one needs to find the period of the solution produced by the optimal centralized algorithm and have the testing period be a multiple of that value. For example, if $F^\star = [1/7, 6/7, 0, 0, \ldots, 0]$, the period is 7, and if the testing period is not a multiple of 7, this test will always indicate failure. This is particularly important since there can be multiple $F^\star$ (if there more than one, there are an infinity of them), so one cannot anticipate all periods.

[5] Based on controllers' policies ($LF^\star$) at moment t, I can compute the utility profile that would be achieved (at the limit) if controllers were to stop exploring and stop learning (i.e. stop calling *UpdateTarget*) after time t. One could assess the success of a run based on whether the at-the-limit utility profile matches $U^\star$ exactly.

This method looks very promising, since it gives at-the-limit statistics, while the previous method only gives statistics about the recent past. However, bad outcomes that were tried during exploration (and later on abandoned) still show up in controllers' historical information of request frequencies. This means the local policies could be slightly off, and since this method simulates turning off learning, it may compute at-the-limit utility profiles that are slightly different than $U^\star$ at all subsequent time steps.

One could address this issue by having the controllers remember only the last N requests, and N should be at least as large as the period induced by some $F^\star$. This parameter (N) affects controllers' decisions, so in practice they would have to choose it without knowing a $F^\star$ to begin with. I leave investigating this direction as future work.

6.5.2 Understanding Exploration

It is important for the controllers to explore enough that they see all outcomes of their actions (otherwise they might get stuck in suboptimal utility profiles). In my experiments all controllers do ϵ-exploration (random action with probability ϵ, follow greedy policy otherwise).

I consider the following setups with respect to the relation between controllers' exploration schedules:

1. **Independent exploration**: each controller has its own independent exploration schedule.[6] This seems somehow wasteful: imagine only the top controllers decides to explore but none of the others; if its policy indicates it should use both actions, this exploration step was essentially wasted.

2. **Systematic (supervised) exploration**: controllers are centrally coordinated during the first 2^{n_b-1} time steps so all outcomes are visited (in random order), then the distributed process resumes as usual. This is not meant as a MAS solution, just a base line for assessing the performance of the exploration methods.

6.5.3 Dealing with Exponentially Many Outcomes

At this point all controllers maintain a collection of outcomes observed for each of the request-action pairs. Since there might be an exponential number of such outcomes, I am investigating the effect of holding on to only a polynomial number of outcomes.

More specifically, each controller maintains for each request-action pair a number of outcomes not larger than the number of beneficiaries in that controller's subtree. When an outcome is observed, it is added to the memory and all Pareto-dominated outcomes are removed. If the memory is over capacity, one outcome is removed; I have implemented the

[6] So far the controllers' schedules are identical, they just use different seeds. I imagine there is some benefit in having controllers lower in the hierarchy do more exploration than top-level controllers.

following 6 simple outcome *culling* heuristics (in the algorithm's pseudocode I referred to this as *TrimToSize*):

Least Frequently Used (LFU) For each outcome in the memory I maintain a counter for how many times the outcome was observed since it was inserted in the memory (note that an outcome can be inserted into the memory, then removed, and then inserted again). The outcome with the smaller value in this usage counter gets removed.

Most Frequently Used (MFU) I remove the outcome with the highest value in the usage counter.

Least Recently Used (LRU) Drop from the memory the outcome least recently observed.

Most Recently Used (MRU) Drop from the memory the outcome most recently observed.

Least Recently First Used (LRFU) Drop the outcome that has been in the memory the longest.

Most Recently First Used (MRFU) Drop the outcome that has been in the memory the least.

For LFU and MFU I break ties randomly; ties are not possible for the rest (since outcomes are observed one at a time).

Note that when using LFU, new outcomes never make it in once the memory is full of undominated outcomes observed at least twice. Also, when using MRFU, new outcomes never make it in once the memory is full of undominated outcomes.

6.6 Experiments

I use six test problems: two have four beneficiaries (**Test-4×4-I**, **Test-4×4-II**), two have six beneficiaries (**Test-6×6-I**, **Test-6×6-II**), and two have eight beneficiaries (**Test-8×8-I**, **Test-8×8-II**). The inputs (rewards for each combination atomic-task and beneficiary) are

Test problem	Period	Testing phase	$WL_{\mathbf{GF}^0}$	$WL_{\mathbf{GF}^\theta}$
Test-4×4-I	83	4150	-21.61	-20.5
Test-4×4-II	13	650	-3.08	-3.07
Test-6×6-I	1	50	0	0
Test-6×6-II	14	700	-12.86	-9.49
Test-8×8-I	8	400	-3	-2.1
Test-8×8-II	49	2450	-15.86	-13.52

Table 6.1: Data for the six test problems: period length, training phase length (equal to 50 period lengths), and WL lower bounds guaranteed by $\mathbf{GF}^\mathbf{0}$ and $\mathbf{GF}^{\boldsymbol{\theta}}$ in a centralized setup with full information.

given in Appendix B. The lengths of the testing and training phases for each of these test problems are presented in Table 6.1. The maximum number of outcomes a controller is allowed to retain for each request-action pair is equal to the local number of beneficiaries in that controller's subtree.

A *treatment* is a combination of values for the parameters: exploration schedule, outcome culling heuristic and wiggle room. Each investigated treatment was run 100 times.

The following statistics are collected for each run:

1. **Does the utility profile experienced during the testing phase matches $U^\star$ exactly?** The limitations of this measure are discussed in Section 6.5.1.

2. **Were all outcomes visited?** This statistic is introduced to detect situations when controllers find the optimal solution by sheer luck after only exploring a part of the search space. Having a run end successfully after only exploring a small part of the search space weakens the result, since, intuitively, the run might have end in failure if a different part of the search space had been explored instead.

   Requiring a run to visit *all* outcomes might be too strong, though. Imagine a controller with no subordinate controllers, just two beneficiaries. There is exactly one outcome for each of the controller's action pairs, so two outcomes for each request. If outcome $r.a_1.o_1$ Pareto-dominates outcome $r.a_2.o_1$, then the controller will learn, after only receiving the request r twice that it should never use action $r.a_2$. Local outcome $r.a_2.o_1$

showed up in a single global outcome, and can only be part of a global outcome again during exploration steps. In this case, not visiting all global outcomes containing local outcome $r.a_2.o_1$ is the right thing to do. Furthermore, the larger the problem size, the more one would need to revisit Pareto-dominated local outcome $r.a_2.o_1$ for all outcomes to be visited.

This statistic could be informative when the success rate is small: if all (or most) of the runs visit all outcomes, then one should not bother increasing exploration; the converse is not necessarily true.

3. **WL**, the worst loss value throughout the run (i.e. during both training and testing phases). The purpose is to assess the impact exploration and learning have on WL.

   If a run converges to a utility profile other than $U^\star$, WL will get worse and worse in time; the longer the run, the worst the WL value. The length of the testing phase should not influence the value of WL. It follows that the distribution of WL values of the 100 runs only makes sense when all 100 runs end in success.

The primary statistic is the success rate (i.e. the fraction of runs achieving $U^\star$ during the testing phase); the other two are secondary in this exploratory work. As a result, I will compute confidence intervals for success rate (Section 6.6.4), but make no statistically significant statements about the last two.

6.6.1 Experiment 1: Supervised Exploration

In this experiment I ignore the exploration part, and study whether the controllers can achieve $U^\star$ once they *see* all outcomes. I used "supervised" exploration schedules (the controllers see all outcomes quickly and in random order), then they perform exploitation for the remaining of the training phase.

In this experiment I compare combinations of all the culling mechanisms (LFU, MFU, LRU, MRU, LRFU, MRFU) and wiggle values ($w = 0$, $w = \frac{1}{t}$, $w = \frac{2}{t}$ and $w = \frac{4}{t}$).

	LFU	MFU	LRU	MRU	LRFU	MRFU
$w = 0$	-109.60 ± 4.14	-109.92 ± 5.13	-110.23 ± 6.88	-110.19 ± 6.87	-110.94 ± 6.88	-110.96 ± 7.88
$w = 1/t$	-112.32 ± 5.36	-112.00 ± 3.97	-112.44 ± 5.37	-112.54 ± 5.92	-113.03 ± 6.44	-112.49 ± 5.32
$w = 2/t$	-113.67 ± 6.17	-113.79 ± 6.62	-112.91 ± 4.47	-113.65 ± 6.15	-112.16 ± 0.95	-113.43 ± 35.68
$w = 4/t$	-117.15 ± 4.73	-116.50 ± 3.04	-116.93 ± 4.25	-116.93 ± 4.25	-116.50 ± 3.04	-117.36 ± 5.15

Table 6.2: WL averages and standard deviations (based on 100 runs) for **Test-4×4-I** using supervised exploration and 1000 training steps.

	LFU	MFU	LRU	MRU	LRFU	MRFU
$w = 0$	-25.31 ± 1.20	-25.09 ± 0.98	-25.02 ± 0.91	-25.12 ± 1.00	-25.29 ± 1.09	-25.09 ± 0.99
$w = 1/t$	-25.20 ± 1.00	-25.17 ± 0.89	-25.14 ± 0.93	-25.07 ± 0.95	-25.31 ± 1.03	-25.19 ± 1.00
$w = 2/t$	-25.11 ± 1.04	-25.21 ± 1.08	-25.36 ± 1.18	-25.30 ± 1.11	-25.11 ± 1.00	-25.23 ± 0.94
$w = 4/t$	-25.26 ± 1.18	-25.07 ± 0.91	-25.16 ± 0.90	-25.05 ± 0.92	-25.13 ± 0.92	-25.30 ± 1.11

Table 6.3: WL averages and standard deviations (based on 100 runs) for **Test-4×4-II** using supervised exploration and 1000 training steps.

	LFU	MFU	LRU	MRU	LRFU	MRFU
$w = 0$	-764 ± 0	-764 ± 0	-764 ± 0	-764 ± 0	-764 ± 0	-764 ± 0
$w = 1/t$	-764 ± 0	-764 ± 0	-764 ± 0	-764 ± 0	-764 ± 0	-764 ± 0
$w = 2/t$	-764 ± 0	-764 ± 0	-764 ± 0	-764 ± 0	-764 ± 0	-764 ± 0
$w = 4/t$	-764 ± 0	-764 ± 0	-764 ± 0	-764 ± 0	-764 ± 0	-764 ± 0

Table 6.4: WL averages and standard deviations (based on 100 runs) for **Test-6×6-I** using supervised exploration and 1000 training steps.

	LFU	MFU	LRU	MRU	LRFU	MRFU
$w = 0$	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0
$w = 1/t$	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0
$w = 2/t$	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0
$w = 4/t$	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0

Table 6.5: WL averages and standard deviations (based on 100 runs) for **Test-6×6-II** using supervised exploration and 1000 training steps. Note that success rate is 100% for $w \in \{0, \frac{1}{t}, \frac{2}{t}\}$, and 0% for $w = \frac{4}{t}$.

	LFU	MFU	LRU	MRU	LRFU	MRFU
$w = 0$	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0	-838.28 ± 142.8	-824 ± 0
$w = 1/t$	-838.28 ± 142.8	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0
$w = 2/t$	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0
$w = 4/t$	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0	-824 ± 0

Table 6.6: WL averages and standard deviations (based on 100 runs) for **Test-6×6-II** using supervised exploration and 2000 training steps.

	LFU	MFU	LRU	MRU	LRFU	MRFU
$w = 0$	-208.7575 ± 0.075	-208.75 ± 0	-208.75 ± 0	-208.75 ± 0	-209.2225 ± 4.725	-208.75 ± 0
$w = 1/t$	-208.75 ± 0	-208.75 ± 0	-208.76 ± 0.1	-208.75 ± 0	-208.75 ± 0	-210.42 ± 16.775 [a]
$w = 2/t$	-208.75 ± 0	-208.75 ± 0	-208.755 ± 0.755	-208.7525 ± 0.025	-208.7525 ± 0.025	-208.75 ± 0
$w = 4/t$	-208.75 ± 0	-208.75 ± 0	-208.75 ± 0	-208.7525 ± 0.025	-208.75 ± 0	-208.7625 ± 0.054

[a] Only 99/100 success rate.

Table 6.7: WL averages and standard deviations (based on 100 runs) for **Test-8×8-I** using supervised exploration.

Training phase length I started with 1000 step training phase for the 4-by-4 tests and investigated appropriate training phase lengths for the larger problems. Table 6.4 shows 1000 training steps are enough for **Test-6×6-I**. Table 6.5 shows 1000 training steps are enough for **Test-6×6-II** for three of the four wiggle room settings (100% success rate), but not for $w = \frac{4}{t}$, where success rate is 0. Once I extended the training phase for **Test-6×6-II** to 2000 time steps, the success rate was 100% across the board (Table 6.6). Based on this, I extend the training phases for **Test-8×8-I** and **Test-8×8-II** to 2000 steps.

The results for the first five test problems (**Test-4×4-I**, **Test-4×4-II**, **Test-6×6-I**, **Test-6×6-II** with 2000 training steps and **Test-8×8-I**) are fairly similar:

- All runs ended in success (with the exception of a single run using MRFU and $w = \frac{1}{t}$).
- The culling heuristic has no clear effect on WL (Tables 6.2, 6.3, 6.4, 6.6 and 6.7).
- The choice of wiggle room had no obvious effect, except for **Test-4×4-I**, where it seems the larger the wiggle room range, the worse the WL value (see Table 6.2). Also, for **Test-6×6-II** $w = \frac{4}{t}$ needed more training steps than the other wiggle room settings (i.e. more than 1000).

The results for **Test-8×8-II** are different from those of the other test problems:

- Not all runs end in success.
- $w = \frac{4}{t}$ outperformed the other wiggle room settings, and achieved 100% success rate for all culling heuristics. Subsequent investigation revealed the reason for all culling heuristics performing identically for $w = \frac{4}{t}$ is that virtually no outcomes had to be culled.
- LFU, MFU, LRU and LRFU seem to do better than MRU and MRFU. Based on this observation, I picked one of the heuristics in the first category (namely LRU) to use in the rest of the experiments in this chapter.

	LFU	MFU	LRU	MRU	LRFU	MRFU
$w = 0$	97	95	98	67	94	68
$w = 1/t$	94	92	97	61	94	67
$w = 2/t$	93	95	93	69	97	63
$w = 4/t$	100	100	100	100	100	100

Table 6.8: Number of successful runs (out of 100 runs) for **Test-8×8-II** using supervised exploration.

To summarize, the controllers are able to find $U^\star$ when they are spoon-fed all the outcomes. This phenomena proved quite robust (with respect to combinations of wiggle room and culling mechanisms) for the first five tests. Based on the success rates on the sixth problem, I chose to focus on a single outcome culling heuristic (LRU) in the rest of the experiments.

6.6.2 Experiment 2: Independent Exploration

In the previous experiment I empirically established that a hierarchy of controllers can converge to an optimal policy if the controllers are presented with all outcomes up front. In this experiment I investigate whether the controllers can converge to the optimal policy with no central entity coordinating the exploration.

In this experiment I investigate the effects of controllers' independent exploration using LRU (one of the culling heuristics with good performance for all wiggle room settings), and the following ϵ-greedy exploration schedule family: $1/(1 + \frac{t}{C_\epsilon})$ [157]. I start with $C_\epsilon = 100$, and investigate whether higher values are necessary (i.e. whether more exploration is needed).

Test-4×4-I and Test-4×4-II The results in Table 6.9 show $C_\epsilon = 100$ is high enough: controllers always see all outcomes and the success rate is around (or over) 90% success rate for the four wiggle room settings. On a side note, $w = \frac{4}{t}$ performed the worst of the four wiggle room settings, and $w = \frac{2}{t}$ seems to be the best choice in this setup.

