PLANNING AND SCHEDULING FACILITY WORKLOAD

by

Ryan R. Squires
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Systems Engineering and Operations Research

Committee:

______________________________  Dr. Karla L. Hoffman, Dissertation Director
______________________________  Dr. Daniel B. Carr, Committee Member
______________________________  Dr. Rajesh Ganesan, Committee Member
______________________________  Dr. Andrew G. Loerch, Committee Member
______________________________  Dr. Ariela Sofe, Department Chair
______________________________  Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering

Date: __________________________ Summer Semester 2014
George Mason University
Fairfax, VA
Planning and Scheduling Facility Workload

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

by

Ryan R. Squires
Master of Science
Naval Postgraduate School, 2007
Master of Science
Missouri University of Science and Technology, 1999
Bachelor of Science
United States Military Academy, 1995

Director: Karla L. Hoffman, Professor
Department of Systems Engineering and Operations Research

Summer Semester 2014
George Mason University
Fairfax, VA
DEDICATION

To Jennifer, Norah and Quinlan.
ACKNOWLEDGEMENTS

I have been very fortunate to have an outstanding advisor and committee for this work. Professor Karla Hoffman has been an exceptional teacher and advisor. I was fortunate to obtain her advice and guidance. Had I been listening more carefully, I might have finished this research much sooner. Professor Andy Loerch has been a reliable source of thoughtful advice pertaining to the academic program at George Mason University and the Army. Professor Daniel Carr and Professor Rajesh Ganesan brought depth of knowledge and enthusiasm to important subjects that should benefit me for the rest of my career. I am deeply grateful to all of my committee members for their instruction and guidance.

I also wish to express gratitude to Clarke Fox, Meyer Kotkin and Kirk Benson for an introduction to my research topic. It has been a fruitful and interesting avenue.

Finally, I am also deeply indebted to those who have helped me along the paths of life, academics and career since that was also necessary in order to arrive at this particular destination. A list of those who have been instrumental in this regard would begin with my parents, Kathy and Jerry Squires. Others to whom I owe a great debt of gratitude include, but are certainly not limited to, Richard Prince, Gerald Davie, Steven Stoddard, Christopher Hill and Matthew Carlyle.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>vii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>viii</td>
</tr>
<tr>
<td>List of Abbreviations</td>
<td>ix</td>
</tr>
<tr>
<td>Abstract</td>
<td>x</td>
</tr>
<tr>
<td>I-Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Army Maintenance Operations</td>
<td></td>
</tr>
<tr>
<td>1.2 Sustainment Level Maintenance Operations</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Intermediate Maintenance Facility Workload</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Problem Statement</td>
<td>5</td>
</tr>
<tr>
<td>1.5 Objective of this Research</td>
<td>8</td>
</tr>
<tr>
<td>1.6 Research Approach</td>
<td>8</td>
</tr>
<tr>
<td>1.7 Research Contributions</td>
<td>11</td>
</tr>
<tr>
<td>II-Literature Review</td>
<td>14</td>
</tr>
<tr>
<td>2.1 Multi-Objective Optimization</td>
<td>14</td>
</tr>
<tr>
<td>2.2 Assignment Models</td>
<td>17</td>
</tr>
<tr>
<td>2.3 Scheduling Models</td>
<td>18</td>
</tr>
<tr>
<td>2.4 Sliding Windows Formulation Model</td>
<td>23</td>
</tr>
<tr>
<td>2.5 Logic Based Benders’ Decomposition Approach</td>
<td>28</td>
</tr>
<tr>
<td>2.6 Dantzig-Wolfe Reformulation</td>
<td>34</td>
</tr>
<tr>
<td>2.7 Integer Programming Column Generation</td>
<td>37</td>
</tr>
<tr>
<td>2.8 Sifting</td>
<td>39</td>
</tr>
<tr>
<td>2.9 Relationship to the Literature</td>
<td>41</td>
</tr>
<tr>
<td>III-Model Formulations</td>
<td>43</td>
</tr>
<tr>
<td>3.1 Standard Formulations</td>
<td>45</td>
</tr>
<tr>
<td>3.1.1 Full Standard Formulation (SF)</td>
<td>46</td>
</tr>
<tr>
<td>3.1.2 Restricted Standard Formulation</td>
<td>50</td>
</tr>
<tr>
<td>Section Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------------</td>
<td>------</td>
</tr>
<tr>
<td>3.2 Benders’ Decomposition</td>
<td>51</td>
</tr>
<tr>
<td>3.2.1 An Initial Benders’ Master Problem</td>
<td>51</td>
</tr>
<tr>
<td>3.2.2 The Benders’ Facility Sub-Problem</td>
<td>54</td>
</tr>
<tr>
<td>3.2.3 Benders’ Cuts</td>
<td>57</td>
</tr>
<tr>
<td>3.2.4 Final Benders’ Master Problem</td>
<td>59</td>
</tr>
<tr>
<td>3.2.5 Benders’ Algorithm Upper Bounds</td>
<td>63</td>
</tr>
<tr>
<td>3.2.6 A Hybrid Benders’ Decomposition Method</td>
<td>64</td>
</tr>
<tr>
<td>3.3 Column Generation Formulation</td>
<td>65</td>
</tr>
<tr>
<td>3.3.1 Column Generation Master Problem (RMP)</td>
<td>67</td>
</tr>
<tr>
<td>3.3.2 Column Generation Pricing Problem</td>
<td>68</td>
</tr>
<tr>
<td>3.4 Sifting Formulation</td>
<td>72</td>
</tr>
<tr>
<td>IV-Experimental Results</td>
<td>76</td>
</tr>
<tr>
<td>4.1 Sliding Windows Formulation</td>
<td>77</td>
</tr>
<tr>
<td>4.2 Standard Formulation Results</td>
<td>79</td>
</tr>
<tr>
<td>4.3 Benders Decomposition Formulation (BDF) Results</td>
<td>80</td>
</tr>
<tr>
<td>4.4 Column Generation Formulation (CG) Results</td>
<td>82</td>
</tr>
<tr>
<td>4.5 Intermediate Comparison of Methods</td>
<td>84</td>
</tr>
<tr>
<td>4.6 Restricted Standard Formulation (RSF) bounded by Benders Decomposition Formulation (BDF) Results</td>
<td>86</td>
</tr>
<tr>
<td>4.7 Results: Sifting Formulation</td>
<td>87</td>
</tr>
<tr>
<td>4.7 Summary of Results</td>
<td>89</td>
</tr>
<tr>
<td>4.8 Tardiness and Cost Trade-Off</td>
<td>94</td>
</tr>
<tr>
<td>V-Insights, Conclusions and Further Research</td>
<td>99</td>
</tr>
<tr>
<td>5.1 Insights</td>
<td>100</td>
</tr>
<tr>
<td>5.2 Conclusions</td>
<td>101</td>
</tr>
<tr>
<td>5.3 Future Work</td>
<td>102</td>
</tr>
<tr>
<td>References</td>
<td>104</td>
</tr>
</tbody>
</table>
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1: Equipment Groups</td>
<td>9</td>
</tr>
<tr>
<td>Table 2: SWF Results Reported in (Kotkin et.al 2011)</td>
<td>10</td>
</tr>
<tr>
<td>Table 3: SWF Results</td>
<td>77</td>
</tr>
<tr>
<td>Table 4: Full Standard Formulation Results</td>
<td>80</td>
</tr>
<tr>
<td>Table 5: Benders' Decomposition Results</td>
<td>81</td>
</tr>
<tr>
<td>Table 6: SWF Results Bounded by Benders' Decomposition Results</td>
<td>82</td>
</tr>
<tr>
<td>Table 7: Column Generation Results</td>
<td>84</td>
</tr>
<tr>
<td>Table 8: Restricted Formulation Bounded by Benders' Decomposition Results</td>
<td>87</td>
</tr>
<tr>
<td>Table 9: Sifting Formulation Results</td>
<td>89</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1: Solution Method Lower Bounds by Instance</td>
<td>85</td>
</tr>
<tr>
<td>Figure 2: Upper Bound Found by Solution Method</td>
<td>90</td>
</tr>
<tr>
<td>Figure 3: Lower Bound Proven by Solution Method</td>
<td>91</td>
</tr>
<tr>
<td>Figure 4: Relative Gap by Solution Method</td>
<td>93</td>
</tr>
<tr>
<td>Figure 5: Equipment Category 3 Pareto Chart</td>
<td>95</td>
</tr>
<tr>
<td>Figure 6: Item Tardiness Distribution for Equipment Category 3</td>
<td>96</td>
</tr>
<tr>
<td>Figure 7: Equipment Category 11 Pareto Chart</td>
<td>97</td>
</tr>
</tbody>
</table>
LIST OF ABBREVIATIONS