	Test-4×4-I			Test-4×4-II		
$w = 0$	100	91	-3862 ± 218	100	94	-653 ± 63
$w = 1/t$	100	96	-3847 ± 206	100	94	-664 ± 54
$w = 2/t$	100	97	-3829 ± 214	100	96	-662 ± 60
$w = 4/t$	100	89	-3801 ± 197	100	93	-652 ± 59

Table 6.9: Statistics for **Test-4×4-I**, **Test-4×4-II** using independent exploration with $C_\epsilon = 100$, 1000-step training, and LRU outcome culling. The first number in each cell is the number of runs (out of 100) where all outcomes were seen, the second is the number of successful runs, and the third number of the average WL $\pm$ the standard deviation.

	$C_\epsilon = 100$			$C_\epsilon = 1000$		
$w = 0$	7	100	-7139 ± 341	100	100	-18381 ± 378
$w = 1/t$	6	100	-7195 ± 272	99	100	-18396 ± 458
$w = 2/t$	9	100	-7210 ± 363	100	100	-18376 ± 399
$w = 4/t$	7	100	-7290 ± 361	100	100	-18369 ± 440

Table 6.10: Statistics for **Test-6×6-I** using independent exploration, 1000 step training and LRU culling. The first number in each cell is the number of runs (out of 100) where all outcomes were seen, the second is the number of successful runs, and the third number is the average WL $\pm$ the standard deviation.

Test-6×6-I For $C_\epsilon = 100$, less than 10% of the runs see all outcomes and success rate is 100%. In this test the optimal solution consists of a single assignment of tasks to beneficiaries, making this test somewhat easier than others of the same size. Because it is not clear whether $C_\epsilon = 100$ offers enough exploration for a 6-by-6 problem, I also try $C_\epsilon = 1000$. In this new setup (last column in Table 6.10) success rate is still 100%, and controllers see all outcomes every time. It seems that the increase in the number of outcomes going from 4-by-4 to 6-to-6 requires more exploration, indeed. Also note the significant increase in WL that comes with the extra exploration.

Test-6×6-II The results for $C_\epsilon = 100$ are in the first column of Table 6.11: only 10–16% of the runs had observed all runs and the success rate is, at best, at 60–70%. More exploration might be needed. $C_\epsilon = 1000$ (see second column of Table 6.11) achieves maximal exploration, but the success rate is compromised. I will address this in the next experiment.

	$C_\epsilon = 100$			$C_\epsilon = 1000$		
$w = 0$	16	66	-7353 ± 373	100	0	-19413 ± 529
$w = 1/t$	13	60	-7372 ± 413	100	0	-19561 ± 565
$w = 2/t$	19	48	-7360 ± 370	100	0	-19565 ± 553
$w = 4/t$	16	28	-7400 ± 428	100	0	-19534 ± 594

Table 6.11: Statistics for **Test-6×6-II** using independent exploration, 1000 step training and LRU culling. The first number in each cell is the number of runs (out of 100) where all outcomes were seen, the second is the number of successful runs, and the third number is the average WL $\pm$ the standard deviation.

	$C_\epsilon = 100$ training =1000			$C_\epsilon = 1000$ training =1000			$C_\epsilon = 2000$ training =2000		
$w = 0$	0	68	-524 ± 34	2	0	-1184 ± 35	69	0	-2355 ± 43
$w = 1/t$	0	81	-516 ± 33	3	0	-1175 ± 35	65	0	-2355 ± 51
$w = 2/t$	0	79	-531 ± 35	2	0	-1180 ± 37	67	0	-2360 ± 45
$w = 4/t$	0	68	-522 ± 32	2	0	-1183 ± 36	72	0	-2361 ± 50

Table 6.12: Statistics for **Test-8×8-I** using independent exploration and LRU culling. The first number in each cell is the number of runs (out of 100) where all outcomes were seen, the second is the number of successful runs, and the third number is the average WL $\pm$ the standard deviation.

Test-8×8-I When using $C_\epsilon = 100$, the success rate is approximately 70–80%, and no run ever visits all outcomes (Table 6.12, first column).

I tried $C_\epsilon = 1000$, since it managed to cover exploration completely for **Test-6×6-II**. In this case $C_\epsilon = 1000$ showed only minor improvements in exploration; again, the success rate is compromised (Table 6.12, second column). Using $C_\epsilon = 2000$ (with a 2000-step training phase) increased drastically the percentage of runs to visit all outcomes (65–70%, see Table 6.12 second column), but it also doubled up WL. I will fix the success rate issue in the next experiment, keeping $C_\epsilon = 1000$.

Test-8×8-II I performed for **Test-8×8-II** the same experiments as for **Test-8×8-I**, and I got fairly similar results (see Table 6.13).

I consider that the first three test problems were solved to a satisfactory degree and focus on the last three. I conjecture that there was a lot of exploration, but not enough time for

training	$C_\epsilon = 100$ training =1000			$C_\epsilon = 1000$ training =1000			$C_\epsilon = 2000$ training =2000		
$w = 0$	0	83	-3209 ± 203	2	0	-9266 ± 2203	76	0	-18587 ± 2515
$w = 1/t$	0	85	-3196 ± 251	0	0	-9582 ± 2816	66	0	-18390 ± 2291
$w = 2/t$	0	91	-3243 ± 220	5	0	-10035 ± 3456	63	0	-18261 ± 2073
$w = 4/t$	0	90	-3271 ± 245	0	0	-8796 ± 293	69	0	-17585 ± 395

Table 6.13: Statistics for **Test-8×8-II** for independent exploration and LRU culling. The first number in each cell is the number of runs (out of 100) where all outcomes were seen, the second is the number of successful runs, and the third number is the average WL $\pm$ the standard deviation.

	2000			4000		
$w = 0$	19	91	-3568 ± 227	23	93	-5636 ± 358
$w = 1/t$	20	92	-3586 ± 231	32	94	-5686 ± 317
$w = 2/t$	22	89	-3571 ± 246	26	95	-5617 ± 312
$w = 4/t$	15	93	-3695 ± 231	35	93	-5666 ± 371

Table 6.14: Statistics for **Test-6×6-II** using LRU culling, independent exploration with $C_\epsilon = 1000$ restricted to the first 10% of the training phase. The two columns contains results for training phase lengths of 2000 steps and 4000 steps training, respectively. The first number in each cell is the number of runs (out of 100) where all outcomes were seen, the second is the number of successful runs, and the third number is the average WL $\pm$ the standard deviation.

controllers to filter out the noise in the frequencies of requests from their superiors. I test this hypothesis in the next experiment.

6.6.3 Experiment 3: Increasing Exploitation

In this experiment, I restrict exploration moves to the first 10% of the training phase. Also, I keep $C_\epsilon = 1000$ (using $C_\epsilon = 2000$ in the last experiment was most likely overkill).

Test-6×6-II My first set of runs use 2000 steps of training, with only first 200 steps open to exploration. The results (Table 6.14) are quite good. Doubling up exploration (4000-step training phase, with exploration restricted to the first 400 steps) produces small improvements in success rate. Diminishing returns are to be expected as one approaches 100%.

	2000			4000		
$w = 0$	0	94	-291 ± 15	0	94	-364 ± 24
$w = 1/t$	0	100	-225 ± 14	0	100	-365 ± 24
$w = 2/t$	0	100	-223 ± 16	0	100	-364 ± 24
$w = 4/t$	0	100	-227 ± 13	0	100	-368 ± 24

Table 6.15: Statistics for **Test-8×8-I** using LRU culling, independent exploration with $C_\epsilon = 1000$ restricted to the first 10% of the training phase. The two columns contains results for training phase lengths of 2000 steps and 4000 steps training, respectively. The first number in each cell is the number of runs (out of 100) where all outcomes were seen, the second is the number of successful runs, and the third number is the average WL $\pm$ the standard deviation.

	2000		
$w = 0$	0	100	-1747 ± 571
$w = 1/t$	0	100	-1892 ± 880
$w = 2/t$	0	100	-1724 ± 587
$w = 4/t$	0	100	-1672 ± 137

Table 6.16: Statistics for **Test-8×8-II** using LRU culling, 2000-step training, independent exploration with $C_\epsilon = 1000$ restricted to the first 10% of the training phase. The first number in each cell is the number of runs (out of 100) where all outcomes were seen, the second is the number of successful runs, and the third number is the average WL $\pm$ the standard deviation.

Test-8×8-I In Table 6.15 I present the results for the 2000-step training phase. Using a 4000-step training phase shows no improvement in an already high success rate (95% for $w = 0$, the other were already at 100%). I note that the success rate is less than 100% when $w = 0$, and increasing the training phase does not help; this is evidence that having wiggle room embedded into the algorithm is beneficial.

Test-8×8-II In Table 6.16 I present the results for the 2000-step training phase. The success rate is 100% for all wiggle room settings, so there is no need to try the 4000-step training phase.

This experiment confirmed the hypothesis at the end of Section 6.6.2: it wasn't the lack of exploration, but the lack of exploitation that was responsible for the 0% success rate in Table 6.12 and Table 6.13. In this section controllers achieved high success rates by restricting the exploration to the first 10% of the training phase and allowing a long period

Problem	% of successful runs	Reference	Confidence Interval
Test-4×4-I	97%	Table 6.9	[88.2151%, 99.7805%]
Test-4×4-II	96%	Table 6.9	[86.6147%, 99.5228%]
Test-6×6-I	100%	Table 6.10	[93.8628%, 100%]
Test-6×6-II	95%	Table 6.14	[85.0843%, 99.1913%]
Test-8×8-I	100%	Table 6.15	[93.8628%, 100%]
Test-8×8-II	100%	Table 6.16	[93.8628%, 100%]

Table 6.17: Confidence intervals for the success rate on each of the six test problems. There is a 99% confidence level for the entire table

of exploitation before the testing phase.

6.6.4 Confidence Intervals for Success Rate

Table 6.6.4 contains confidence intervals for one success rate value for each of the six test problems. They use the same settings for outcome culling (i.e. LRU) and wiggle-room (i.e. $w = \frac{2}{t}$), but different independent-exploration variations. This is because the first three problems were considered solved after Experiment 2 (Section 6.6.2) and were not used in Experiment 3 (Section 6.6.3). I compute the confidence intervals using the method proposed in [114]. There is a 99% confidence level for the entire table (i.e. $\alpha = 0.01$), or (using the Bonferroni correction) a $\frac{0.01}{6}$ significance level for each of the six confidence intervals.

6.7 Discussion

The experiments in this chapter constitute proof of concept that a hierarchy of controllers running my algorithm can converge to $U^\star$ in this particular static, deterministic, task allocation problem with limited communication, polynomial space and polynomial complexity per time step.

These experiments are also of an exploratory nature (they were used to figure out appropriate exploration-exploitation settings for the Hierarchical Task Division Problem), which partly explains why they do not consist of larger problems. Another reason is that

for large problems these experiments would take too long to simulate on a single machine. That is because controllers with the same depth are meant to run in parallel, but they are executed sequentially when the controller hierarchy is simulated on a single machine. For this reason, an appealing future work direction would be to investigate the effect of reducing the number of linear programs solved per time step by having the controllers switch from recomputing their local $F^\star$ ($LF^\star$) profiles every time step to recomputing their $LF^\star$ profiles every T steps (unless new outcomes are discovered).

I leave as future work validating this approach presented in this chapter on problems featuring more beneficiaries, unbalanced controller hierarchies, and different branching factors. I also leave as future work the tuning of parameters (i.e. finding good exploration schedules, or culling heuristics), and applying this to stochastic-reward problems. This is actually the reason I went with an ϵ-greedy exploration schedule (rather than having the controllers explore every step for the first T steps, then perform greedy exploitation for the rest of the training phase): the ϵ-greedy exploration schedule is relevant when generalizing the problem to stochastic-rewards.

Chapter 7: Future Work

This chapter consists of a discussion about extending the applicability of the algorithms proposed in this dissertation to real-world problems (Section 7.1), and a list of open problems relevant to long-term fairness (Section 7.2).

7.1 Other Complicating Factors

Throughout this dissertation I used my $\mathbf{GF}^{\theta}$ algorithms to solve BASE PROBLEM extended with three complicating factors: finite time-horizons, stateful domains and controller-hierarchies. In this section I discuss possible directions for other BASE PROBLEM extensions: stochastic rewards and dynamic beneficiary sets. I introduce two new variants of $\mathbf{GF}^{\theta}$, but provide no proofs of correctness or performance guarantees.

The problems I solved in Chapters 4 and 5 feature a fixed set of beneficiaries; and static, deterministic rewards, that are known upfront. I proposed the following approach to solve these problems:

Algorithm 3. Single Controller; Fixed Beneficiary Set; Fixed, Known Rewards.

1: Compute a long-term action frequency profile $F^{\star}$; ▷ Section 3.5, 4.3.4, or 5.4.1.
2: Select one of the $\mathbf{GF}^{\theta}$ algorithms; ▷ i.e. choose θ: Section 4.2 or 5.3.
3: **while** true **do**
4: Use $\mathbf{GF}^{\theta}$ to pick actions repeatedly. ▷ Section 4.2

Note that only the k values change (the number of times each action was used), while $F^{\star}$ and θ are only computed once in the beginning.

7.1.1 Stochastic Rewards

When the rewards are not known before hand, my approach (Chapter 6) has the controllers (1) use ϵ-exploration and (2) recompute the local versions of $F^\star$ (but *reuse* the k values). The same principle could be applied to stationary stochastic rewards:

Algorithm 4. Single Controller; Fixed Beneficiary Set; Stochastic Rewards.

1: **while** true **do**
2: **if** exploration step **then**
3: $a \leftarrow \text{random}(1, \ldots, n_a)$
4: **else**
5: $a \leftarrow \mathbf{GF}^{\boldsymbol{\theta}}$;
6: Execute action a, observe reward profile $[R_1(a), \ldots, R_{n_b}(a)]$.
7: Update profile of average rewards $[\overline{R}_1(a), \ldots, \overline{R}_{n_b}(a)]$ for action a.
8: Recompute $F^\star$ for $[\overline{R}_1(\cdot), \ldots, \overline{R}_{n_b}(\cdot)]$

A few observations should make this algorithm sketch more intuitive.

- The algorithm updates the average reward $\overline{R}_b(a)$ for each action a and beneficiary b; and at each iteration it solves a deterministic-reward problem using $\overline{R}_b(\cdot)$ as reward functions for each beneficiary. One can reduce the amount of per-step computation by executing line 8 every τ time steps (i.e. iterations).

- Although the $F^\star$ is recomputed, the k values are not reset. There are two arguments for this decisions:

  1. Regardless of the expectations the controller had before each time action a was used, the actual distribution of rewards was the same, so the k values should be preserved;

  2. Beneficiaries' utilities should converge at the limit to $\overline{U^\star}$, the solution of the deterministic reward problem using $\mathbf{E}[R_b(\cdot)]$ as reward functions for each beneficiary.

This means that the k values divided by time converge to $\overline{F^\star}$, whether they are reset or not.

7.1.2 Dynamic Beneficiary Set

Imagine extending the professor-assignment domain (Example 3.1) to allow professors to get hired or to retire. If the beneficiary set shrinks, some previous action may become infeasible (the classes taught by the professor that retired are not covered), and new actions (class assignments) need to be created. If the beneficiary set grows, one still needs new actions (or the new hire would always be idle). The new set of actions may include action a where the new hire is idle (to allow him to go on sabbatical), and this action may be identical to a previous action (with respect to who teaches what). However, I argue that the k_a value should be reset, otherwise the system will consider that the new hire (on his first day on the job) already took k_a semesters of sabbatical.

Algorithm 5. Single Controller; Dynamic Beneficiary Set; Deterministic Rewards.

1: **while** true **do**
2: **while** no beneficiary change **do**
3: Use $\mathbf{GF}^{\boldsymbol{\theta}}$ to pick an action;
4: Update current Windfall values;
5: Compute new action set;
6: Reset k values for new action set;
7: Compute $F^\star$ and $U^\star$ for new action set;
8: Recompute θ values based on previous Windfall values.

It is intuitively very important that beneficiaries do not start with a clean slate every time there is a change in the beneficiary set. Intuitively, the windfall values should not be lost: if a beneficiary fell behind (negative windfall), the system should not clear that just because a new professor was hired. Although the behavior of $\mathbf{GF}^{\boldsymbol{\theta}}$ is not directly affected

by the Windfall values, I propose to affect it indirectly though the θ values. Line 8 uses the following equation generalized from Equation 4.15:

$$\mathrm{WL}_b = -\sum_{a=1}^{n_a^\star} \theta_a X_{a,b} + \sum_{a=1}^{n_a^\star} X_{a,b}^- + \mathrm{Windfall}_b(D'', t) \tag{7.1}$$

Same modification (adding current windfall values to beneficiaries' WL bounds) applies verbatim to Equations 4.14, and 4.16, 4.17.

The $\mathbf{GF}^{\boldsymbol{\theta}}$ algorithms are robust in the sense of [55]: one can disregard their decisions and (as long as the k values are updated properly) the algorithm will continue to function and eventually get back "on schedule." Alternatively, one can use the $\mathbf{GF}^{\boldsymbol{\theta}}$ version sketched in this section to re-tune θ based on the Windfall values after the algorithm was disobeyed.

7.2 Open Problems

This section contains two open problems related to the finite time horizon problem in Chapter 4.

Finding the optimal *expected* utility profile At page 50, footnote 1, I discuss the issue of leximin-optimizing the expected utility profile at the end of the game for arbitrary game-length probability distributions. To the best of my knowledge this is still an open problem, even when the deadline is known (i.e. the distribution has support of size one).

Time-discounted model An alternative to the average-reward utility model I used in this work is the time-discounted utility model, which is very popular in economics, game theory [87, 124], reinforcement learning [165, 171], etc.:

$$U_b(S) = \frac{1}{1-\gamma_b} \sum_{t=1} R_b(S_t) \times \gamma^{t-1}.$$

The following procedure finds $U^\star$ under this model, but only when the beneficiaries are patient enough, and all γ_b are equal. Specifically, if $\forall b \in \mathbb{B}$: $\gamma_b = \gamma \geq \frac{n_b - 1}{n_b}$, then the space of feasible utility profiles coincides with $\mathbb{H}$, and computing $U^\star$ is identical to solving the BASE PROBLEM (Section 3.2); furthermore, the sequence of actions that achieves $U^\star$ at the limit can be produced with the algorithm in [161]. This approach has limited applicability, however, because the lower bound for γ converges to 1 as the number of beneficiaries increases.

When $\gamma < \frac{n_b - 1}{n_b}$, the space of feasible utility profiles is not compact anymore; $U^\star$ might not even be on the convex hull boundary. Furthermore, if γ_b values are *not* all equal, then the space of feasible utility profiles is not contained inside $\mathbb{H}$.[1] Finding $U^\star$ in these cases is still an open problem to the best of my knowledge.

[1] This phenomenon is related to what [148] calls *serendipity of disagreement*. Imagine two people have to split a pizza (which they both value at 1) and the first person likes the crust more than the toppings while the second likes the toppings more than the crust. Then a division that gives the first person more than half the crust and less than half the amount of topping has a sum of utilities strictly greater than 1.

Going back to the problem at hand, imagine two beneficiaries having to share a homogeneous cake and that any piece of cake yields the same reward (equal to the size of the piece relative to the entire cake) to both beneficiaries. Furthermore, the beneficiaries repeatedly divide identical cakes, and the first beneficiary is more patient than the second. In this case it is possible to achieve a utility profile with the sum strictly greater than 1, by giving the entire cake to the less the patient beneficiary the first few times and to the more patient beneficiary for the rest of time.

Chapter 8: Conclusion

Fairness is important to many multi-agent systems, and it is precisely in the context of fairness in multi-agent systems that this dissertation operates. Specifically, this dissertation focuses on the theory and practice of improving fairness-efficiency tradeoffs when agents interact multiple times (as opposed to a single-shot interaction). This venue has received insufficient attention in the literature so far, and this dissertation makes significant contributions in this direction. I elaborate on these contributions in the remainder of this chapter.

This work was motivated by the urban traffic problem domain. However, the large number of beneficiaries; the inherent distributed nature of the domain; and high degree of uncertainty, stochasticity, and dynamism make the domain too complex for a principled approach. As a result, I started with a simple theoretical model, and studied three of these complicating features: stochastic or unknown time horizon, stateful domain and distributed decision making.

This work focuses on the computational aspect of achieving long-term fairness (rather than on the philosophical aspect of what is fair in a specific application), under the average-reward utility model. Specifically, I introduced the $\mathbf{GF}^{\boldsymbol{\theta}}$ family of algorithms, which are able to converge to any given feasible outcome, while limiting the worst case losses. I have shown that even $\mathbf{GF}^{\mathbf{01}}$ (a "lighter-weight" version of $\mathbf{GF}^{\boldsymbol{\theta}}$ which does not require linear programming to tune the θ values) behaves optimally in the best case, and outperforms relevant algorithms from the literature in the worst case. Additionally, I provided empirical results for the behavior of $\mathbf{GF}^{\boldsymbol{\theta}}$ and $\mathbf{GF}^{\mathbf{01}}$ on randomly generated problem instances.

One can decide beforehand on the social welfare versus worst-case loss tradeoff that

makes sense in one's problem domain (one should use the multi-objective algorithm I introduced in Chapter 5 to generate a Pareto-front of such tradeoff points). Although this work has been presented in the context of a specific fairness measure (i.e. leximin), the proposed approach works for other social welfare measures as well. One can trivially modify the multi-objective algorithm in Chapter 5 to select long-term outcomes with properties such as an arbitrary fairness-efficiency tradeoff (see Section 4.3.4), or the stability issue discussed in the context of cooperative game theory in Section 1.1.1. When one decides on the $F^\star$ profile that makes sense for one's application, one can use the $\mathbf{GF}^{\boldsymbol{\theta}}$ algorithm out of the box to generate an infinite sequence of actions guaranteed to converge to the desired long-term outcome.

8.1 Contributions

This section summarizes the contributions of this dissertation.

Theoretical results

- I proposed the worst-case loss solution concept for problems with an unknown or stochastic finite time-horizon (I also discussed two alternatives, which are listed in the open problem list in Section 7.2).
- I proved hardness results related to this solution concept (with emphasis on the problem of optimizing the worst-case loss).

Algorithmic contributions

- I introduced $\mathbf{GF}^{\boldsymbol{\theta}}$, a parameterized family of approximation algorithms for generating infinite sequences of actions with guaranteed bounds on worst loss; additionally, I provided:

  - preprocessing algorithms for tuning the parameterization to optimize the worst-case loss bound.
  - a preprocessing algorithm optimized for a special class of resource (task) allocation problems.

- I extended $\mathbf{GF}^{\boldsymbol{\theta}}$ and the parameterization tuning algorithm to stateful domains; additionally, I proposed (and empirically tested) a multi-objective genetic algorithm approach to finding social-welfare vs. worst-case loss tradeoff points.

- I proposed an algorithm built around a version of $\mathbf{GF}^{\boldsymbol{\theta}}$ for the controller-hierarchy domain (this algorithm was also only evaluated empirically).

Social welfare measures All the algorithms are presented in the context of leximin, which is a widely used social welfare measure with solid theoretical properties; moreover, there is an LP formulation that would compute the input for $\mathbf{GF}^{\boldsymbol{\theta}}$ (i.e. $F^\star$, $U^\star$). However, one can use the $\mathbf{GF}^{\boldsymbol{\theta}}$ algorithms with almost any other social welfare measure, as long as one can compute its corresponding $F^\star$ profiles. For OWA there is also an LP formulation; for others, one could solve differential equations, or use hill climbing, simulated annealing, or evolutionary computation.

Handling additional complicating factors The work in this dissertation studied three real-world-inspired extensions to a simple theoretical model. The algorithms used to solve these extensions are generalizable to tackle other extensions (Section 7.1), such as stochastic rewards and non-stationary beneficiary sets.

Combining complicating factors The initial plan was to investigate each of the three extensions in isolation, and then learn to combine the resulting approaches. The algorithms used on the last two extensions are based on the $\mathbf{GF}^{\boldsymbol{\theta}}$ family of algorithms, which was

proposed for the first extension; therefore the stateful domain and the distributed decision-making were already studied in combination with finite time horizon. The same is true for the other extensions listed in the future work chapter.

Appendix A: Additional Fine Time Horizon Results

This appendix contains a number of results needed in Chapter 4. Specifically, Section A.1 contains NP-hardness proofs for building WL-optimal finite-length sequences (Theorem A.1) and finding $F^\star$ profiles with optimal WL bounds (Theorem A.4). Additionally, Theorem A.2 proves the convergence to $U^\star$ of any algorithm able to lower-bound all windfall values, and Theorem A.3 establishes $\mathbf{GF}^{\boldsymbol{\theta}\prime}$s convergence to a desired $F^\star$ profile.

Section A.2 contains mathematical derivations needed (in combination with Theorem 4.1) to compute WL bounds in Equations 4.14–4.17. Finally, Section A.3 discusses a tight example for the approximation ratio established in Equation 4.21.

A.1 Additional Theorems

Theorem A.1. Problem 2 is NP-hard.

Proof. I prove Problem 2 is NP-hard through a reduction from PARTITION PROBLEM [66]. In the PARTITION PROBLEM one is given a multiset V of positive integer numbers and has to decide if V can be partitioned into two subsets whose elements sum to the same value. I will show that the answer to this decision problem is "YES" if and only if the optimal worst loss for a corresponding WORST LOSS OPTIMIZATION PROBLEM instance is equal to a specific value.

For an arbitrary PARTITION PROBLEM instance $V = \{v_1, \ldots, v_{|V|}\}$, the associated WORST LOSS OPTIMIZATION PROBLEM instance has $|V| + 1$ actions and $|V| + 3$ beneficiaries. I differentiate between two types of beneficiaries. I refer to the first $|V| + 1$ beneficiaries as $b_1 \ldots b_{|V|+1}$, and to the last two as b' and b''. The rewards are presented in the following table:

	b_1	b_2	$\dots$	$b_{\lvert V\rvert}$	$b_{\lvert V\rvert+1}$	b'	b''
a_1	$\delta - \frac{\sigma}{2}$	$\delta + \frac{\sigma}{2\lvert V\rvert}$	$\dots$	$\delta + \frac{\sigma}{2\lvert V\rvert}$	$\delta + \frac{\sigma}{2\lvert V\rvert}$	$\delta + v_1$	$\delta - v_1$
a_2	$\delta + \frac{\sigma}{2\lvert V\rvert}$	$\delta - \frac{\sigma}{2}$	$\dots$	$\delta + \frac{\sigma}{2\lvert V\rvert}$	$\delta + \frac{\sigma}{2\lvert V\rvert}$	$\delta + v_2$	$\delta - v_2$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$a_{\lvert V\rvert}$	$\delta + \frac{\sigma}{2\lvert V\rvert}$	$\delta + \frac{\sigma}{2\lvert V\rvert}$	$\dots$	$\delta - \frac{\sigma}{2}$	$\delta + \frac{\sigma}{2\lvert V\rvert}$	$\delta + v_{\lvert V\rvert}$	$\delta - v_{\lvert V\rvert}$
$a_{\lvert V\rvert+1}$	$\delta + \frac{\sigma}{2\lvert V\rvert}$	$\delta + \frac{\sigma}{2\lvert V\rvert}$	$\dots$	$\delta + \frac{\sigma}{2\lvert V\rvert}$	$\delta - \frac{\sigma}{2}$	$\delta - \sigma$	$\delta + \sigma$

where σ is the sum of elements in V (i.e. $\sigma = \sum_{i=1}^{|V|} v_i$), and δ is an arbitrary constant. The computational effort for the transformation is obviously linear in the size of the input.

I will show that I can solve any instance of the PARTITION PROBLEM by solving the corresponding WORST LOSS OPTIMIZATION PROBLEM instance and analyzing the windfalls for the first $|V| + 1$ steps of the optimal-WL sequence. Specifically, I map the sequence of actions into a partition $(V', V - V')$ of V such that all actions used before action $a_{|V|+1}$ correspond to elements in V', and the actions used after action $a_{|V|+1}$ correspond to elements in $V - V'$. I use the first $|V| + 1$ beneficiaries to ensure that each action is used exactly once during the first $|V| + 1$ steps (proof follows shortly). Additionally, I use beneficiaries b' and b'' to associate the sums of elements in V' and $V - V'$ with windfalls.

The first step is to prove that $U^\star = [\delta \dots \delta]$. Note that each action dispenses the same sum of rewards: $(|V| + 3) \times \delta$. It follows that the sum of all beneficiaries' utilities is $(|V| + 3) \times \delta$ at any time, regardless of the sequence of actions. Therefore no vector U can be leximin superior to $[\delta \dots \delta]$ since U can't have a component strictly greater than δ without having some other component strictly smaller than δ. Note that the utility vector $[\delta \dots \delta]$ is achievable: $U(S_{1:|V|+1}) = [\delta \dots \delta]$ for any sequence S starting with a permutation of the $|V| + 1$ actions. Since the vector $[\delta \dots \delta]$ is achievable and no utility profile can be leximin-superior, then $U^\star$ must be equal to $[\delta \dots \delta]$.

For any sequence S it holds that $\text{Windfall}_{b_j}(S, 1) = -\frac{\sigma}{2}$ if $S_1 = a_j$ (whichever action a_j is

used first, it yields a windfall of $-\frac{\sigma}{2}$ to beneficiary b_j), so the worst loss value WL $\leq -\frac{\sigma}{2}$.

Claim *The answer to the* PARTITION PROBLEM *instance is "YES" (i.e. there exists* $V' \subset V$ *such that* $\sum_{v \in V'} v = \sum_{v \in V - V'} v = \frac{\sigma}{2}$*) if and only if* WL $= -\frac{\sigma}{2}$.

First I will prove the IF part of the claim: if the WORST LOSS OPTIMIZATION PROBLEM solver produces a sequence S such that $\forall b \in \mathbb{B}, \forall t \in \mathbb{N}$: Windfall$_b(S,t) \geq -\frac{\sigma}{2}$, then V can be partitioned into two subsets of equal size.

I submit that each of the $|V|+1$ actions is used exactly once in the first $|V|+1$ positions of S. If action a_j were to show up k times ($k \geq 2$), then Windfall$_{b_j}(S, |V|+1) = -\frac{\sigma}{2} \times k + (|V|+1-k) \times \frac{\sigma}{2 \times |V|} = -\frac{\sigma}{2}\left(k - \frac{|V|+1-k}{|V|}\right) \leq -\frac{\sigma}{2}\left(2 - \frac{|V|+1-2}{|V|}\right) = -\frac{\sigma}{2}\left(2 - \frac{|V|-1}{|V|}\right) = -\frac{\sigma}{2}\left(1 + \frac{1}{|V|}\right) < -\frac{\sigma}{2}$, which is a contradiction. Also, if action a_j were not used at all during the first $|V|+1$ positions of S then some other action a_i had to be used more than once, leading to the same contradiction.

Given that each action is used exactly once in the first $|V|+1$ steps, Windfall$_{b_j}(S,t) \geq -\frac{\sigma}{2}, \forall t \in \{1 \ldots |V|+1\}, \forall j \in \{1 \ldots |V|+1\}$, because Windfall$_{b_j}(S,t)$ is the sum of at most one negative term (i.e. $-\frac{\sigma}{2}$) and several positive ones (i.e. $\frac{\sigma}{2 \times |V|}$). As for the beneficiaries b' and b'', their worst losses depend on the subset V' of elements in S that go before the first occurrence of action $a_{|V|+1}$. The windfall of beneficiary b'' decreases monotonically from zero until action $|V|+1$ is chosen, then abruptly becomes positive and thereafter decreases monotonically to zero. Thus the worst loss for beneficiary b'' occurs right before action $a_{|V|+1}$:

$$-\frac{\sigma}{2} \leq \text{Windfall}_{b''}(S, |V'|) = -\sum_{v \in V'} v \Rightarrow \sum_{v \in V'} v \leq \frac{\sigma}{2}$$

The worst loss for beneficiary b' occurs immediately after action $a_{|V|+1}$, since

$\text{Windfall}_{b'}(S,t) = -\text{Windfall}_{b''}(S,t), \forall S, \forall t$:

$$-\frac{\sigma}{2} \leq \text{Windfall}_{b'}(S, |V'|+1) = \sum_{v \in V'} v - \sigma \Rightarrow \sum_{v \in V'} v \geq \frac{\sigma}{2}$$

Consequently $\sum_{v \in V'} v = \frac{\sigma}{2} \Rightarrow \sum_{v \in V - V'} v = \frac{\sigma}{2}$, concluding the proof for the first part of the claim.