A&M ................................................................................................................. Aviation and Missile
AFSB ................................................................. Army Field Sustainment Brigade
AFSBn .................................................................. Army Field Sustainment Battalion
MWP .............................................. Army Intermediate Maintenance Workload Problem
AMC ...................................................................................... Army Materiel Command
AMSAA .................................................... Army Materiel Systems Analysis Activity
APF ........................................................... Active Period Formulation
ASC ............................................................... Army Sustainment Command
BD .............................................................. Benders’ Decomposition
C&E ................................................................. Communications and Electronics
CG ................................................................................. Column Generation
CP ................................................................................. Constraint Programming
DoL ........................................................................... Directorate of Logistics
DoM ........................................................................... Directorate of Materiel
EDD ................................................................. Earliest Due Date
GAP ................................................................. Generalized Assignment Problem
IP ......................................................................................... Integer Program
IAG ................................................................. Item Arrival Group
JM&L ............................................................... Joint Munitions and Lethality
JSSP ......................................................... Job Shop Scheduling Problem
LBE ................................................................. Left Behind Equipment
LCMC .......................................................... Life Cycle Management Command
LIW ............................................................... Logistics Information Warehouse
MCJSSP .................................................. Multiple Capacitated Job Shop Scheduling Problem
MILP ............................................................. Mixed Integer Linear Program
MIP ................................................................. Mixed Integer Program
MS ................................................................................. Minimum Slack
PSP ............................................................. Planning and Scheduling Problem
RMP ............................................................. Restricted Master Problem
RSF ............................................................. Restricted Standard Formulation
SF ...................................................................................... Standard Formulation
SoR ................................................................................. Source of Repair
SOR_ID ....................................................... Source of Repair Identification
SWF ............................................................ Sliding Windows Formulation
TA&A ............................................................. Tank Automotive and Armaments
UIC ........................................................................ Unite Identification Code
ABSTRACT

PLANNING AND SCHEDULING FACILITY WORKLOAD

Ryan R. Squires, PhD
George Mason University, 2014
Dissertation Director: Dr. Karla L. Hoffman

This research considers a military maintenance planning and scheduling problem posed by the Army Materiel Systems Analysis Activity (AMSAA). The problem requires that we assign each of a large set of military equipment items to one of a number of possible maintenance facilities and schedule repairs for those items at each facility. Each item originates at a particular location and must arrive at a certain destination; transportation costs are incurred for moving the equipment items from origin to repair facility and from repair facility to destination. In general, we would like to achieve a minimal cost for operating this system. However, we would also like to understand how varying system total allowed tardiness affects total operating cost. Ideally, we would like to have some knowledge of the efficient frontier, so that decision makers can make well-informed decisions concerning the trade-off between cost and tardiness (readiness).

AMSAA constructed an integer program, which they call the “Sliding Windows Formulation” (SWF), to assign and schedule item repairs. In order to visualize the
efficient frontier, they employ an objective function that is a scalar weighted combination of cost and tardiness. The weights can be varied and the model resolved to obtain different solutions. In order to achieve feasible solutions, the developer of the SWF system groups similar items into a batch represented by a single variable and limit the amount of exploration that the solver may perform (limiting the size of the branch and bound tree). Despite these restrictions, large instances still require runtimes in excess of sixteen hours. For these instances, sliding windows formulation is solved in six-month segments and assembled into a single solution, which (Kotkin et.al. 2011) identifies as the “Rolling Horizon” heuristic.

We opt to model the multi-objective aspect of the this problem by minimizing cost and constraining system total tardiness. Initially, we construct and compare three models: a model in standard formulation and two models based on different decomposition methods. One decomposed model is based on Bender’s Decomposition, an approach that has been used by (Hooker 2007) to solve the Planning and Scheduling Problem (PSP), a problem similar to our problem of interest. We also construct a model based on extensive reformulation and column generation using Dantzig-Wolfe decomposition. While each of these approaches has certain merits, we find neither entirely satisfactory. Therefore, we consider the use of variable restriction to obtain good feasible solutions. The success of this approach in obtaining upper bounds quickly led us to outline two alternative methods: one approach uses the upper bound obtained from a variable restricted model with a lower bound obtained from a Benders Decomposition Formulation. Alternatively, we also use an approach known as sifting to obtain a strong
lower bound and use the resulting variables to find an upper bound on the problem. We demonstrate substantially reduced optimization times for the largest instances of the MWP. We also demonstrate the use of our improved solution methods as means to identify and visualize the efficient frontier that trades off cost for tardiness.
I-INTRODUCTION

In this section, we describe U.S. Army maintenance operations and the organizations that perform them. We describe the work performed at Directorate of Logistics and Directorate of Materiel (DoL/DoM) intermediate maintenance facilities in support of Army maintenance. Within this context, we define the Army Intermediate Maintenance Workload Problem (MWP). Finally, we outline our research questions and approach.

1.1 Army Maintenance Operations

The purpose of the United States Army’s maintenance system is to “generate/regenerate combat power and to preserve the capital investment of weapons systems and equipment to enable mission accomplishment.” (US Army 2011) In order to accomplish this, the Army organizes its maintenance operations into the field-level and the sustainment-level. Field-level maintenance, the lower level of maintenance, typically occurs at or near the current location of the equipment and is typically conducted by the operator or crew. When the complexity of a field-level maintenance repair or service is beyond the operator’s level of training, specifically trained maintainers, assigned to deployable Army units also perform field-level maintenance. An objective of field-level maintenance operations is to repair the item and return it to the user as soon as possible. Higher-level maintenance is known as sustainment maintenance. In general, sustainment-
level maintenance removes the item of equipment from a unit. The losing unit, if still authorized the item, would then request a like-item replacement. The sustainment-level maintenance organization repairs or disposes of the item, as appropriate. If repaired, the item is returned to the Army supply system for delivery to a unit in need of the particular item. In other words, when an item of equipment undergoes sustainment-level maintenance it will not typically be returned to the original user or even unit.

1.2 Sustainment Level Maintenance Operations
Sustainment-level maintenance consists of two sub-levels: depot sustainment and below-depot sustainment. Depot-level maintenance occurs at one of the Army’s depot repair facilities, while below-depot level sustainment typically occurs at an Army installation Directorate of Materiel or Directorate of Logistics (DoM/DoL) intermediate maintenance facility. The Army owns and operates approximately 160 intermediate maintenance facilities on major Army installations (Kotkin et.al. 2011). Intermediate maintenance facilities are organized into shops with each shop providing capabilities to perform certain maintenance operations on certain types of equipment. Three job types for intermediate maintenance facilities are: (1) Left Behind Equipment (LBE), (2) Periodic and (3) RESET. LBE jobs are services on equipment items left at a home station by a deploying unit unnecessary for its particular deployed mission. The responsibility for maintaining this LBE often falls upon intermediate maintenance facilities. RESET jobs are for equipment that has been returned after deployment. This equipment is often in need of extensive repair before it can be used again and these items are submitted to sustainment maintenance organizations that must restore it to a specified level of
operability and then deliver it to a unit in need of the particular item of equipment. In some cases, RESET sustainment maintenance is performed at an Army depot. In other cases, it will be performed by intermediate maintenance facilities. The third job type, periodic maintenance, refers to recurring services on items of equipment, often owned by the installation. Of the three major job types for intermediate maintenance facilities, only RESET jobs are strictly sustainment-level maintenance. Generally, LBE and periodic maintenance jobs qualify as field-level maintenance since equipment is not surrendered to the Army Supply System but rather returned to the original unit.

Army installations own and operate the intermediate maintenance facilities, but below depot level maintenance of materiel is overseen by the Life-Cycle Management Commands (LCMCs), subordinate elements of Army Materiel Command, each with responsibility for a broad category of equipment. While the LCMCs, in general, exercise control over their respective depot level sustainment facilities, the LCMCs must coordinate below-depot level sustainment maintenance at DoM/DoL facilities through the Army Sustainment Command (ASC) and its subordinate Army Field Support Brigades (AFSBs) and Army Field Support Battalions (AFSBns) who actually perform work loading of facilities within their areas of responsibility. (US Army, 2011)

1.3 Intermediate Maintenance Facility Workload
Managing the workload of intermediate maintenance facilities has many factors. Each item requiring service at an intermediate maintenance facility, whether RESET, LBE or periodic must be assigned to an intermediate maintenance facility as a job. If movement of the equipment is necessary, transportation costs are incurred from origin to
facility and from facility to final destination. Transportation times also become a factor as this may require a week or more. There are also costs associated with conducting a particular job type at a particular facility. In other words, the release date for a job is its arrival into the maintenance system, not its arrival at a facility. Similarly, due date for a job is its desired due date to an Army unit, not its desired completion time at a maintenance facility. RESET jobs will typically arrive by ship to a port, which will be their origin. The scheduled arrival of the ship will be the release date for RESET jobs. LBE and Periodic jobs typically originate on Army installations. The release date for LBE jobs is based on a unit deployment while the release date of Periodic jobs is based on a scheduled service for the equipment item. Both RESET and LBE jobs tend to arrive at the same time and as part of large groups of items. Job due dates are prescribed according to job type. Job destination for RESET jobs is according to unit demand for the particular equipment item and is prescribed as part of the job. Job destination for other types of jobs (Periodic and LBE) will typically be the originating installation. Note that each unit has a unique unit identification code (UIC). Each UIC maps to a particular location, usually an Army installation. A location may have one or many UICs. Maintenance facility shops are identified by a unique source of repair identification (SOR_ID). Like the UIC, the SOR_ID uniquely determines the facility’s location. Each intermediate maintenance facility has only a given number of direct labor hours available for use and is specific to the type of repair. Fortunately, all jobs do not compete for all resources. This means that workloads can be partitioned according to several equipment-type-groups that do not compete for maintenance facility resources. However, even
within an equipment-type-group partition, not all facilities are able to perform all jobs. (Kotkin et.al. 2011).

1.4 Problem Statement

Given continued budgetary pressures and a fiscally conservative environment, the Army now faces and will continue to face difficult decisions concerning its ability to repair all equipment in a timely fashion. In (Kotkin et.al. 2011), a team at the Army Materiel Systems Analysis Activity (AMSAA), identified the potential to save money and reduce job tardiness by optimizing the workload of intermediate maintenance facilities. The team developed a model that they call “Sliding Windows Formulation” (SWF) to assign jobs to intermediate maintenance facilities and schedule the intermediate maintenance using a binary-integer program. In this model, complete batches of similar items (item-batches), with similar characteristics, are assigned to a facility and a starting week based on a binary optimization, where the variables determine these assignments. The processing time for item-batches varies according to facility and the particular type of equipment. During each week of processing, an item-batch consumes a given quantity of labor hours at its assigned facility. Again, this value will vary according to the facility and equipment type. Once an item–batch starts processing, other item repairs do not preempt it. The total labor hours consumed by all item-batches processing must not exceed the facility’s weekly available labor hours. An implementation of this integer program model, called the Sliding Window Formulation (SWF) has been shown to provide feasible solutions to instances of the Army Intermediate Maintenance Workload Problem (MWP). (Kotkin et.al 2011) shows that these solutions may save several
hundreds of millions of dollars over previous workload plans for the two-year model time horizon while also reducing total systemic tardiness by thousands of weeks for the same two-year planning horizon.

Scheduling item-batches rather than individual jobs simplifies the model by reducing the number of variables in the formulation. This was deemed necessary due to the size of many problem instances. Even after adopting this simplifying measure, it was often necessary to limit the size of the solution search space by limiting the solver’s branch and bound tree size. Even so, solution time for the larger instances approaches twenty hours. The problem is compounded by the need to provide decision makers with an understanding of the trade-off between cost and tardiness in the system (i.e. can we reduce our costs substantially by accepting some job tardiness?). This led the team to form a composite objective function and perform multiple model runs with varied weight trade-off objectives (Kotkin et.al. 2011). This method aims to provide some solution points to outline the efficient frontier (i.e. the Pareto optimal surface). However, a sufficient number of runs to guarantee that the full trade-off space was understood is generally not reasonable due to the total computation time required. Additionally, the restriction of the problem induced by the batching of items means that the lower bound for the true problem of interest is not known.