I now prove the ONLY-IF part of the claim: if there exists a partition of V into V' and $V - V'$ such that $\sum_{v \in V'} v = \sum_{v \in V - V'} v$, then there exists a sequence of length λ with a worst loss of $-\frac{\sigma}{2}$.

Let S be a periodic sequence whose period consists of an arbitrary permutation of the actions in $\{a_i | v_i \in V'\}$ followed by action $a_{|V|+1}$ and an arbitrary permutation of the actions in $\{a_i | v_i \in V - V'\}$. Because S is periodic, the sequence of windfalls produced by S is also periodic. Therefore it is enough to verify that windfalls are greater than or equal to $-\frac{\sigma}{2}$ during the first $|V|+1$ time steps. Note that each action is used exactly once during the first $|V|+1$ steps, so I can verify the worst loss for each beneficiary using arguments similar to those in the first part of the claim's proof. There is a single action a_j that affects beneficiary b_j's windfall in a negative way, so $\text{Windfall}_{b_j}(S,t) \geq -\frac{\sigma}{2}$. The windfall of beneficiary b' improves monotonically from zero to $\frac{\sigma}{2}$ during the first $|V'|$ steps; then becomes $-\frac{\sigma}{2}$ at the next step as a result of action $a_{|V|+1}$; then improves monotonically afterwards. The windfall of beneficiary b'' worsens monotonically, until it reaches $-\frac{\sigma}{2}$ after step $|V'|$; then it becomes equal to $\frac{\sigma}{2}$ due to action $a_{|V|+1}$ and will remain positive until the end of the period.

Sequence S is proof that a worst loss of $-\frac{\sigma}{2}$ is possible, and since $\text{WL} \leq -\frac{\sigma}{2}$, then $\text{WL} = -\frac{\sigma}{2}$ is optimal. Therefore a correct WORST LOSS OPTIMIZATION PROBLEM solver must find some sequence with the same performance. This concludes the proof for the second part of the claim.

Therefore the answer to any instance of the PARTITION PROBLEM can be decided by solving the corresponding WORST LOSS OPTIMIZATION PROBLEM instance. To summarize: compute the windfalls for the first $|V| + 1$ steps of the optimal-WL sequence. If at any time some windfall was strictly less than $-\frac{\sigma}{2}$, then the partitioning is impossible; otherwise make a multiset V' consisting of all v_j such that action a_j was encountered before action $a_{|V|+1}$, and return the partition $(V', V - V')$.

The transformations between the two problems are obviously polynomial in the size of the input, so I conclude that the existence of a polynomial time algorithm for the WORST LOSS OPTIMIZATION PROBLEM implies $P = NP$. □

Theorem A.2. Let $S \in \mathbb{S}$ be an infinite sequence for which I can lower-bound all windfalls by some constant r (i.e. $\text{Windfall}_b(S, t) \geq r, \forall b \in \mathbb{B}$ and $\forall t \in \mathbb{N}$). Then the sequence $U(S)$ converges to $U^\star$ (where $U(S)$ denotes the sequence of utility profiles visited by the sequence of actions S).

Proof. Since $\mathbb{U}$ is bounded (all rewards are finite), Bolzano-Weierstrass theorem [164] guarantees that the sequence $U(S)$ has a convergent subsequence W. Let U' be the limit of W. W_t denotes the t^{th} term of W and $W_{b,t}$ denotes the component corresponding to beneficiary b of the utility profile W_t.

Since $\text{Windfall}_b(S, t) \geq r$, it follows that $U_b(S_{1:t}) - U_b^\star \geq \frac{r}{t}, \forall b \in \mathbb{B}$. W is a subsequence of $U(S)$, so $\forall t' \in \mathbb{N}, \exists t \geq t'$ such that $W_{t'} = U(S_{1:t})$. If follows that $W_{b,t'} - U_b^\star = U_b(S_{1:t}) - U_b^\star \geq \frac{r}{t} \geq \frac{r}{t'}$. At the limit (as $t' \to \infty$) this inequality becomes $U'_b - U_b^\star \geq 0, \forall b \in \mathbb{B}$. Since $U^\star$ is Pareto-optimal in $\mathbb{U}$ and $U' \in \mathbb{U}$ ($\mathbb{U}$ is closed, so it contains all its limit points), it must be that $U' = U^\star$, proving that any convergent subsequence of $U(S)$ must converge to $U^\star$. But any subsequence of $U(S)$ is bounded ($U(S) \in \mathbb{U}$, which is bounded), so that subsequence must have a subsubsequence that converges to $U^\star$. I will now show that this is sufficient for $U(S)$ to converge to $U^\star$.

Let me assume $U(S)$ does not converge to $U^\star$; it follows that $\exists \epsilon > 0$ such that $\forall t'$,

$\exists f(t') > t'$ such that $\|U(S_{1:f(t')}\| \geq \epsilon$. Since the sequence $U(S)$ must have an infinite number of terms at least ϵ away from $U^\star$, I can build a subsequence consisting exclusively of such terms (i.e. at least ϵ away from $U^\star$). This leads to a contradiction, because such a subsequence cannot possibly have a subsequence converging to $U^\star$. The result follows. □

Theorem A.3. Let the vector F consist of the coordinates of an arbitrary point in the n_a-dimensional unit simplex. Then any sequence D'' generated by any of my methods (using F for $F^\star$) will use the actions in the proportions prescribed by F at the limit. Formally: $\lim_{t\to\infty} \frac{1}{t} k_a(D''_{1:t}) = F_a, \forall a \in \mathbb{A}$.

Proof. I start by noting that $F_a = 0$ implies $k_a(D''_{1:t}) = 0$, since I never use action a if $F_a = 0$). The result holds trivially for such actions.

I now focus on the actions with strictly positive F values. It follows from Lemma 5 that:

$$\forall i, j \in \{1 \dots n_a\}, \forall t \in \mathbb{N} : \; k_i(D''_{1:t}) \leq \frac{k_j(D''_{1:t}) + \theta_j}{F_j} F_i + 1 - \theta_i$$

and I sum over all indices $i = 1 \dots n_a$ (using $k_i(D''_{1:t}) = \frac{k_i(D''_{1:t}) + \theta_i}{F_i} F_i - \theta_i$ for $i = j$):

$$\sum_{i=1}^{n_a} k_i(D''_{1:t}) \leq \frac{k_j(D''_{1:t}) + \theta_j}{F_j} \sum_{i=1}^{n_a} F_i + \sum_{i=1}^{n_a - 1} 1 - \sum_{i=1}^{n_a} \theta_i$$

but $\sum_{i=1}^{n_a} k_i(D''_{1:t}) = t$, and $\sum_{i=1}^{n_a} F_i = 1$, so:

$$t \leq \frac{k_j(D''_{1:t}) + \theta_j}{F_j} + n_a - 1 - \sum_{i=1}^{n_a} \theta_i$$

$$\frac{k_j(D''_{1:t})}{t} \geq F_j\left(1 - \frac{n_a - 1 - \sum_i^{n_a} \theta_i}{t}\right) - \frac{\theta_j}{t}$$

$$\frac{k_j(D''_{1:t})}{t} - F_j \geq -\frac{\theta_j + F_j(n_a - 1 - \sum_i^{n_a} \theta_i)}{t}.$$

I repeat the process, but sum over all indices j this time:

$$k_j(D''_{1:t}) \geq \frac{k_i(D''_{1:t}) + \theta_i - 1}{F_i} F_j - \theta_j$$

(using the equality $k_i(D''_{1:t}) = \frac{k_i(D''_{1:t}) + \theta_i - 1}{F_i} F_i + 1 - \theta_i$ when $j = i$)

$$\sum_{j=1}^{n_a} k_j(D''_{1:t}) \geq 1 + \frac{k_i(D''_{1:t}) + \theta_i - 1}{F_i} \sum_{j=1}^{n_a} F_j - \sum_{j=1}^{n_a} \theta_j$$

$$t \geq 1 + \frac{k_i(D''_{1:t}) + \theta_i - 1}{F_i} - \sum_{j=1}^{n_a} \theta_j$$

$$\frac{k_i(D''_{1:t})}{t} \leq F_i + \frac{1 - \theta_i + F_i(-1 + \sum_{j=1}^{n_a} \theta_j)}{t}$$

$$\frac{k_i(D''_{1:t})}{t} - F_i \leq \frac{1 - \theta_i + F_i(-1 + \sum_{j=1}^{n_a} \theta_j)}{t}.$$

$$-\frac{\theta_j + F_j(n_a - 1 - \sum_i^{n_a} \theta_i)}{t} \leq \frac{k_j(D''_{1:t})}{t} - F_j \leq \frac{1 - \theta_j + F_j(-1 + \sum_{i=1}^{n_a} \theta_i)}{t}. \tag{A.1}$$

Note that both bounds collapse to zero as $t \to \infty$, so $\lim_{t\to\infty} \frac{1}{t} k_j(D''_{1:t}) = F_j$. The result follows. □

Theorem A.4. Problem 3 is NP-hard.

Proof. For this proof I use a reduction from PARTITION PROBLEM [66]. Given a multiset of positive integer numbers $V = \{v_1, \dots v_{|V|}\}$, I associate a problem with $2|V| + 1$ actions $(a_1, \dots, a_n, a'_1, \dots, a'_{|V|}, a_\sigma)$ and $2|V| + 4$ beneficiaries $(b_+, b'_+, b_{+1}, \dots, b_{+|V|}, b_-, b'_-, b_{-1}, \dots b_{-|V|})$. The rewards are defined such that $U^\star = [0 \dots 0]$,

so the rewards coincide with the X values. The rewards (and thus the X values) are presented in the table below:

	"Positive" beneficiaries						"Negative" beneficiaries					
	b_+	b'_+	b_{+1}	b_{+2}	$\dots$	$b_{+\lvert V\rvert}$	b_-	b'_-	b_{-1}	b_{-2}	$\dots$	$b_{-\lvert V\rvert}$
a_1	v_1	0	$\frac{\sigma}{2}$	0	$\dots$	0	$-v_1$	0	$-\frac{\sigma}{2}$	0	$\dots$	0
a'_1	0	v_1	$\frac{\sigma}{2}$	0	$\dots$	0	0	$-v_1$	$-\frac{\sigma}{2}$	0	$\dots$	0
a_2	v_2	0	0	$\frac{\sigma}{2}$	$\dots$	0	$-v_2$	0	0	$-\frac{\sigma}{2}$	$\dots$	0
a'_2	0	v_2	0	$\frac{\sigma}{2}$	$\dots$	0	0	$-v_2$	0	$-\frac{\sigma}{2}$	$\dots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
a_n	v_n	0	0	0	$\dots$	$\frac{\sigma}{2}$	$-v_n$	0	0	0	$\dots$	$-\frac{\sigma}{2}$
a'_n	0	v_n	0	0	$\dots$	$\frac{\sigma}{2}$	0	v_n	0	0	$\dots$	$-\frac{\sigma}{2}$
a_σ	$-\frac{\sigma}{2}$	$-\frac{\sigma}{2}$	$-\frac{\sigma}{2}$	$-\frac{\sigma}{2}$	$\dots$	$-\frac{\sigma}{2}$	$\frac{\sigma}{2}$	$\frac{\sigma}{2}$	$\frac{\sigma}{2}$	$\frac{\sigma}{2}$	$\dots$	$\frac{\sigma}{2}$

where $\sigma = \sum_{i=1}^{|V|} v_i$. Note that I have two types of beneficiaries ("positive" and "negative"), and for every positive beneficiary there is a negative beneficiary who receives the same values as the first beneficiary, but multiplied by -1. The reason for this choice is two-fold. First, it makes sure $U^\star = [0 \dots 0]$ (since no beneficiary can get a strictly positive utility without another getting a strictly negative utility). Second, $LB_1 = LB_2$ for any $F^\star$, since a "positive" beneficiary has the same X values as the corresponding "negative" beneficiary, multiplied by -1.

In this construction I reuse some elements from the previous NP-completeness proof (Theorem A.1). I associate actions with the elements $v_i \in V$ and I use beneficiaries b_+, b'_+, b_-, b'_- to map sums of elements from V into worst losses. The novel element is that I associate two actions (a_i and a'_i) with each $v_i \in V$ so that multiple $F^\star$ vectors exist. I also use action a_σ and beneficiaries b_{+i} and b_{-i} to force the solver to choose only one of a_i and a'_i when choosing an $F^\star$ vector. I map $F^\star$ into a partition of V by grouping together all v_i

elements for which $F^\star$ chose to use a_i over a'_i.

Claim *There exists an equal-sum partition for V if and only if $|LB_2| = \frac{\sigma}{2}$.*

Note that a_σ is indispensable in order for the utility profile $[0 \dots 0]$ to be achievable (without a_σ at least one of the beneficiaries b_- and b'_- will have a strictly negative utility). It follows that $|LB_2| \geq \frac{\sigma}{2}$ (by Equation 4.19) and $|LB_1| \geq \frac{\sigma}{2}$ (by Equation 4.18). Additionally, any valid $F^\star$ must use at least one of a_i or a'_i (because of beneficiary b_{+i} or beneficiary b_{-i}).

I prove the ONLY-IF part of the claim first. If there exist an equal-sum partition of V into V' and $V - V'$, then there exist $F^\star$ vectors that produce $|LB_1| = |LB_2| = \frac{\sigma}{2}$ (which are optimal since I already showed that $|LB_1|$ and $|LB_2| \geq \frac{\sigma}{2}$); an example of such an $F^\star$ vector is the following:

$$F^\star_{a_\sigma} = \frac{1}{2|V|+1}; \quad F^\star_{a_i} = \begin{cases} \frac{1}{2|V|+1} & \text{if } v_i \in V' \\ 0 & \text{otherwise} \end{cases}; \quad F^\star_{a'_i} = \begin{cases} 0 & \text{if } v_i \in V' \\ \frac{1}{2|V|+1} & \text{otherwise.} \end{cases}$$

I now prove the IF part of the claim. If the optimal subset has $|LB_2| = \frac{\sigma}{2}$, then no valid $F^\star$ can use both a_i and a'_i for any $i = 1 \dots |V|$ (otherwise the windfall lower bound of either beneficiary b_{+i} or b_{-i} would be worse than $-\frac{\sigma}{2}$). This implies that the $F^\star$ values for all chosen a_i and a'_i are equal to $F^\star_\sigma = \frac{1}{n+1}$. Because $\sum_{a \in \mathbb{A}} X_{a,b_+} F^\star_a = 0$, the sum of positive X values for b_+ must be equal to $\frac{\sigma}{2}$, and the equal-sum partition may be built as: $V' = \{v_i | F^\star_{a_i} > 0\}$, $V - V' = \{v_i | F^\star_{a_i} = 0\}$. □

A.2 WL Derivations for GF$^{\theta}$

A.2.1 WL Derivation for $C(t) = t$

It results from Equation A.1 (proof of Theorem A.3) that the following functions:

$$\Delta_l(a) = -\theta_a + F_a^\star \sum_i^{n_a^\star} \theta_i - F_a^\star(n_a - 1)$$

$$\Delta_h(a) = -\theta_a + F_a^\star \sum_{i=1}^{n_a^\star} \theta_i + 1 - F_a^\star$$

satisfy Equation 4.10 when $C(t) = t$. It follows from Theorem 4.1 that

$$\text{Windfall}_b(D'', t) \geq (1 - n_a^\star) \sum_{a=1}^{n_a^\star} F_a^\star X_{a,b}^+ + \sum_{a=1}^{n_a^\star} (1 - F_a^\star) X_{a,b}^- + \sum_{a=1}^{n_a^\star} \left[F_a^\star \left(\sum_i^{n_a^\star} \theta_i \right) - \theta_a \right] X_{a,b}.$$

I use Equation 4.2 in the form $\sum_{a=1}^{n_a^\star} F_a^\star X_{a,b}^- = -\sum_{a=1}^{n_a^\star} F_a^\star X_{a,b}^+$:

$$\text{Windfall}_b(D'', t) \geq (n_a^\star - 2) \sum_{a=1}^{n_a^\star} F_a^\star X_{a,b}^- + \sum_{a=1}^{n_a^\star} X_{a,b}^- - \sum_{a=1}^{n_a^\star} \theta_a X_{a,b} + \left(\sum_i^{n_a} \theta_i \right) \sum_{a=1}^{n_a^\star} F_a^\star X_{a,b}.$$

I use Equation 4.2 again, this time on the last term:

$$\text{Windfall}_b(D'', t) \geq (n_a^\star - 2) \sum_{a=1}^{n_a^\star} F_a^\star X_{a,b}^- + \sum_{a=1}^{n_a^\star} X_{a,b}^- - \sum_{a=1}^{n_a^\star} \theta_a X_{a,b} \tag{A.2}$$

A.2.2 WL Derivation for $C(t) = \frac{k_{a'}(D''_{1:t})}{F^\star_{a'}}$

By on Lemma 5, the following functions satisfy Equation 4.10 when $C(t) = \frac{k_{a'}(D''_{1:t})}{F^\star_{a'}}$:

$$\Delta_l(a) = \begin{cases} -\theta_a + F^\star_a \frac{\theta_{a'}}{F^\star_{a'}} - \frac{F^\star_a}{F^\star_{a'}} & \text{if } a \neq a' \\ 0 & \text{if } a = a' \end{cases}$$

$$\Delta_h(a) = \begin{cases} -\theta_a + F^\star_a \frac{\theta_{a'}}{F^\star_{a'}} + 1 & \text{if } a \neq a' \\ 0 & \text{if } a = a' \end{cases}$$

It follows from Theorem 4.1 that:

$$\begin{aligned}
\text{Windfall}_b(D'', t) &\geq -\sum_{\substack{a=1 \\ a\neq a'}}^{n^\star_a} \theta_a X_{a,b} + \frac{\theta_{a'}}{F^\star_{a'}} \sum_{\substack{a=1 \\ a\neq a'}}^{n^\star_a} F^\star_a X_{a,b} - \frac{1}{F^\star_{a'}} \sum_{\substack{a=1 \\ a\neq a'}}^{n^\star_a} F^\star_a X^+_{a,b} + \sum_{\substack{a=1 \\ a\neq a'}}^{n^\star_a} X^-_{a,b} \\
&\geq -\sum_{a=1}^{n^\star_a} \theta_a X_{a,b} + \frac{\theta_{a'}}{F^\star_{a'}} \sum_{a=1}^{n^\star_a} F^\star_a X_{a,b} - \frac{1}{F^\star_{a'}} \sum_{\substack{a=1 \\ a\neq a'}}^{n^\star_a} F^\star_a X^+_{a,b} + \sum_{\substack{a=1 \\ a\neq a'}}^{n^\star_a} X^-_{a,b} \\
&\geq -\sum_{a=1}^{n^\star_a} \theta_a X_{a,b} - \frac{1}{F^\star_{a'}} \sum_{\substack{a=1 \\ a\neq a'}}^{n^\star_a} F^\star_a X^+_{a,b} + \sum_{\substack{a=1 \\ a\neq a'}}^{n^\star_a} X^-_{a,b} \\
&\geq -\sum_{a=1}^{n^\star_a} \theta_a X_{a,b} + \frac{1}{F^\star_{a'}} \sum_{a=1}^{n^\star_a} F^\star_a X^-_{a,b} + \sum_{a=1}^{n^\star_a} X^-_{a,b} + |X_{a',b}|. \qquad \text{(A.3)}
\end{aligned}$$

A.2.3 WL Derivation for $C(t) = \frac{k_{a'}(D''_{1:t}) + \theta_{a'}}{F^\star_{a'}}$

The $C(t)$ in this subsection is a translation of the previous subsection by a constant $\delta = \frac{\theta_{a'}}{F^\star_{a'}}$.

It follows from the argument in footnote 5 (page 59) that the windfall bounds I can derive for

$C(t) = \frac{k_{a'}(D''_{1:t})+\theta_{a'}}{F^\star_{a'}}$ are identical with those I derived for $C(t) = \frac{k_{a'}(D''_{1:t})}{F^\star_{a'}}$ (i.e. Equation A.3).

A.2.4 WL Derivation for $C(t) = \min_{a'} \frac{k_{a'}(D''_{1:t})}{F^\star_{a'}}$

Starting from Lemma 5:

$$-\frac{1}{F^\star_{a'}} \leq \frac{k_a(D''_{1:t}) + \theta_a}{F^\star_a} - \frac{k_{a'}(D''_{1:t}) + \theta_{a'}}{F^\star_{a'}} \leq \frac{1}{F^\star_a}$$

$$-\frac{1-\theta_{a'}}{F^\star_{a'}} - \frac{\theta_a}{F^\star_a} \leq \frac{k_a(D''_{1:t})}{F^\star_a} - \frac{k_{a'}(D''_{1:t})}{F^\star_{a'}} \leq \frac{1-\theta_a}{F^\star_a} + \frac{\theta_{a'}}{F^\star_{a'}}$$

$$0 \leq \frac{k_a(D''_{1:t})}{F^\star_a} - \min_{a'} \frac{k_{a'}(D''_{1:t})}{F^\star_{a'}} \leq \frac{1-\theta_a}{F^\star_a} + \max_{a''} \frac{\theta_{a'}}{F^\star_{a''}}$$

$$0 \leq k_a(D''_{1:t}) - F^\star_a \min_{a'} \frac{k_{a'}(D''_{1:t})}{F^\star_{a'}} \leq 1 - \theta_a + F^\star_a \max_{a''} \frac{\theta_{a'}}{F^\star_{a''}}.$$

It follows from Theorem 4.1 with $C(t) = \min_{a'} \frac{k_{a'}(D''_{1:t})}{F^\star_{a'}}$, $\Delta_l(a) = 0$ and $\Delta_h(a) = 1 - \theta_a + F^\star_a \max_{a''} \frac{\theta_{a'}}{F^\star_{a''}}$ that:

$$\begin{aligned}\text{Windfall}_b(D'', t) &\geq \sum_{a=1}^{n^\star_a} (1-\theta_a) X^-_{a,b} + (\max_{a''} \frac{\theta_{a''}}{F^\star_{a''}}) \sum_{a=1}^{n^\star_a} F^\star_a X^-_{a,b} \\ &\geq \sum_{a=1}^{n^\star_a} X^-_{a,b} - \sum_{a=1}^{n^\star_a} \theta_a X^-_{a,b} + (\max_{a''} \frac{\theta_{a''}}{F^\star_{a''}}) \sum_{a=1}^{n^\star_a} F^\star_a X^-_{a,b} \end{aligned} \tag{A.4}$$

A.2.5 WL Derivation for $C(t) = \min_{a'} \frac{k_{a'}(D''_{1:t})+\theta_{a'}}{F^\star_{a'}}$

Starting from Lemma 5:

$$-\frac{1}{F^\star_{a'}} \leq \frac{k_a(D''_{1:t}) + \theta_a}{F^\star_a} - \frac{k_{a'}(D''_{1:t}) + \theta_{a'}}{F^\star_{a'}} \leq \frac{1}{F^\star_a}$$

$$0 \leq \frac{k_a(D''_{1:t}) + \theta_a}{F^\star_a} - \min_{a'} \frac{k_{a'}(D''_{1:t}) + \theta_{a'}}{F^\star_{a'}} \leq \frac{1}{F^\star_a}$$

$$-\theta_a \leq k_a(D''_{1:t}) - F^\star_a \min_{a'} \frac{k_{a'}(D''_{1:t}) + \theta_{a'}}{F^\star_{a'}} \leq 1 - \theta_a.$$

Now I use Theorem 4.1 with $C(t) = \min_{a'} \frac{k_{a'}(D''_{1:t})+\theta_{a'}}{F^\star_{a'}}$, $\Delta_l(a) = -\theta_a$ and $\Delta_h(a) = 1 - \theta_a$:

$$\begin{aligned}\text{Windfall}_b(D'', t) &\geq \sum_{a=1}^{n^\star_a} X^-_{a,b} - \sum_{a=1}^{n^\star_a} \theta_a X^-_{a,b} - \sum_{a=1}^{n^\star_a} \theta_a X^+_{a,b} \\ &\geq -\sum_{a=1}^{n^\star_a} \theta_a X_{a,b} + \sum_{a=1}^{n^\star_a} X^-_{a,b}. \end{aligned} \tag{A.5}$$

Note that the bound in Equation A.5 is tighter than or equal to the bound in Equation A.2, provided $n^\star_a \geq 2$. This causes no loss in generality, since $n^\star_a = 1$ corresponds to the trivial case where a single action is to be played over and over, all windfall values are always zero, and the θ values are irrelevant.

The bound in Equation A.5 is also tighter than or equal to the bound in Equation A.4, provided $\theta_a \geq 0, \forall a$ (Equation 4.13).

A.2.6 WL Derivation for $C(t) = \max_{a'} \frac{k_{a'}(D''_{1:t})+\theta_{a'}}{F^\star_{a'}}$

Starting from Lemma 5:

$$-\frac{1}{F^\star_{a'}} \leq \frac{k_a(D''_{1:t})+\theta_a}{F^\star_a} - \frac{k_{a'}(D''_{1:t})+\theta_{a'}}{F^\star_{a'}} \leq \frac{1}{F^\star_a}$$

$$-\max_{\alpha''} \frac{1}{F^\star_{a''}} \leq \frac{k_a(D''_{1:t})+\theta_a}{F^\star_a} - \max_{a'} \frac{k_{a'}(D''_{1:t})+\theta_{a'}}{F^\star_{a'}} \leq 0$$

$$-\frac{F^\star_a}{\min_{\alpha''} F^\star_{a''}} - \theta_a \leq k_a(D''_{1:t}) - F^\star_a \max_{a'} \frac{k_{a'}(D''_{1:t})+\theta_{a'}}{F^\star_{a'}} \leq -\theta_a.$$

It follows from Theorem 4.1 with $C(t) = \max_{a'} \frac{k_{a'}(D''_{1:t})+\theta_{a'}}{F^\star_{a'}}$, $\Delta_l(a) = -\frac{F^\star_a}{\min_{\alpha''} F^\star_{a''}} - \theta_a$ and $\Delta_h(a) = -\theta_a$ that:

$$\begin{aligned}\text{Windfall}_b(D'', t) &\geq -\sum_{a=1}^{n^\star_a} \theta_a X_{a,b} - \frac{F^\star_{a''}}{\min_{a''} F^\star_{a''}} \sum_{a=1}^{n^\star_a} F^\star_a X^+_{a,b} \\ &\geq -\sum_{a=1}^{n^\star_a} \theta_a X_{a,b} + \frac{F^\star_{a''}}{\min_{a''} F^\star_{a''}} \sum_{a=1}^{n^\star_a} F^\star_a X^-_{a,b}. \end{aligned} \tag{A.6}$$

A.2.7 WL Derivation for $C(t) = \max_{a'} \frac{k_{a'}(D''_{1:t})}{F^\star_{a'}}$

Starting from Lemma 5:

$$-\frac{1}{F^\star_{a'}} \leq \frac{k_a(D''_{1:t})+\theta_a}{F^\star_a} - \frac{k_{a'}(D''_{1:t})+\theta_{a'}}{F^\star_{a'}} \leq \frac{1}{F^\star_a}$$

$$-\frac{1-\theta_{a'}}{F^\star_{a'}} - \frac{\theta_a}{F^\star_a} \leq \frac{k_a(D''_{1:t})}{F^\star_a} - \frac{k_{a'}(D''_{1:t})}{F^\star_{a'}} \leq \frac{1-\theta_a}{F^\star_a} + \frac{\theta_{a'}}{F^\star_{a'}}$$

$$-\max_{\alpha''} \frac{1-\theta_{a''}}{F^\star_{a''}} - \frac{\theta_a}{F^\star_a} \leq \frac{k_a(D''_{1:t})}{F^\star_a} - \max_{a'} \frac{k_{a'}(D''_{1:t})}{F^\star_{a'}} \leq 0$$

$$-F^\star_a \max_{\alpha''} \frac{1-\theta_{a''}}{F^\star_{a''}} - \theta_a \leq k_a(D''_{1:t}) - F^\star_a \max_{a'} \frac{k_{a'}(D''_{1:t})}{F^\star_{a'}} \leq 0.$$

It follows from Theorem 4.1 with $C(t) = \max_{a'} \frac{k_{a'}(D''_{1:t})}{F^\star_{a'}}$, $\Delta_l(a) = -F^\star_a \max_{a''} \frac{1-\theta_{a''}}{F^\star_{a''}} - \theta_a$ and $\Delta_h(a) = 0$ that:

$$\begin{aligned}\text{Windfall}_b(D'', t) &\geq -\sum_{a=1}^{n^\star_a} \theta_a X^+_{a,b} - (\max_{a''} \frac{1-\theta_{a''}}{F^\star_{a''}}) \sum_{a=1}^{n^\star_a} F^\star_a X^+_{a,b} \\ &\geq -\sum_{a=1}^{n^\star_a} \theta_a X^+_{a,b} + (\max_{a''} \frac{1-\theta_{a''}}{F^\star_{a''}}) \sum_{a=1}^{n^\star_a} F^\star_a X^-_{a,b}. \end{aligned} \tag{A.7}$$

A.3 Tight Example for Equation 4.21

It is customary in the literature [176] to accompany the approximation ratio proof of an algorithm with a *tight example*, an infinite family of problem instances of arbitrary size for which the algorithm's performance is equal to ϵ times the optimal result. The purpose of the tight example is to show that the analysis that led to the approximation ratio cannot be improved. I argue that this endeavor is less important when the approximation ratio is not a constant, but a function of the problem instance (e.g. Equation 4.21). Imagine two approximation ratio functions, with the first always smaller then or equal to the second on every problem instance. If the two functions are equal on all instances of some tight example, then they are both considered "tight," although the first approximation ratio is intuitively tighter than the second.

I will present a tight example for the approximation ratio in Equation 4.21. Then I will show that this tight example also applies to a weaker approximation ratio function.

For any integer value $n \geq 2$ we define the following problem instance P_n with n beneficiaries $(b_1 \dots b_n)$, and n actions $(a_1 \dots a_n)$. $R_{b_j}(a_i)$ is equal to 1 if $i = j$ and 0 otherwise. $U^\star = [\frac{1}{n}, \dots, \frac{1}{n}]$, $F^\star = [\frac{1}{n}, \dots, \frac{1}{n}]$ and $X_{ai,bj} = \frac{n-1}{n}$ if $i = j$ and $-\frac{1}{n}$ otherwise.

Note that $\text{WL}^0 = \text{WL}^1 = -1 + \frac{1}{n} = \text{WL}^{\text{OPT}}(P_n)$, which means both $\mathbf{GF^0}$ and $\mathbf{GF^1}$ solve P_n optimally. It follows that $\mathbf{GF^\theta}$ also solves all P_n optimally. This means the approximation

ratio in Equation 4.21 is tight for $\mathbf{GF^0}$, $\mathbf{GF^1}$, and $\mathbf{GF^\theta}$.

I now introduce a new approximation ratio: $\overline{\mathrm{WL}} = \max(\overline{\mathrm{WL}^{(0)}}, \overline{\mathrm{WL}^{(1)}})$, where $\overline{\mathrm{WL}^{(0)}}$ and $\overline{\mathrm{WL}^{(1)}}$ are defined just like WL^0 and WL^1, with the difference that every positive $X_{a,b}$ is replaced by $\max_a X_{a,b}$ and each negative $X_{a,b}$ is replaced by $\min_a X_{a,b}$.