(Ralphs and Galati 2010) indicate that decomposition methods have become the preferred solution approaches for many combinatorial problems and there is strong reason to believe that decomposition methods will provide a means for better solution of MWP. Indeed, the structure of the MWP suggests the application of these techniques, in
particular Logic-Based Benders’ Decomposition and Dantzig-Wolfe Reformulation. In brief, decomposition methods seek to exploit the underlying problem structure to yield better bounds for the problem of concern. These bounds can then be used to drive the branch-and-bound algorithm to an optimal or at least proven good solution (i.e. a solution with a bound on how far it can be from optimality). Since these methods offer a good prospect for improved solution quality and possible reduced solution time, we explored their utility with regard to this particular application. Our research questions were designed to help to identify useful formulations, implementations and an overall methodology for the MWP.

Can a model based on Logic-Based Benders’ Decomposition provide optimal or near-optimal (minimum cost) solutions to instances of the MWP within a reasonable time for a given amount of acceptable system tardiness (average job tardiness)?

Can a model based on Dantzig-Wolfe (extensive) reformulation and column generation provide optimal or near-optimal (minimum cost) solutions to instances of the MWP within a reasonable time for a given amount of acceptable system tardiness (average job tardiness)?

For the most challenging instances of the MWP, can these new formulations and implementations, be used together and in conjunction with existing formulation results (SWF) to identify solutions along the efficient frontier for instances of the MWP (i.e. are there formulation and implementation synergies)?
1.5 **Objective of this Research**

The objective of this research is to identify formulations, implementations and methodologies for solving full-sized instances of the MWP that improve solution quality and/or reduce solution time.

1.6 **Research Approach**

AMSAA, the Army Materiel Systems Analysis Activity, made available twelve data sets originally used for their modeling of the MWP. Each is based on a partition of equipment items that do not compete for maintenance resources. Of these twelve, one (Group 9) could not be scheduled within a two-year time frame. We exclude this set from consideration. Groups 3 and 5 each contain job types that cannot be completed at any facility. For these two groups, we follow the precedent established in (Kotkin, et.al. 2011) and exclude from the dataset those jobs that cannot be completed at any facility.
These data sets were originally created from two sources, a centralized repository of maintenance data called Logistics Information Warehouse (LIW) and Life Cycle Management Command (LCMC) maintenance job lists. In order to protect this information, which is designated “For Official Use Only”, all information not directly relevant to the optimization problem will be obscured.

We report the feasible solutions identified in (Kotkin, et.al. 2011) in Table 2. These were obtained by the SWF model and based on item-batches. Some Equipment Groups have multiple associated solutions that were obtained through repeat solution and varied objective function weights.
Table 2: SWF Results Reported in (Kotkin et.al 2011)

<table>
<thead>
<tr>
<th>Equipment Group / Weight Setting</th>
<th>2. SWF Operating Cost from (Kotkin et.al. 2011) (USD)</th>
<th>3. SWF Tardiness from (Kotkin et.al. 2011) (wks)</th>
<th>4. Runtime Reported in (Kotkin et.al. 2011) (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/M</td>
<td>67339.72</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2/M</td>
<td>1309658</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>3/T</td>
<td>2563741</td>
<td>229</td>
<td>14</td>
</tr>
<tr>
<td>3/M</td>
<td>2531260</td>
<td>235</td>
<td>14</td>
</tr>
<tr>
<td>3/C</td>
<td>2323977</td>
<td>1317</td>
<td>15</td>
</tr>
<tr>
<td>4/M</td>
<td>1324432</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>4/C</td>
<td>1162923</td>
<td>3137</td>
<td>57</td>
</tr>
<tr>
<td>5/M</td>
<td>2392397</td>
<td>1736</td>
<td>4514</td>
</tr>
<tr>
<td>6/M</td>
<td>3751779</td>
<td>0</td>
<td>33</td>
</tr>
<tr>
<td>7/M</td>
<td>16051777</td>
<td>0</td>
<td>180</td>
</tr>
<tr>
<td>8/T</td>
<td>18857438</td>
<td>602</td>
<td>369</td>
</tr>
<tr>
<td>8/C</td>
<td>18856734</td>
<td>608</td>
<td>842</td>
</tr>
<tr>
<td>10/M</td>
<td>145474000</td>
<td>360</td>
<td>57706</td>
</tr>
<tr>
<td>11/M</td>
<td>50183986</td>
<td>840</td>
<td>68460</td>
</tr>
<tr>
<td>12/M</td>
<td>260209631</td>
<td>2547</td>
<td>71473</td>
</tr>
</tbody>
</table>

Modeling computation time results in (Kotkin, et.al. 2011) demonstrate a run time in excess of sixteen hours for Equipment Group 10 and run times approaching twenty-hours for Equipment Groups 11 and 12.

We first created a standard formulation model of the un-restricted MWP. This alternative formulation uses integer variables that, in effect, permit the model to select batch sizes rather than forcing the model to assign and schedule a complete batch as a unit. Rather than minimizing a weighted combination of objectives, we elect in all our models to constrain tardiness to a fixed quantity (or less) and minimize total operating
cost. We feel this gives a better means to explore the Pareto surface than that provided by varying weights in a complex objective function. Using the same cost minimizing, tardiness constraining approach, we constructed models based on Benders Decomposition and Dantzig-Wolfe Decomposition and Column Generation. We attempted to solve each constructed model once for each of the 14 solutions given by the feasible solutions above in Table 2.

When we found all three models less than completely satisfactory, we developed a variable restricted model to provide good feasible solutions and upper bounds. Like SWF, this model represents a restriction of the full problem and provides no valid or useful lower bound. We paired the results of this model with lower bound obtained using the Benders Decomposition to give MWP solutions of known quality for all problem instances. We also developed a sifting formulation that uses the variable restricted model to produce a feasible basis and then prices those variables not included in the restricted model to obtain both lower bounds and excellent upper bounds.

We compare the solution methods using several metrics including upper and lower bound, and solution time.

1.7 Research Contributions
The military its maintenance and materiel management organizations can benefit from this research. Good, workable methods of solving instances of this maintenance planning and scheduling problem will provide maintenance planners with the ability to reduce maintenance costs and understand the readiness impacts implicit in a given level of funding. Applications for this research exist outside the Army as well. According to
problems structurally similar to the MWP arise often in manufacturing and supply chain applications. Similar problems seem to arise often in the chemical production context both (Turkay and Grossmann 1996) and (Timpe 2002) consider related problems in the chemical production context.