Note 1: clearly $\mathrm{WL}^{(0)} \geq \overline{\mathrm{WL}^{(0)}}$ and $\mathrm{WL}^{(1)} \geq \overline{\mathrm{WL}^{(1)}}$, so this new bound function is intuitively looser than the original bound.

Note 2: for any P_n: all positive X values are equal to $1 - \frac{1}{n}$, and all negative values are equal to $-\frac{1}{n}$. Therefore, $\mathrm{WL}^{(0)} = \overline{\mathrm{WL}^{(0)}}$ and $\mathrm{WL}^{(1)} = \overline{\mathrm{WL}^{(1)}}$.

In conclusion, the new bound is tight based on the $\{P_n\}$ tight example, yet it is intuitively looser than the original bound.

Appendix B: Test Problems

This appendix contains a number of test problems used in the empirical studies in Chapter 5 (Section B.1) and Chapter 6 (Section B.2). For each test problem I present the input, the leximin-optimal utility profile and how it can be achieved.

B.1 Stateful Test Problems

All instances are strongly connected and have rational rewards, so there exists a (unique) leximin-optimal utility profile achievable at the limit; I refer to it as $U^{\infty\star}$, and let $F^{\infty\star}$ be a long-term action frequency profile corresponding to $U^{\infty\star}$.

Example B.1. This stateful domain example consists of three states and three beneficiaries. All edges (actions) coming out of node (state) S_i go into node $S_{(i+1) \pmod 3}$, so decisions affect the rewards, but not the sequence of visited states.

		Beneficiaries			$F^{\infty\star}$
		b_1	b_2	b_3	
$S_0 \to S_1$	a_0	0	0	0	8/24
$S_1 \to S_2$	a_0	0	10	10	8/24
	a_1	10	10	0	0
$S_2 \to S_0$	a_0	20	0	0	5/24
	a_1	0	10	0	2/24
	a_2	0	0	20	1/24

$U^{\infty\star} = [\frac{25}{6}, \frac{25}{6}, \frac{25}{6}]$, and it is achievable in finite time. There are an infinity of $F^{\infty\star}$ vectors.

Example B.2. This stateful domain example consists of six beneficiaries, five states and one action between each pair of states (i.e. a fully connected graph with self-loops on each state).

	Beneficiaries						$F^{\infty\star}$
	b_1	b_2	b_3	b_4	b_5	b_6	
$S_0 \rightarrow S_0$	3	5	5	2	4	0	9/28
$S_0 \rightarrow S_1$	2	5	5	3	3	0	0
$S_0 \rightarrow S_2$	0	1	5	2	0	3	0
$S_0 \rightarrow S_3$	5	4	4	4	3	1	0
$S_0 \rightarrow S_4$	3	1	3	4	5	0	0
$S_1 \rightarrow S_0$	3	2	0	5	4	2	0
$S_1 \rightarrow S_1$	0	5	1	0	2	2	0
$S_1 \rightarrow S_2$	2	5	3	0	0	1	0
$S_1 \rightarrow S_3$	1	2	0	2	1	1	0
$S_1 \rightarrow S_4$	1	1	0	4	1	1	0
$S_2 \rightarrow S_0$	4	4	0	2	4	3	0
$S_2 \rightarrow S_1$	2	3	5	3	3	2	0
$S_2 \rightarrow S_2$	2	3	0	2	2	2	0
$S_2 \rightarrow S_3$	5	3	4	2	2	2	2/28
$S_2 \rightarrow S_4$	2	0	2	2	0	4	0
$S_3 \rightarrow S_0$	5	5	5	5	2	1	0
$S_3 \rightarrow S_1$	3	0	3	3	1	2	0
$S_3 \rightarrow S_2$	5	5	3	0	2	5	2/28
$S_3 \rightarrow S_3$	2	2	2	5	2	2	0
$S_3 \rightarrow S_4$	5	5	4	3	2	1	0
$S_4 \rightarrow S_0$	2	4	1	5	4	2	0
$S_4 \rightarrow S_1$	2	0	1	3	5	2	0
$S_4 \rightarrow S_2$	2	2	2	0	4	1	0
$S_4 \rightarrow S_3$	0	5	2	5	2	3	0

Continued on next page

Continued from previous page

	Beneficiaries						$F^{\infty\star}$
	b_1	b_2	b_3	b_4	b_5	b_6	
$S_4 \to S_4$	5	2	2	5	3	5	15/28

$U^{\infty\star} = [\frac{61}{14}, \frac{13}{4}, \frac{89}{28}, \frac{97}{28}, \frac{89}{28}, \frac{89}{28}]$. There is an unique $F^{\infty\star}$ profile, and the $F^{\infty\star}$-induced graph consists of 3 strongly connected components ($\{S_0\}$, $\{S_2, S_3\}$ and $\{S_4\}$), so $U^{\infty\star}$ can only be achieved at the limit.

Example B.3. This example has 4 beneficiaries, 5 states, 2 actions between every pair of distinct states and 1 self loop on each state.

	Beneficiaries				$F^{\infty\star}$
	b_1	b_2	b_3	b_4	
$S_0 \to S_0$	5	5	1	4	0
$S_0 \to S_1$	1	0	7	1	0
	5	1	49	7	0
$S_0 \to S_2$	5	3	7	8	0
	0	5	4	6	0
$S_0 \to S_3$	3	4	57	13	0
	8	10	1	31	0
$S_0 \to S_4$	11	2	4	8	0
	2	50	3	22	3338/142493
$S_1 \to S_0$	3	7	1	33	0
	1	4	24	38	0
$S_1 \to S_1$	9	10	8	12	0
$S_1 \to S_2$	33	10	4	2	3762/142493
	1	3	22	2	0

Continued on next page

Continued from previous page

	Beneficiaries				$F^{\infty\star}$
	b_1	b_2	b_3	b_4	
$S_1 \to S_3$	8	7	2	0	0
	25	12	0	5	0
$S_1 \to S_4$	19	18	14	0	43821/142493
	11	5	8	3	0
$S_2 \to S_0$	1	1	5	6	0
	4	5	13	2	0
$S_2 \to S_1$	12	2	0	7	0
	1	16	4	3	0
$S_2 \to S_2$	13	4	19	17	0
$S_2 \to S_3$	11	13	37	2	3762/142493
	1	16	24	4	0
$S_2 \to S_4$	11	0	2	9	0
	1	8	24	3	0
$S_3 \to S_0$	12	10	1	6	3338/142493
	5	1	15	3	0
$S_3 \to S_1$	4	2	4	7	0
	28	3	6	2	424/142493
$S_3 \to S_2$	9	6	39	1	0
	22	7	7	4	0
$S_3 \to S_3$	27	6	5	22	36889/142493
$S_3 \to S_4$	0	5	3	2	0
	1	4	3	4	0

Continued on next page

Continued from previous page

	Beneficiaries				$F^{\infty\star}$
	b_1	b_2	b_3	b_4	
$S_4 \to S_0$	11	2	2	16	0
	2	1	25	10	0
$S_4 \to S_1$	9	9	11	3	0
	3	19	26	27	47159/142493
$S_4 \to S_2$	13	0	9	7	0
	14	1	15	38	0
$S_4 \to S_3$	1	4	14	3	0
	7	9	16	7	0
$S_4 \to S_4$	0	8	12	18	0

In this problem $U^{\infty\star} = [\frac{2194211}{142493}, \frac{2194211}{142493}, \frac{2194211}{142493}, \frac{2194211}{142493}]$, and the unique $F^{\infty\star}$ induces a strongly connected graph, so $U^{\infty\star}$ is achievable in finite time (that is, every 142493 steps).

B.2 Controller-hierarchy Test Problems

Test-4×4-I Rewards:

		Beneficiaries			
		b_1	b_2	b_3	b_4
Tasks	t_1	12	10	2	2
	t_2	49	56	7	2
	t_3	14	9	12	3
	t_4	13	7	0	20

$U^\star = [\frac{2624}{83}, \frac{2624}{83}, 12, 20]$, and the corresponding fractions of time each beneficiary gets assigned each of the tasks:

		Beneficiaries			
		b_1	b_2	b_3	b_4
Tasks	t_1	39/83	44/83		
	t_2	44/83	39/83		
	t_3			1	
	t_4				1

Test-4×4-II Rewards:

		Beneficiaries			
		b_1	b_2	b_3	b_4
Tasks	t_1	9	1	2	8
	t_2	2	4	7	8
	t_3	1	12	10	6
	t_4	4	11	22	7

$U^\star = [\frac{92}{13}, \frac{92}{13}, \frac{115}{13}, \frac{96}{13}]$, and the corresponding fractions of time each beneficiary gets assigned each of the tasks:

		Beneficiaries			
		b_1	b_2	b_3	b_4
Tasks	t_1	8/13			5/13
	t_2		8/13	5/13	
	t_3		5/13	8/13	
	t_4	5/13			8/13

Test-6×6-I Rewards:

		Beneficiaries					
		b_1	b_2	b_3	b_4	b_5	b_6
Tasks	t_1	0	4	46	3	9	10
	t_2	4	6	0	25	4	4
	t_3	13	3	9	18	5	12
	t_4	19	7	17	5	18	15
	t_5	0	7	4	7	19	4
	t_6	33	4	12	3	3	6

$U^\star = [33, 7, 17, 25, 9, 12]$, and the corresponding fractions of time each beneficiary gets assigned each of the tasks:

		Beneficiaries					
		b_1	b_2	b_3	b_4	b_5	b_6
Tasks	t_1					1	
	t_2				1		
	t_3						1
	t_4			1			
	t_5		1				
	t_6	1					

Test-6×6-II Rewards:

		Beneficiaries					
		b_1	b_2	b_3	b_4	b_5	b_6
Tasks	t_1	11	4	3	5	12	8
	t_2	8	1	5	44	2	9
	t_3	3	8	10	6	14	18
	t_4	14	5	12	12	4	40
	t_5	1	2	20	16	7	14
	t_6	2	17	25	3	14	2

$U^\star = [\frac{59}{7}, \frac{59}{7}, \frac{136}{7}, 44, 13, 13]$, and the corresponding fractions of time each beneficiary gets assigned each of the tasks:

		Beneficiaries					
		b_1	b_2	b_3	b_4	b_5	b_6
Tasks	t_1					1/2	1/2
	t_2				1		
	t_3					1/2	1/2
	t_4	4/7		3/7			
	t_5	3/7	4/7				
	t_6		3/7	4/7			

Test-8×8-I Rewards:

		Beneficiaries							
		b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
Tasks	t_1	1	3	4	1	2	1	3	1
	t_2	3	1	1	3	3	1	4	2
	t_3	4	5	3	2	1	4	3	1
	t_4	2	1	2	1	3	3	2	1
	t_5	4	3	2	4	3	5	2	5
	t_6	5	1	3	1	3	3	2	1
	t_7	4	1	2	1	1	2	1	1
	t_8	3	4	3	4	2	1	4	3

$U^\star = [\frac{15}{4}, \frac{15}{4}, \frac{13}{4}, \frac{15}{4}, \frac{11}{4}, \frac{7}{4}, \frac{11}{4}, \frac{11}{4}]$, and the corresponding fractions of time each beneficiary gets assigned each of the tasks:

		Beneficiaries							
		b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
Tasks	t_1			1/4				3/4	
	t_2				1/4				3/4
	t_3	1/8	1/8				3/4		
	t_4	1/8	1/8			3/4			
	t_5				3/4				1/4
	t_6			3/4				1/4	
	t_7	3/4					1/4		
	t_8		3/4			1/4			

Test-8×8-II Rewards:

		Beneficiaries							
		b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
Tasks	t_1	0	16	36	5	7	1	12	20
	t_2	7	17	2	3	3	20	15	17
	t_3	33	4	22	19	13	16	7	28
	t_4	2	5	5	1	2	1	3	4
	t_5	3	26	3	22	3	0	2	12
	t_6	8	4	10	15	0	14	14	13
	t_7	6	5	20	13	2	1	1	6
	t_8	6	6	22	18	7	39	36	4

$U^\star = [\frac{692}{49}, \frac{1022}{49}, \frac{764}{49}, \frac{726}{49}\frac{505}{49}, \frac{505}{49}, \frac{723}{49}, \frac{884}{49}]$, and the corresponding fractions of time each beneficiary gets assigned each of the tasks:

		Beneficiaries							
		b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
Tasks	t_1				12/49				37/49
	t_2			12/49				37/49	
	t_3	12/49				37/49			
	t_4		12/49				37/49		
	t_5		37/49						12/49
	t_6	37/49						12/49	
	t_7			37/49		12/49			
	t_8				37/49		12/49		

Bibliography

Bibliography

[1] Sherief Abdallah and Victor Lesser. Learning the task allocation game. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 850–857. ACM, 2006.

[2] Sherief Abdallah and Victor Lesser. Multiagent reinforcement learning and self-organization in a network of agents. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8. ACM, 2007.

[3] David Abraham, Ning Chen, Vijay Kumar, and Vahab S. Mirrokni. Assignment problems in rental markets. In *Proceedings of the* 2^{nd} *International Workshop on Internet and Network Economics*, pages 198–213, 2006.

[4] Noa Agmon, Sarit Kraus, and Gal A. Kaminka. Uncertainties in adversarial patrol. In *AAMAS '09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 1267–1268, 2009.

[5] Mazda Ahmadi and Peter Stone. A multi-robot system for continuous area sweeping tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1724–1729, 2006.

[6] Shoshana Anily, Celia A. Glass, and Refael Hassin. The scheduling of maintenance service. *Discrete Applied Mathematics*, 82(1-3):27–42, 1998.

[7] Arash Asadpour and Amin Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 114–121. ACM, 2007.

[8] Benjemin Avi-Itzhak and Hanoch Levy. On measuring fairness in queues. *Advances in Applied Probability*, 36(3):919–936, September 2004.

[9] Gabriel Balan and Sean Luke. History-based traffic control. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi Agent Systems – AAMAS-2006*, pages 616–621. ACM, May 2006.

[10] Gabriel Balan, Dana Richards, and Sean Luke. Long-term fairness with bounded worst-case losses. *Journal of Autonomous Agents and Multi-Agent Systems*, 2009.

[11] Scott A. Banachowski and Scott A. Brandt. Better real-time response for time-share scheduling. In *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing*. IEEE Computer Society, 2003.

[12] Nikhil Bansal and Mor Harchol-Balter. Analysis of srpt scheduling: investigating unfairness. In *SIGMETRICS '01: Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 279–290. ACM Press, 2001.

[13] Amotz Bar-Noy, Vladimir Dreizin, and Boaz Patt-Shamir. Efficient algorithms for periodic scheduling. *Computer Networks*, 45(2):155–173, 2004.

[14] Sanjoy K. Baruah. Scheduling periodic tasks on uniform multiprocessors. In *12th Euromicro Conference on Real-Time Systems*, pages 7–13, 2000.

[15] Sanjoy K. Baruah, N. K. Cohen, C. G. Plaxton, and D.A. Varvel. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica*, 15:600–625, 1994.

[16] Sanjoy K. Baruah, Johannes E. Gehrk, C. Greg Plaxton, Ion Stoica, Hussein Abdel-Wahab, and Kevin Jeffay. Fair on-line scheduling of a dynamic set of tasks on a single resource. *Information Processing Letters*, 64(1):43 – 51, 1997.

[17] Ivona Bezáková and Varsha Dani. Allocating indivisible goods. *SIGecom Exchanges*, 5(3):11–18, 2005.

[18] V. Bhaskar. Egalitarianism and efficiency in repeated symmetric games. *Games and Economic Behavior*, 32(2):247–262, August 2000.

[19] Ella Bingham. Reinforcement learning in neurofuzzy traffic signal control. *European Journal of Operational Research*, 131(2):232–241, July 2001.

[20] François Bonhoure, Yves Dallery, and William J. Stewart. On the use of periodicity properties for the efficient numerical solution of certain Markov chains. *Numerical linear algebra with applications*, 1(3):265–286, 1994.

[21] P. A. Borodin. The Banach-Mazur theorem for spaces with asymmetric norm. *Mathematical Notes*, 69:298–305, 2001.

[22] Walter Bossert and John A. Weymark. Utility in social choice. In *Handbook of Utility Theory*, volume 2, pages 1099–1078. Kluwer Academic Publishers, 2004.

[23] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *Theoretical Aspects of Rationality and Knowledge*, pages 195–201, 1996.

[24] Sylvain Bouveret and Michel Lemaître. Comparison of two constraint programming algorithms for computing leximin-optimal allocations. In *Proceedings of the Workshop on Modelling and Solving Problems with Constraints*, August 2006.

[25] Sylvain Bouveret and Michel Lemaítre. Un algorithme de programmation par contraintes pour la recherche d'allocations leximin-optimales. In *Deuxièmes Journées Francophones de Programmation par Contraintes*, 2006.

[26] Zvika Brakerski, Aviv Nisgav, and Boaz Patt-Shamir. General perfectly periodic scheduling. In *PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 163–172. ACM Press, 2002.

[27] Steven J. Brams. Fair division. In Barry R. Weingast and Donald Wittman, editors, *Oxford Handbook of Political Economy*. Oxford University Press, 2005.

[28] Campbell Brown. Prioritarianism for variable populations. *Philosophical Studies*, 134(3):325–361, 2007.

[29] Edmund Burke, Patrick De Causmaecker, Greet V, En Berghe, and Hendrik Van Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7:441–499, 2004.

[30] Edmund K. Burke, Nottingham Ng Bb, To Nurse Rostering, and Patrick De Causmaecker. A multi criteria meta-heuristic approach to nurse rostering. In *Proceedings of Congress on Evolutionary Computation*, pages 1197–1202. IEEE Press, 2002.

[31] Colin F. Camerer. *Advances in Behavioral Economics (The Roundtable Series in Behavioral Economics)*. Princeton University Press, 2003.

[32] Deeparnab Chakrabarty, Julia Chuzhoy, and Sanjeev Khanna. On allocating goods to maximize fairness. *CoRR*, abs/0901.0205, 2009.

[33] Wei-Mei Chen and Mu-Kai Huang. Efficient perfectly periodic scheduling for data broadcasting. In *International Symposium on Ubiquitous Multimedia Computing*, pages 29–34. IEEE Computer Society, 2008.

[34] Y. Chevaleyre, U. Endriss, S. Estivie, and N. Maudet. Welfare engineering in practice: on the variety of multiagent resource allocation problems. In M.P. Gleizes, Andrea Omicini, and Franco Zambonelli, editors, *Proceedings of the Fifth International Workshop Engineering Societies in the Agent World*, volume 3451 of *Lecture Notes in Artificial Intelligence*, pages 335–347. Springer-Verlag, 2004.

[35] Yann Chevaleyre, Paul E. Dunne, Ulle Endriss, Jérôme Lang, Michel Lemaître, Nicolas Maudet, Julian Padget, Steve Phelps, Juan A. Rodríguez-Aguilar, and Paulo Sousa. Issues in multiagent resource allocation. *Informatica*, 30:3–31, 2006.

[36] Yann Chevaleyre, Francois Sempe, and Geber Ramalho. A theoretical analysis of multi-agent patrolling strategies. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1524–1525, Washington, DC, USA, 2004. IEEE Computer Society.

[37] Hillermeier Claus. *Nonlinear Multiobjective Optimization: A Generalized Homotopy Approach*, volume 135 of *ISNM*. Birkhauser, 2001.

[38] Milind W. Dawande, H. Neil Geismar, and Suresh P. Sethi. Dominance of cyclic solutions and challenges in the scheduling of robotic cells. *SIAM Review*, 47(4):709–721, 2005.

[39] Kenneth A. De Jong. *Evolutionary Computation: A Unified Approach*. MIT Press, Cambridge, MA, 2006.

[40] Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons, Chichester, 2001.

[41] Jean Derks and Hans Peters. Orderings, excess functions, and the nucleolus. *Mathematical Social Sciences*, 36:175–182, 1998.

[42] Luc Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.

[43] Thomas G. Dietterich. The maxq method for hierarchical reinforcement learning. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 118–126, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[44] Jim Dowling, Raymond Cunningham, Anthony Harrington, Eoin Curran, and Vinny Cahill. Emergent consensus in decentralised systems using collaborative reinforcement learning. *LNCS 3460, Self-Star Properties in Complex Information Systems*, pages 63–80, 2005.

[45] Bart Du Laing. Equality in exchange revisited - from an evolutionary (genetic and cultural) point of view. In Michael Freeman and Oliver R. Goodenough, editors, *Law, mind and brain*, pages 267–298. Ashgate, 2009.

[46] Didier Dubois and Philippe Fortemps. Computing improved optimal solutions to max-min flexible constraint satisfaction problems. *European Journal of Operational Research*, 118(1):95–126, 1999.

[47] Didier Dubois, Philippe Fortemps, Marc Pirlot, and Henri Prade. Leximin optimality and fuzzy set-theoretic operations. *European Journal of Operational Research*, 130(1):20–28, 2001.

[48] Gabriele Eichfelder. *Adaptive Scalarization Methods in Multiobjective Optimization*. Springer-Verlag, 2008.

[49] Yehuda Elmaliach, Noa Agmon, and Gal A. Kaminka. Multi-robot area patrol under frequency constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 385–390, 2007.

[50] Ulle Endriss, Nicolas Maudet, Fariba Sadri, and Francesca Toni. Negotiating socially optimal allocations of resources: An overview. *Journal of Artificial Intelligence Research*, 25:315–348, 2006.

[51] Ulrich Endriss and Nicolas Maudet. Welfare engineering for multiagent systems. In A. Omicini, P. Petta, and J. Pitt, editors, *Proceedings of the 4th International Workshop Engineering Societies in the Agent World (ESAW-2003)*, volume 3071 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 93–106. Springer-Verlag, 2004.

[52] D. H. J. Epema. An analysis of decay-usage scheduling in multiprocessors. In *SIGMETRICS '95/PERFORMANCE '95: Proceedings of the 1995 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 74–85. ACM Press, 1995.

[53] D. H. J. Epema and J. F. C. M. de Jongh. Proportional-share scheduling in single-server and multiple-server computing systems. *SIGMETRICS Performance Evaluation Review*, 27(3):7–10, 1999.

[54] Jakob Eriksson, Hari Balakrishnan, and Samuel Madden. Cabernet: Vehicular Content Delivery Using WiFi. In *14th ACM MOBICOM*, San Francisco, CA, 2008.

[55] Ronald Fagin and John H. Williams. A fair carpool scheduling algorithm. *IBM Journal of Research and Development*, 27(2):133–139, 1983.

[56] Steven R. Finch. *Mathematical Constants*. Cambridge University Press, 2003.

[57] Felix Fischer, Michael Rovatsos, and Gerhard Weiss. Hierarchical reinforcement learning in communication-mediated multiagent coordination. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1334–1335. IEEE Computer Society, 2004.

[58] Marc Fleurbaey. Social welfare, priority to the worst-off and the dimensions of individual well-being. IDEP Working Papers 0312, Institut d'economie publique (IDEP), Marseille, France, 2003.

[59] Marc Fleurbaey and Francois Maniquet. Fair allocation with unequal production skills: The no envy approach to compensation. *Mathematical Social Sciences*, 32(1):71–93, August 1996.

[60] Monique Florenzano and Cuong Le Van. *Finite Dimensional Convexity and Optimization*. Springer, 2001.

[61] Philippe Fortemps and Marc Pirlot. Conjoint axiomatization of Min, DiscriMin and LexiMin. *Fuzzy Sets and Systems*, 148(2):211–229, 2004.

[62] Dean P. Foster and Rakesh V. Vohra. Calibrated learning and correlated equilibria. *Games and Economic Behavior*, 21:40–55, 1995.

[63] Eric J. Friedman and Shane G. Henderson. Fairness and efficiency in web server protocols. In *SIGMETRICS '03: Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 229–237. ACM Press, 2003.

[64] Thibault Gajdos. Measuring inequalities without linearity in envy through choquet integral with symmetric capacities. Technical Report halshs-00085888-v1, HAL, CCSD, July 2006.

[65] Haichang Gao and Weizhou Zhong. Multiobjective optimization using clustering based two phase pso. *International Conference on Natural Computation*, 6:520–524, 2008.

[66] Michael R. Garey and David S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman, January 1979.

[67] Mitsuo Gen, Runwei Cheng, and Lin Lin. *Network Models and Optimization: Multiobjective Genetic Algorithm Approach*. Springer, 2008.

[68] Daniel Golovin. Max-min fair allocation of indivisible goods. Technical Report CMU-CS-05-144, School of Computer Science, Carnegie Mellon University, 2005.

[69] Stephen C. Graves. A review of production scheduling. *Operational Research*, 29(4):646–675, 1981.

[70] Amy Greenwald and Keith Hall. Correlated-Q learning. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 242–249, 2003.

[71] Zhu Han and H. Vincent Poor. Lifetime improvement in wireless sensor networks via collaborative beamforming and cooperative transmission. *IET Microwaves, Antennas and Propagation*, 1:1103–1110, 2007.

[72] Mor Harchol-Balter, Bianca Schroeder, Nikhil Bansal, and Mukesh Agrawal. Size-based scheduling to improve web performance. *ACM Transactions on Computer Systems (TOCS)*, 21(2):207–233, 2003.

[73] Dzung T. Hoang, Elliot Linzer, and Jeffrey Scott Vitter. Lexicographic bit allocation for MPEG video. *Journal of Visual Communication and Image Representation*, 8:384–404, 1997. Special issue on high-fidelity media processing.

[74] Leslie Hogben, editor. *Handbook of Linear Algebra (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, November 2006.

[75] Rajanra K. Jain, Dah-Ming W. Chiu, and William R. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer system. Technical Report DEC-TR-301, Digital Equipment Corporation, September 1984.

[76] Kevin Jeffay, F. Donelson Smith, A. Moorthy, and James Anderson. Proportional share scheduling of operating system services for real-time applications. In *RTSS '98: Proceedings of the IEEE Real-Time Systems Symposium*, page 480. IEEE Computer Society, 1998.

[77] Albert Jones and Luis C. Rabelo. Survey of job shop scheduling techniques. Technical report, NISTIR, National Institute of Standards and Technology, 1998.

[78] Geert Jonker, John-Jules Ch. Meyer, and Frank Dignum. Efficiency and fairness in air traffic control. In *BNAIC - Proceedings of the Seventeenth Belgium-Netherlands Conference on Artificial Intelligence*, pages 151–157, October 2005.

[79] Spiros Kapetanakis and Daniel Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *Eighteenth national conference on Artificial intelligence*, pages 326–331, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[80] Spiros Kapetanakis, Daniel Kudenko, and Malcom Strens. Learning to coordinate using commitment sequences in cooperative multi-agent systems. *The Interdisciplinary Journal of Artificial Intelligence and the Simulation of Behaviour*, 1(4), 2003.

[81] Maria Kihl, Mihail Sichitiu, Ted Ekeroth, and Michael Rozenberg. Reliable geographical multicast routing in vehicular ad-hoc networks. In *WWIC '07: Proceedings of the 5th international conference on Wired/Wireless Internet Communications*, pages 315–325, Berlin, Heidelberg, 2007. Springer-Verlag.

[82] Rachelle S. Klein, Hanan Luss, and Donald R. Smith. A lexicographic minimax algorithm for multiperiod resource allocation. *Mathematical Programming*, 55(2):213–234, 1992.

[83] Jelle R. Kok. *Coordination and Learning in Cooperative Multiagent Systems*. PhD thesis, Faculty of Science, University of Amsterdam, 2006.

[84] Michael M. Kostreva, Wlodzimierz Ogryczak, and Adam Wierzbicki. Equitable aggregations and multiple criteria analysis. *European Journal of Operational Research*, 158(2):362–377, October 2004.

[85] Sau-Him Paul Lau and Vai-Lam Mui. Achieving intertemporal efficiency and symmetry through intratemporal asymmetry: (eventual) turn taking in a class of repeated mixed-interest games. Technical Report 1092, Hong Kong Institute of Economics and Business Strategy, November 2003.

[86] Sau-Him Paul Lau and Vai-Lam Mui. Using turn taking to mitigate conflict and coordination problems in the battle of the sexes game. Technical Report 1129, Hong Kong Institute of Economics and Business Strategy, April 2005.

[87] Sau-Him Paul Lau and Vai-Lam Mui. Using turn taking to mitigate coordination and conflict problems in the repeated battle of the sexes game. *Theory and Decision*, 65(2):153–183, 2008.

[88] Yahia Lebbah, Michel Rueher, and Claude Michel. A global filtering algorithm for handling systems of quadratic equations and inequations. In *CP '02: Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, pages 109–123, London, UK, 2002. Springer-Verlag.

[89] Duan-Shin Lee, Chun-Min Chen, and Chih-Yuan Tang. Weighted fair queueing and compensation techniques for wireless packet switched networks. *IEEE Transactions on Vehicular Technology*, 56(1):297–311, 2007.

[90] M. Lemaître, G. Verfaillie, H. Fargier, J. Lang, N. Bataille, and J.-M. Lachiver. Equitable allocation of earth observing satellites resources. In *Proceeding of ONERA-DLR Aerospace Symposium ODAS'03*, 2003.

[91] Seymour Lipschutz and Marc Lipson. *Schaum's Outline of Theory and Problems of Linear Algebra*. McGraw-Hill, Inc., New York, NY, USA, third edition, 2001.

[92] R. J. Lipton, E. Markakis, E. Mossel, and A. Saberi. On approximately fair allocations of indivisible goods. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 125–131. ACM, 2004.

[93] Ami Litman and Shiri Moran-Schein. On distributed smooth scheduling. In *SPAA '05: Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 76–85, New York, NY, USA, 2005. ACM.

[94] Ami Litman and Shiri Moran-Schein. On distributed smooth scheduling. In *SPAA '05: Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 76–85. ACM, 2005.

[95] Ami Litman and Shiri Moran-Schein. On distributed smooth scheduling. Technical Report CS-2005-04, Technion - Israel Institute of Technology, 2005.

[96] Henry X. Liu, Jun-Seok Oh, and Will Recker. Adaptive signal control system with on-line performance measure. *Transportation Research Record*, 1811:131–138, 2002.

[97] Bonifacio Llamazares. Choosing owa operator weights in the field of social choice. *Information Sciences*, 177(21):4745–4756, 2007.

[98] Daphne Lopez and S. V. Raja. Virtual time fair queuing algorithm for a computational grid. In *ICDCN '09: Proceedings of the 10th International Conference on Distributed Computing and Networking*, pages 468–474, Berlin, Heidelberg, 2009. Springer-Verlag.

[99] Sean Luke. ECJ 19: A Java EC research system. Available at http://cs.gmu.edu/∼eclab/projects/ecj/, 2009.

[100] Ying Ma, Zhiqiang Cao, Chao Zhou, and Min Tan. A hierarchical task allocation method based on task case and its application to multi-robot hunting. *Intelligent Networks and Intelligent Systems, International Workshop on*, 0:689–692, 2008.

[101] Marvin B. Mandell. Modelling effectiveness-equity trade-offs in public service delivery systems. *Management Science*, 37(4):467–482, 1991.

[102] David McCarthy. Utilitarianism and prioritarianism I. *Economics and Philosophy*, 22:335–363, 2006.

[103] David McCarthy. Utilitarianism and prioritarianism II. *Economics and Philosophy*, 24:1–33, 2008.

[104] Neville Mehta, Soumya Ray, Prasad Tadepalli, and Thomas G. Dietterich. Automatic discovery and transfer of maxq hierarchies. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *ICML*, volume 307 of *ACM International Conference Proceeding Series*, pages 648–655. ACM, 2008.

[105] Harvey H. Millar and Mona Kiragu. Cyclic and non-cyclic scheduling of 12 h shift nurses by network programming. *European Journal of Operational Research*, 104(3):582–592, 1998.

[106] David J. Montana and Steven Czerwinski. Evolving control laws for a network of traffic signals. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 333–338, Stanford University, CA, USA, 1996. MIT Press.