This research should also be of interest the academic communities concerned with general integer and combinatorial optimization as well as those interested in industrial planning and scheduling. In our consideration of this particular problem, we consider several areas of active research. First, we offer several insights concerning good modeling practice and the development of good problem restrictions. Second, decomposition methods and their utility for particular problems is an active area of study. While a Bender’s Decomposition framework has been applied to similar problems (Hooker 2007), our problem requires substantially different treatment that prior work due to the multi-objective aspect of the problem. Additionally, the prior work using the Bender’s Decomposition framework uses randomly generated instance data rather than real-world instance data. To our knowledge, our application of column generation, variable pricing and sifting to a planning and scheduling type problem is unique. In terms of broad integer programming technique, there are few examples within the literature of the use of both Bender’s Decomposition and Column Generation within the context of a single problem. Our research provides some insight into the strength and weakness of each approach relative to a particular problem. Finally, even given the other unique aspects of our problem, our research considers planning and scheduling problem
instances far larger than any other we have seen in the relevant literature, that is, other than (Kotkin et.al 2011).
II-LITERATURE REVIEW

The Army Intermediate Maintenance Workload Problem (MWP) presents several notable modeling challenges. First, it is multi-objective in nature. Second, the problem requires the modeling of two sequential actions: item assignment and facility scheduling. In item assignment, items must be assigned to facilities for their repair, which determines the total cost of the solution. Then, in facility scheduling, items assigned to each facility must be scheduled within the facility’s constraints. The scheduling step determines the total tardiness. It is possible to model these actions in a single formulation as was done in previous work on the problem in (Kotkin et.al. 2011) but such modeling limited the size of the problem that could be solved and resulted in some very long solution times. Alternatively, it is also possible to decompose the model into a master problem that looks at the problem as a whole and sub-problems that decompose the sequential actions and provide information back to the master problem via dual price information. In this research, we evaluate decomposition approaches in order to identify overall methodologies for the solution of the MWP.

2.1 Multi-Objective Optimization

Because the MWP is concerned with both tardiness and cost, it may be considered a multi-objective optimization. There is an extensive literature on the topic of multi-objective optimization and important concepts in multi-objective optimization theory
come from both the field of mathematical optimization as well as the field of game theory. Some terminology and concepts from the literature of multi-objective optimization will be relevant to consideration of the MWP.

When an optimization model has more than one objective function or performance metric (e.g. cost and total tardiness), a feasible solution to the model may be scored according to each objective function. Efficient points are those solutions where, “no other feasible solution scores at least as well in all objective functions and strictly better in one.” Efficient points are also known as Pareto optimal or non-dominated. The efficient frontier is the collection of all efficient points for a model. (Rardin 1998).

(Rardin 1998) outlines three approaches to optimize models with vector valued (i.e. multi-objective) objectives. One simple approach for multi-objective problems is called preemptive optimization. In this approach, objectives are ranked according to their importance. An optimal solution for the most important objective is found and this objective value is used to form a constraint that forces the model to meet this objective value when it is optimized according to the second most important objective. If there are more than two objectives, formation of constraints and optimization with respect to lesser objectives continues until the model is optimized with respect to the least important objective. This method will generally find a single efficient point. A drawback is that the method effectively weights the most important objective(s) very heavily since less important objectives can only impact the final solution if they can give an alternate optimal solution with respect to the (fixed) more important objectives.
A second approach for multi-objective problems from (Rardin 1998) is weighted sums of objectives. In this method, each objective is weighted according to its relative importance and the weighted objectives values are summed into a single scalar value for optimization. For this reason, other sources refer to this method as scalarization. A feasible solution under the weighted sums (or scalarization) approach will be an efficient point under certain conditions, however varying the weights in a weighted sum/scalarization will not necessarily find all solutions to a multi-objective problem (Pappalardo 2008). A third approach for multi-objective problems from (Rardin 1998) is called goal programming. This research does not anticipate the direct use of goal programming, but a brief coverage of this important and popular approach for multi-objective optimization follows. In goal programming, decision makers establish target values or goals for each objective. These are then included in the model formulation as soft constraints that have a non-negative deficiency variable to permit non-attainment of the soft constraint in feasible solutions. Optimization of the goal program can then proceed in either a preemptive manner (i.e. minimize a one deficiency variable at a time) or as a minimization of an appropriately weighted sum of deficiency variables. In some circumstances, the straightforward goal programming formulation can produce solutions that are not efficient, however there are modification techniques that can be used to ensure the method gives efficient solutions.

Although largely outside the scope of this research, it is important to note that multi-objective optimization typically includes a second phase in which a decision-maker, presented with either a set of objectives or a set of Pareto optimal solutions,
selects weights for the objectives or selects from among the Pareto-optimal solutions. Four classes of decision methods are listed in (Pappalardo 2008).

In this dissertation, since there are only two goals, one can set one of the goals as the objective function and solve for a specific restriction on the other goal that is placed as a constraint of the problem. By varying the right-hand-side of this goal-constraint, one can determine a Pareto frontier for viable ranges of this goal.

2.2 Assignment Models
The generalized assignment problem (GAP) model is a classic problem in the field of combinatorial optimization that is known to be NP-hard (Savelsbergh 1997). It is, for present purposes, a reasonable model of the assignment step that establishes solution cost in the MWP. A minimization version of the GAP asks one to assign each job in a set to an agent (in our case, a repair facility) at minimum cost within some working capacity constraint for each agent. This version of the GAP is defined below. This formulation is known as the standard formulation of the integer program. An alternative formulation, known as the extensive formulation will be presented later. For the GAP, we define the following sets:

- $I$ set of jobs for assignment,
- $K$ set of agents (e.g. facilities).

We define the following variables:

- $x_{ik}$ binary variables equal to 1 if task $i$ is assigned to agent $k$ and 0 otherwise.

Parameters for the GAP include:

- $c_{ik}$ cost to assign job $i$ to agent $k$ (scalar),
\( \sigma_{ik} \) resource consumed when job \( i \) assigned to agent \( k \),

\( m_k \) resource available to agent \( k \).

**Objective:**

\[
\min \sum_{i} \sum_{k} c_{ik} x_{ik}
\]

**Subject to:**

\[
\sum_{k} x_{ik} = 1 \forall i \in I \tag{1}
\]

\[
\sum_{i} \sigma_{ik} x_{ik} \leq m_k \forall k \in K \tag{2}
\]

Constraint set (1) ensures that each job is assigned to exactly one agent. Constraint set (2) ensures that the jobs assigned to that agent do not exceed the agent’s working capacity.