[107] Paul Morris, Robert Morris, Lina Khatib, Sailesh Ramakrishnan, and Andrew Bachmann. Strategies for global optimization of temporal preferences. In *Principles and Practice of Constraint Programming*, volume 3258 of *Lecture Notes in Computer Science*, pages 408–422. Springer, 2004.

[108] Hervé Moulin. *Fair Division and Collective Welfare*. Cambridge University Press, 1998.

[109] Hervé Moulin. *Fair Division and Collective Welfare*. The MIT Press, 2003.

[110] Dritan Nace and James B. Orlin. Lexicographically minimum and maximum load linear programming problems. *Operational Research*, 55(1):182–187, 2007.

[111] Moni Naor. On fairness in the carpool problem. *Journal of Algorithms*, 55(1):93–98, 2005.

[112] Daniel B. Neill. Cooperation and coordination in the turn-taking dilemma. In *TARK '03: Proceedings of the 9th conference on Theoretical aspects of rationality and knowledge*, pages 231–244, New York, NY, USA, 2003. ACM Press.

[113] Travis Newhouse and Joseph Pasquale. Alps: An application-level proportional-share scheduler. In *Proceedings of the International Symposium on High Performance Distributed Computing*, June 2006.

[114] B. J. Nicholson. On the f-distribution for calculating bayes credible intervals for fraction nonconforming. *Reliability, IEEE Transactions on*, R-34(3):227–228, 1985.

[115] Jason Nieh, Christopher Vaill, and Hua Zhong. Virtual-time round-robin: An o(1) proportional share scheduler. In *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, pages 245–259, Berkeley, CA, USA, 2001. USENIX Association.

[116] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[117] Wlodzimierz Ogryczak. On the lexicographic minimax approach to location problems. *European Journal of Operational Research*, 100:566–585, 1997.

[118] Wlodzimierz Ogryczak, Marcin Milewski, and Adam Wierzbicki. On fair and efficient bandwidth allocation by the multiple target approach. In *NGI 2006, 2nd Conference on Next Generation Internet Design and Engineering*, pages 48–55. IEEE, 2006.

[119] Wlodzimierz Ogryczak, Michal Pióro, and Artur Tomaszewski. Telecommunications network design and max-min optimization problem. *Journal of Telecommunications and Information Technology*, 3, 2005.

[120] Wlodzimierz Ogryczak and Tomasz Śliwiński. On solving linear programs with the ordered weighted averaging objective. *European Journal of Operational Research*, 148:80–91, 2003.

[121] Wlodzimierz Ogryczak and Tomasz Śliwiński. Lexicographic max-min optimization for efficient and fair bandwidth allocation. In *International Network Optimization Conference (INOC)*, 2007.

[122] Erwin Ooghe. Reasonable agreement implies universal leximin. In *Conference "New Directions in Inequality Analysis"*, 2000.

[123] Erwin Ooghe. A consensus leximin rule. Technical Report 2001-09, THEMA (THéorie Economique, Modélisation et Applications), Université de Cergy-Pontoise, 2001. available at http://ideas.repec.org/p/ema/worpap/2001-09.html.

[124] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[125] Liviu Panait. *The Analysis and Design of Concurrent Learning Algorithms for Cooperative Multiagent Systems*. PhD thesis, George Mason University, Fairfax, VA, 2006.

[126] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[127] Liviu Panait, Keith Sullivan, and Sean Luke. Lenience towards teammates helps in cooperative multiagent learning. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi Agent Systems – AAMAS-2006*. ACM, 2006.

[128] Christos H. Papadimitriou. Computing correlated equilibria in multi-player games. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 49–56, New York, NY, USA, 2005. ACM Press.

[129] Shailesh Patil and Vijay K. Garg. Adaptive general perfectly periodic scheduling. *Information Processing Letters*, 98(3):107–114, 2006.

[130] Maarten Peeters, Ville Könönen, Katja Verbeeck, Sven Segbroeck, and Ann Nowé. Coordinated exploration in conflicting multi-stage games. In *KES '08: Proceedings of the 12th international conference on Knowledge-Based Intelligent Information and Engineering Systems, Part II*, pages 391–402, Berlin, Heidelberg, 2008. Springer-Verlag.

[131] Maarten Peeters, Katja Verbeeck, and Ann Nowé. Multi-agent learning in conflicting multi-level games with incomplete information. In *Proc. of the AAAI 2004 Fall Symposium Series on Artificial Multiagent Learning*, 2004.

[132] Maarten Peeters, Katja Verbeeck, and Ann Nowé. Solving multi-stage games with hierarchical learning automata that bootstrap. In Karl Tuyls, Ann Nowé, Zahia Guessoum, and Daniel Kudenko, editors, *Adaptive Agents and Multi-Agents Systems*, volume 4865 of *Lecture Notes in Computer Science*, pages 169–187. Springer, 2007.

[133] Patrice Perny and Olivier Spanjaard. An axiomatic approach to robustness in search problems with multiple scenarios. In *Proceedings of the 19th conference on Uncertainty in Artificial Intelligence*, pages 469–476, 2003.

[134] Patrice Perny and Olivier Spanjaard. Near admissible algorithms for multiobjective search. In *18th European Conference on Artificial Intelligence ECAI-08*, pages 490–494. IOS Press, 2008.

[135] Martin Peterson and Sven Ove Hansson. Equality and priority. *Utilitas*, 17:299–309, 2005.

[136] Robert R. Phelps. *Lectures on Choquet's Theorem*. Springer, 2001.

[137] Marc Ponsen, Pieter Spronck, and Karl Tuyls. Hierarchical reinforcement learning with deictic representation in a computer game. In *Proceedings of the 18th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2006)*, pages 251–258, 2006.

[138] Jos A. M. Potters and Stef H. Tijs. The nucleolus of a matrix game and other nucleoli. *Mathematics of Operations Research*, 17(1):164–174, 1992.

[139] Matthew Rabin. Fairness in repeated games. Economics Working Papers 97-252, University of California at Berkeley, 1997.

[140] Roy Radner. The organization of decentralized information processing. *Econometrica*, 61(5):1109–1146, 1993.

[141] Božidar Radunović and Jean-Yves Le Boudec. A unified framework for max-min and min-max fairness with applications. *IEEE/ACM Transactions on Networking*, 15(5):1073–1083, 2007.

[142] Srikanth Ramamurthy and Mark Moir. Static-priority periodic scheduling on multiprocessors. In *Real-Time Systems Symposium*, pages 69–78, 2000.

[143] Eric Rasmusen. *Games and Information*. Blackwell, second edition, 1994.

[144] John Rawls. *A Theory of Justice*. Universal Law Publishing Co Ltd, 2005.

[145] Maxim Raya and Jean-Pierre Hubaux. The security of vehicular ad hoc networks. In *SASN '05: Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*, pages 11–21, New York, NY, USA, 2005. ACM Press.

[146] David Raz, Hanoch Levy, and Benjamin Avi-Itzhak. A resource-allocation queueing fairness measure. In *SIGMETRICS '04/Performance '04: Proceedings of the joint international conference on Measurement and modeling of computer systems*, pages 130–141. ACM Press, 2004.

[147] Kevin W. S. Roberts. Interpersonal comparability and social choice theory. *Review of Economic Studies*, 47(2):421–39, January 1980.

[148] Jack Robertson and William Webb. *Cake-Cutting Algorithms: Be Fair If You Can*. A. K. Peters, Ltd., 1998.

[149] Jean-Paul Rodrigue, Claude Comtois, and Brian Slack. *The Geography of Transport Systems*. Routledge, 2006.

[150] John E. Roemer. Eclectic distributional ethics. Yale School of Management Working Papers ysm348, Yale School of Management, July 2004.

[151] Gerald Sabin, Garima Kochhar, and P. Sadayappan. Job fairness in non-preemptive job scheduling. In *Proceedings of the 2004 International Conference on Parallel Processing (ICPP'04)*, pages 186–194, Washington, DC, USA, 2004. IEEE Computer Society.

[152] Werner Sandmann. A discrimination frequency based queueing fairness measure with regard to job seniority and service requirement. In R. Sabella, editor, *Proceedings of the 1st Euro NGI Conference on Next Generation Internet Networks - Traffic Engineering (NGI2005)*. IEEE Computer Society Press, April 2005.

[153] Werner Sandmann. Scheduling to improve queue justice. In *Proceedings of the 20th European Conference on Modelling and Simulation (ECMS 2006)*, pages 372–377, 2006.

[154] H. H. Schaefer. *Topological Vector Spaces*. Number 3 in Graduate Texts in Mathematics. Springer-Verlag, second edition, 1999.

[155] Rudolf Schneider. On the chairman assignment problem. *Discrete Mathematics*, 159(1):217–222, 1996.

[156] Bianca Schroeder and Mor Harchol-Balter. Evaluation of task assignment policies for supercomputing servers: The case for load unbalancing and fairness. In *Proceedings of the Ninth IEEE International Symposium on High Performance Distributed Computing (HPDC)*. IEEE Press, August 2000.

[157] Erik Schultink, Ruggiero Cavallo, and David C. Parkes. Economic hierarchical Q-learning. In *Proc. 23rd AAAI Conference on Artificial Intelligence (AAAI'08)*, pages 689–695, 2008.

[158] David W. Sellers. A survey of approaches to the job shop scheduling problem. In *Proceedings of the 28th Southeastern Symposium on System Theory (SSST '96)*, pages 396–400. IEEE Computer Society, 1996.

[159] Sergey Sevast'janov. On the compact summation of vectors. *Diskretnaya Matematika*, 3(3):66–72, 1991. In Russian.

[160] Sergey Sevast'janov and Wojciech Banaszczyk. To the Steinitz lemma in coordinate form. *Discrete Mathematics*, 169(1):145–152, 1997.

[161] Sylvain Sorin. On repeated games with complete information. *Mathematics of Operations Research*, 11(1):147–160, 1986.

[162] Tamon Stephen, Levent Tuncel, and Hanan Luss. On equitable resource allocation problems: a lexicographic minimax approach. *Oper. Res.*, 47(3):361–376, 1999.

[163] Tamon Stephen, Levent Tuncel, and Hanan Luss. On equitable resource allocation problems: a lexicographic minimax approach. *Operations Research*, 47(3):361–376, 1999.

[164] David S. G. Stirling. *Mathematical Analysis and Proof*. Albion Publishing, 1997.

[165] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[166] Poj Tangamchit, John Dolan, and Pradeep Khosla. Learning-based task allocation in decentralized multirobot systems. In *Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems (DARS 2000)*, pages 381–390, 2000.

[167] Larry S. Temkin. Equality, priority or what? *Economics and Philosophy*, 19:61–87, April 2003.

[168] Chen-Khong Tham, Rui-An Lou, and Hoon-Tong Ngin. Minimum queuing delay: an adaptive algorithm for scheduling in a dynamic job shop. In *Procedings of 25th Conference of IEEE Industrial Electronics Society (IECON'99)*, volume 2, pages 962–067, 1999.

[169] Robert Tijdeman. The chairman assignment problem. *Discrete Mathematics*, 32(3):323–330, 1980.

[170] Walter Trockel. On the meaning of the nash product. Working Papers 354, Bielefeld University, Institute of Mathematical Economics, 2004.

[171] John N. Tsitsiklis and Benjamin Van Roy. On average versus discounted reward temporal-difference learning. *Machine Learning*, 49(2-3):179–191, 2002.

[172] Bertil Tungodden. Egalitarianism: Is leximin the only option? *Economics and Philosophy*, 16:229–245, 2000.

[173] R.J.M. Vaessens, E.H.L. Aarts, and J.K. Lenstra. Job shop scheduling by local search. *INFORMS Journal on Computing*, 8(3):302–317, 1996.

[174] Timothy Van Zandt. Hierarchical computation of the resource allocation problem. *European Economic Review*, 39(3–4):700–708, April 1995.

[175] Timothy Van Zandt. Real-time decentralized information processing as a model of organizations with boundedly rational agents. *Review of Economic Studies*, 66(3):633–58, 1999.

[176] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2004.

[177] Katja Verbeeck, Ann Nowé, Johan Parent, and Karl Tuyls. Exploring selfish reinforcement learning in repeated games with stochastic rewards. *Journal of Autonomous Agents and Multi-Agent Systems*, 14(3):239–269, 2006.

[178] Katja Verbeeck, Johan Parent, and Ann Nowé. Homo egualis reinforcement learning agents for load balancing. In *Innovative Concepts for Agent-Based Systems: 1st International Workshop on Radical Agent Concepts*, volume 2564 of *Lecture Notes in Computer Science*, pages 81–91, January 2003.

[179] Wim F. J. Verhaegh, Emile H. L. Aarts, and Paul C. N. van Gorp. Period assignment in multidimensional periodic scheduling. In *ICCAD '98: Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design*, pages 585–592. ACM Press, 1998.

[180] Le Yi Wang and Loren Schwiebert. Robust control and rate coordination for efficiency and fairness in ABR traffic with explicit rate marking. In *Proceedings of the 2000 American Control Conference*, pages 1975–1979, June 2000.

[181] Karl Widerquist. Who exploits who? *Political Studies*, 54:444–464, 2006.

[182] Adam Wierman. Fairness and classifications. *SIGMETRICS Performance Evaluation Review*, 34(4):4–12, 2007.

[183] Adam Wierman and Mor Harchol-Balter. Classifying scheduling policies with respect to unfairness in an M/GI/1. In *SIGMETRICS '03: Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 238–249. ACM Press, 2003.

[184] Mark Wineberg and Steffen Christensen. An introduction to statistical analysis for evolutionary computation. In *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, pages 2639–2664. ACM, 2008.

[185] Ronald R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):183–190, 1988.

[186] Ronald R. Yager. On the analytic representation of the Leximin ordering and its application to flexible constraint propagation. *European Journal of Operational Research*, 102:176–192, 1997.

[187] Amel Yahyaoui and Farhat Fnaiech. Recent trends in intelligent job shop scheduling. In *Procedings of the 1st IEEE International Conference on E-Learning in Industrial Electronics*, pages 191–195, 2006.

[188] David K. Y. Yau. ARC-H: Uniform CPU scheduling for heterogeneous services. In *ICMCS '99: Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, page 127. IEEE Computer Society, 1999.

[189] H. Peyton Young. *Individual Strategy and Social Structure: An Evolutionary Theory of Institutions*. Princeton University Press, 1998.

[190] Timothy Van Zandt. Real-time hierarchical resource allocation. Discussion Papers 1231, Northwestern University, Center for Mathematical Studies in Economics and Management Science, 1997.

[191] Timothy Van Zandt. The scheduling and organization of periodic associative computation: Essential networks. *Review of Economic Design*, 3(1):15–27, 1997.

[192] Timothy Van Zandt. The scheduling and organization of periodic associative computation: Efficient networks. *Review of Economic Design*, 3(2):93–127, 1998.

[193] Chongjie Zhang, Sherief Abdallah, and Victor Lesser. Efficient multi-agent reinforcement learning through automated supervision. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 1365–1370. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[194] Yunkai Zhou. *Resource Allocation in Computer Networks: Fundamental Principles and Practical Strategies*. PhD thesis, Drexel University, 2003.

[195] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. TIK Report 103, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Zurich, Switzerland, 2001.

[196] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[197] Moshe Zukerman, Liansheng Tan, Hanwu Wang, and Iradj Ouveysi. Efficiency-fairness tradeoff in telecommunications networks. *IEEE Communications Letters*, 9(7):643–645, July 2005.

Curriculum Vitae

Gabriel Catalin Balan was born in Ploiesti, Romania on October 15^{th}, 1976. He received a Bachelor of Science degree in Computer Science from Bucharest Polytechnic University in 2000, and a Doctor of Philosophy degree from George Mason University in 2009. His research interests cover machine learning (evolutionary computation, associative memories) and multi-agent systems (coordination, cooperation, fairness, and favors).

Gabriel Balan co-authored fifteen refereed paper on machine learning and multi-agent systems in journals and conference proceedings. His professional service includes reviewing workshop, conference, and journal papers in these fields. Gabriel Balan was a major coauthor of two open-source projects: the ECJ evolutionary computation research library, and the MASON discrete-event simulation toolkit.