A modified version of the GAP will take on the role of a *master model* in both proposed new models. In a Logic-Based Benders’ decomposition, a version of the GAP, together with discovered cuts, will take on the role of a master or coordinating model. In a Dantzig-Wolfe reformulation, a reformulated version of the GAP will take on the role of a *restricted master problem* (RMP). These may be left out of a Logic Based Benders’ Decomposition formulation. These may also form the basis of a sub-problem relaxation as will be seen in the discussion of the work in (Hooker 2007)

### 2.3 Scheduling Models

Industrial and manufacturing scheduling is a long-standing and active area of research. However, much, if not most work in this area pertains to serial (single-job-at-a-time) processing and scheduling. In this research, we have capacitated or cumulative
scheduling, a generalization of the more common serial scheduling and processing. The facilities of interest have the ability to process multiple jobs or items during a time unit subject to a resource constraint. It is not particularly difficult to model a capacitated scheduling problem with the necessary parallel capacitated processing within an integer or combinatorial optimization framework. There are two alternative views of the problem: One can specify continuous variables $x_i$ that define the start time of job $i$ or, alternatively, one can divide the periods into discrete time points and let $x_{is} = 1$ if job $i$ starts at time $s$. The continuous model allows symmetries in the formulation that make it very difficult for a standard integer optimization code to solve. Several researchers confirm that the continuous model is very slow and difficult to solve, including (Hooker 2007) and (Kotkin et.al. 2011). We therefore provide here the discrete time model for a single facility seeking to minimize tardiness: We define the following sets for the cumulative scheduling model:

$I$ set of jobs for processing,

$S$ set of time periods.

Parameters for the cumulative scheduling model include:

$l_i$ processing time (duration) for job $i$,

$\mu_i$ resource required for job $i$ for each time unit of processing,

$r_i$ release date for job $i$ (time when job is available to start processing),

$d_i$ due date for job $i$ (time when job should be complete),

$m$ resource capacity per time unit.

Parameterized sets for the cumulative scheduling model include:
\[ U_i = \{s' \mid s - l_i < s' \leq s\} \] set time periods during which job \(i\) will be processing if the job begins at time \(s\).

Variables for the cumulative scheduling model include:

- \(x_{is}\) binary variables equal to 1 if task \(i\) is starts at the beginning of time period \(s\) and zero otherwise,
- \(t_i\) continuous variables representing the tardiness of job \(i\).

**Objective (Minimum Tardiness):**

\[
\min \sum_{i} t_i
\]

**Subject to:**

\[
\sum_{s} x_{is} = 1 \forall i \in I
\] \hspace{1cm} (1)

\[
\sum_{l} \sum_{s' \in U_i} \mu_{i} x_{is'}
\] \hspace{1cm} (2)

\[
\sum_{0<s<s'} x_{is} = 0 \forall i \in I
\] \hspace{1cm} (3)

\[
t_i \geq \sum_{s} (s + l_i - d_i)x_{is} \forall i \in I
\] \hspace{1cm} (4)

\[
t_i \geq 0 \forall i \in I
\] \hspace{1cm} (5)

\[
x_{is} \in \{0,1\} \forall i \in I, s \in S
\] \hspace{1cm} (6)

Constraint set (1) ensures that each job is scheduled for processing. Constraint set (2) ensures that the required resources for all jobs processing do not exceed the facility’s available resources (e.g. direct labor hours). Constraint set (3) ensures that no job begins before it is released. Note that it is also possible to enforce (3) by instantiating
variables only for \( s \geq r_i \). We adopt this approach in our modeling. Constraint set (4) constrains the tardiness for each job to be greater than the job’s start time plus its processing time minus its due date. Constraint set (5) ensures that each job has a tardiness exceeding zero, since there is no “credit” for early job completion. Constraint set (6) establishes each \( x_{ts} \) as a binary variable. The minimization of this problem will drive each \( t_i \) to its lowest possible value established by the greater of either constraint set (5) or constraint set (6).

Although it is fairly easy to define the necessary model, scheduling models in general, whether for machines or facilities are computationally challenging and known to be NP-hard. In many situations, approximate methods like dispatching rules are used to produce “good” rather than known optimal schedules. The Earliest Due Date (EDD) is one such rule. When using the EDD scheduling rule, as a machine becomes available, the available job with the next (i.e. earliest) due date begins processing on the available machine. The Minimum Slack (MS) rule is another dispatching rule. Using this rule, as a machine becomes available, the job with the least slack begins processing on the available machine (Pinedo 2009). When the present time is \( s \), slack is defined as:

\[
S_i(t) = \max(d_i - l_i - s, 0)
\]

Neither the EDD nor the MS dispatching rules is suitable for the MWP due to the need for cumulative scheduling to take advantage of parallel processing. However, some
simple and obvious modifications might give a rule that quickly produces feasible, though not provably optimal minimum tardiness schedules for a single facility.

We would have no evidence concerning the quality of this rule with respect to creating schedules with minimal tardiness, but it could provide schedules relatively quickly. Given these feasible solutions, we can simply audit each solution for tardiness measures. These feasible solutions give an upper bound (no worse than value) for the objective function (total tardiness), so they permit one to eliminate regions of the feasible space from consideration and may speed solution considerably. Coverage here is primarily concerned with single-facility scheduling because this is the expected form of sub-problems. However, it is important to note that dispatching rules can be used for environments with multiple machines (parallel-machine configurations). Analogously, it would be possible to create a dispatching rule system to both assign and schedule jobs for a facility-type scheduling problem.

Constraint Programming is another solution method frequently used for scheduling problems. The Constraint Programming paradigm does not include a concept of optimality, only of constraints that must be satisfied by candidate solutions. Two works that make use of Constraint Programming for facility-type scheduling include (Nuitjen and Aarts 1997) and (Hooker 2007). (Hooker 2007) will be discussed in greater detail in a subsequent section. (Nuitjen and Aarts 1997) concerns a problem called the Multiple Capacitated Job Shop Scheduling Problem (MCJSSP). The MCJSSP, a generalization of the Job Shop Scheduling Problem (JSSP), pertains to a set of jobs, each of which must be processed on a set of machines in a particular order, specific to the job.
While the JSSP has machine-type scheduling, the MCJSSP permits parallel capacitated processing (facility-type scheduling). (Nuitjen and Aarts 1007) uses a Constraint Programming framework to determine if a set of jobs can be completed by some overall deadline (i.e. makespan).

2.4 Sliding Windows Formulation Model

Recognizing inefficiency in the maintenance system as it was being operated prior to 2010, AMSAA in (Kotkin et.al. 2011), performed the first work directly relevant to the MWP. (Kotkin et.al. 2011) models the problem as a bi-objective integer program that seeks to minimize a weighted sum of cost and tardiness. In order to identify a Pareto-optimal surface, (Kotkin et.al. 2011) suggests varying weights associated with the cost and tardiness components of the objective function. From the earlier discussion of multi-objective optimization, this method is a weighted-sum optimization and each feasible solution will be an efficient or Pareto-optimal point.

Each item-batch represents an individual piece of equipment or batch set of similar items requiring a particular type of below-depot-level sustainment (intermediate) maintenance. Each item-batch also has an origin and destination. Processing times and man-hour (i.e. resource) requirements are based on averages for each type of item of equipment by job type (LBE, RESET or Periodic) and by facility. Each work facility is capacitated according to its direct labor hours available. Transportation costs and transport times to and from each location are determined according to Army distribution data. There is also a cost associated with each equipment item, facility and repair type. (Kotkin et.al. 2011) develops two formulations for the problem, the Active Period
Formulation (AFP) and the Sliding Windows Formulation (SWF). The APF is a continuous time, event-based model that is similar to a model developed in (Turkay and Grossman 1996). The SWF is a discrete time formulation. In testing, the SWF was found to be superior in terms of solution time. (Hooker 2007) also found event-based formulations difficult to solve computationally. The bulk of analysis in (Kotkin et.al. 2011) was performed using the SWF, so only this formulation of the model is described in depth. The following sets are defined for SWF:

$I$ set of repair job-batches,
$K$ set of repair facilities,
$S$ set of time periods,
$W$ set of resource,

Parameters for SWF include:

$q_i$ item quantity associated with job-batch $i$,
$\sigma_{ik}^1$ cost per item to ship as a carcass from origin location of job-batch $i$ to location of facility $k$,
$\sigma_{ik}^2$ cost per item to ship finished item from location of facility $k$ to location of destination for job-batch $i$
$p_{ik}$ cost per item to perform job $i$ at facility $k$
$r_i$ release date of item $i$ at its origin
$d_i$ due date of item $i$ at its destination
$z_i$ tardiness weight for job $i$
$l_{ik}$ time required to perform job $i$ at location $k$

$\tau^1_{ik}$ time to ship item as carcass from location $i$ to facility $k$

$\tau^2_{ik}$ time to ship finished item from location $i$ to facility $k$

$m^w_{ks}$ capacity of resource $w$ at location $k$ during time period $s$

$\mu^w_{ik}$ amount of resource $w$ required job $i$ requires during each time period of its repair time ($l_{ik}$) if it is repaired at facility $k$

$\lambda_1$ objective function weight for tardiness

$\lambda_2$ objective function weight for cost

$M$ objective function weight for unscheduled jobs

$c_{ik}$ total cost per item, including shipping costs, to repair an item in job-batch $i$ at facility $k$; $c_{ik} = \sigma^1_{ik} + \rho_{ik} + \sigma^2_{ik}$.

An additional parameterized set for SWF includes:

$U_{iks} = \{s' | s - l_{ik} < s' \leq s\}$ set time periods during which job $i$ will be processing if the job begins at time $s$ on facility $k$.

Variables:

$x_{iks}$ binary variables equal to 1 if job-batch $i$ begins processing at facility $k$ at time period $s$ and 0 otherwise,

$w_i$ binary variables =1 if job-batch $i$ is not scheduled and 0 otherwise,

$T_i$ positive continuous variables representing the tardiness of job-batch $i$.

Objective:
min \lambda_i \sum_i z_i q_i T_i + \lambda_2 \sum_i \sum_K \sum_s q_i c_{ik} x_{iks} + M \sum_i q_i w_i

Formulation:

\begin{align*}
T_i & \geq \sum_K \sum_s (s + l_{ik} + \tau_{ik} - d_i) x_{iks} \forall i \in I \\
\sum_K \sum_s x_{iks} + w_i & = 1 \forall i \in I \\
\sum_s x_{iks} & = 0 \forall i \in I \\
\sum_I \sum_{s' \in U_{iks}} \mu_{iks}^{w} x_{iks} & \leq m_{iks}^{w} \forall k \in K, s \in S, w \in W
\end{align*}

Constraint set (1) sets tardiness for each job-batch by adding the starting time period of a job-batch, the job-batch’s processing time at its assigned facility and its transportation time from repair facility to destination and subtracting the due date of the job. Constraint set (2) requires that each job be scheduled. Constraint set (3) prevents a job-batch from beginning processing at a repair facility before its release date plus the transportation time from the job’s origin to the repair facility. Constraint set (4) enforces the resource constraint at for each time period, at each facility and for each resource type. Constraint set (5) prevents negative tardiness.

The SWF formulation does not permit preemption of jobs and does not include any job precedence constraints. In practice, the problem is reduced in size by solving smaller problems each considering one of fifteen separate categories of equipment that do
not share maintenance facility resources. To further reduce the problem size, similar items with similar characteristics are batched into a single item-batch with processing time and required direct labor-hours as an aggregate of the component items’ estimates. Even after these reductions, due to residual problem size, the model has often been used in a “rolling-time window” fashion in which, over the time horizon, a subset of jobs are considered (based on release date). These are scheduled by the model and fixed. Then the next subset of jobs are scheduled and fixed. Most data for the model are estimated from data available on Logistics Information Warehouse (LIW), an Army-wide system for maintenance and logistics data.

The MWP and its AMSAA models (i.e. APF and SWF) differ from most published literature for several reasons. Cumulative scheduling, the processing of multiple item repairs simultaneously at a facility limited according to available man-hours differentiates the problem from the well-researched parallel machine scheduling models, where jobs are processed one at a time on a set of machines. As already noted, another atypical feature is the bi-objective function. Although the SWF model formulation accounts for the possibility of multiple resource types (index $W$) each capacitating processing at the repair facilities, in practice, processing is only capacitated by direct labor hours available (i.e., $|W| = 1$). A final interesting aspect of the model formulation is the explicit inclusion of transportation times and their impacts on tardiness. This can be easily implemented as a separate data structure as suggested by the model, so it will does complicate the problem’s solution.
2.5 Logic Based Benders’ Decomposition Approach

(Benders 1962) first developed the method of Benders’ Decomposition where a problem is decomposed into a problem or set of problems that are relatively easy to solve when a set of particular complicating variables take on known values. Benders’ method solves the simpler, linear sub-problem in order to obtain pricing information to create constraints, or cuts, in the master problem that are functions of the complicating variables. The benefit of this information exchange is that the master typically will not need to enumerate as many values of complicating variables as would be necessary for a straight branch and bound solution approach. Another benefit is that the Benders’ cuts that are generated often tighten the lower bound on a minimization problem quickly. Thus, this bound coupled with good feasible solutions can be used to get provably good solutions to users in reasonable times. This is not true for the straightforward optimization formulation where symmetry often causes the lower bound to be a poor approximation to the optimal integer solution. One problem with the Benders’ decomposition method is that the sub-problems must be linear programming problems since it is the dual prices from those linear programs that are used in the cuts. Scheduling, as a sub-problem, is non-linear and so, classical Benders’ Decomposition is limited in its utility.

Using non-linear duality, (Geoffrion 1972) generalizes Benders’ method to include non-linear, convex sub-problems. (Hooker and Yan 1995) and (Hooker 1995) generalized the Benders’ method to include an even wider array of problems with Logic-Based Benders’ Decomposition. In Logic-Based Benders’ Decomposition, knowledge of the particular problem structure is leveraged to devise a bound-generation scheme.
(Hooker 2007) takes a Logic Based Benders’ Decomposition approach to a problem called the Planning and Scheduling Problem (PSP), which is fundamentally similar to the MWP. Jobs or tasks are assigned to entities for processing. Whereas in the MWP, jobs are assigned to maintenance facilities, for (Hooker 2007), jobs are assigned to “machines”. Each job must be scheduled no earlier than its release date and must finish no later than its due date or deadline. In addition, each job has a processing time and resource requirement during each unit of processing time. There is a cost to assign each job to each machine. Processing time and resource requirements may also vary according to which machine a job is assigned. Significantly, jobs may run in parallel subject to a machine resource capacity constraint (i.e. parallel capacitated processing and facility type scheduling). (Hooker 2007) considers three different objectives, minimizing total cost, makespan (completion time of last job to finish), and total tardiness. When minimizing total cost, (Hooker 2007) treats the due dates as fixed, modeling them as constraints. When minimizing tardiness, job completion may exceed due date but at cost to the objective function. (Hooker 2007) considers three formulation and solution approaches and compares the solution times for each. The work is meant to demonstrate the efficacy of Logic-Based-Benders’ Decomposition and the integration of integer programming and constraint programming to solve this particular type of problem.

(Hooker 2007) uses mixed integer linear programming (MILP) and constraint programming (CP) to solve the PSP. MILP is used to assign tasks or jobs to facilities while CP schedules the jobs at each facility. Logic-Based-Benders’ Decomposition links the master (MILP) and subordinate (CP) problems.
Also, note that although (Hooker 2007) schedules jobs on “machines”, these machines do have the ability to process jobs in parallel subject to a capacity constraint. This is cumulative scheduling.

We present the standard formulation for minimizing cost while meeting all deadlines (tardiness=0). Of the three objective functions considered in (Hooker, 2007), this is the most relevant to the MWP and can be modified to include a tardiness constraint. Sets for the minimum cost PSP include:

$I$ set of jobs,
$K$ set of facilities,
$S$ set of discrete time periods.

Parameters for the minimum cost PSP include:

$c_{ik}$ cost per item to perform job $i$ at facility $k$,
$r_i$ release date of job $i$,
$d_i$ due date of job $i$,
$l_{ik}$ time required to perform job $i$ at facility $k$,
$\mu_{ik}$ resource required per time period to perform job $i$ at facility $k$,
$m_k$ resource available per time period facility $k$.

Parameterized sets for the minimum cost PSP include:

$U_{iks} = \{s' \mid s - l_{ik} < s' \leq s\}$ set time periods during which job $i$ will be processing if the job begins at time $s$ on facility $k$.

Variables:
\( x_{iks} \) binary variables equal to 1 if job \( i \) begins processing at facility \( k \) at time period \( s \) and 0 otherwise.

**Objective (Minimum Cost):**

\[
\min \sum_{i} \sum_{k} \sum_{s} c_{iks} x_{iks}
\]

**Subject to:**

\[
\sum_{k} \sum_{s} x_{iks} = 1 \forall i \in I
\]

(1)

\[
\sum_{i} \sum_{s \in d_{is}} \mu_{iks} x_{iks} \leq m_{k} \forall k \in K, s \in S
\]

(2)

\[
x_{iks} = 0 \forall i \in I, k \in K, s \in S \setminus s < r_{i}
\]

(3)

\[
x_{iks} = 0 \forall i \in I, k \in K, s \in S \setminus s > d_{i} - l_{ik}
\]

(4)

\[
x_{iks} \in \{0,1\} \forall i \in I, k \in K, s \in S
\]

(5)

Constraint set (1) ensures that each job begins processing exactly once at only one facility during one time period. Constraint set (2) ensures that the aggregate resource demands of all jobs processing during a time period at a particular facility do not exceed the facility’s capacity. Constraint set (3) prevents the starting of a job’s processing before its release date. Constraint set (4) ensures that each job will begin processing by a time period that will permit its completion before the job’s due date. Constraint set (5) sets all variables as binary.

(Hooker 2007) uses Logic-Based Benders’ Decomposition approach to add logically derived cuts to bound the master problem. For the minimum-cost objective, a feasible assignment to all machines indicates optimality. For infeasible assignments, one
can rule out the particular job-machine assignments that yield an infeasible (machine-
level) sub-problem. To strengthen the cut(s), Hooker proposes the use of a greedy
algorithm to identify a more specific set of tasks that create the infeasibility on a
particular facility and then the use of this set to generate a Benders’ type cut in the master
problem. (Hooker 2007) also suggests the use of a relaxed sub-problem within the
master. This takes the form of a set of inequalities for each facility that prevents
assignments of jobs that would exhaust all of a facility’s resources for time spans \((t_1 \text{ to } t_2)\)
that contain the time windows of a set of jobs. The master problem formulation for the
minimum cost formulation follows. Sets for the master problem include:

- **H** set of iterations of sub-problem solution.
- **I(*s*₁,*s*₂)** set of jobs where \([r_i, d_i] \subset [s_1, s_2]\),
- **G** complete set of unique **I(*s*₁,*s*₂)**,
- **Iₜₖ** set of jobs assigned to facility *k* during iteration *h*,
- **\(\overline{I}_{hk}\)** set of jobs that creates infeasibility at facility *k* during iteration *h*,
- **Kₜ** set of facilities for which scheduling problem is infeasible at iteration *h*.

Variables for the master problem include:

- \(x_{ik}\) binary variables equal to 1 if job *i* assigned for processing at facility *k* and
0 otherwise.

**Logic Based Benders’ Decomposition Master Problem Objective (Minimum Cost):**

\[
\min \sum_i \sum_k c_{ik} x_{ik}
\]
\[
\sum_i x_{ik} = 1 \forall i \in I \tag{1}
\]
\[
x_{ik} \in B \forall i \in i, k \in K \tag{2}
\]
\[
\frac{1}{m_k} \sum_{i \in I(s_1, s_2)} l_{ik} \mu_{ik} x_{ik} \leq s_2 - s_1 \forall k \in K, I(s_1, s_2) \in G \tag{3}
\]
\[
\sum_{i \in I_{id}} (1 - x_{ik}) \geq \left\lceil \frac{1}{h_k} \right\rceil \forall k \in K_h, h \in H | h \leq |H| - 1 \tag{4}
\]

The computational results in (Hooker 2007) demonstrate that a logic-based Bender’s decomposition approach with a sub-problem formulated using constraint programming is a viable and promising solution method for the PSP. A significant limitation of this work with regard to the MWP is that the size of PSP problem instances are much smaller than the typical size required for an instance of the MWP. For its experimental design, (Hooker 2007) generated instances in a random manner with between 10 and 26 tasks and between 2 and 4 facilities. MWP problem instances may have from 17 to 123,000 items and from 1 to 47 facilities.

Some more recent work on Logic Based Benders’ Decomposition is (Coban and Hooker 2011). This research considers single machine scheduling without cumulative scheduling over a long time horizon. The work considers two variants of the problem. In the segmented version, the job horizon is split into a set of segments, each consisting of some fixed number of time units. Each job must be assigned to a time segment and scheduled within that time segment so that it completes before the end of the assigned time segment. In the unsegmented version, despite the name, the time horizon is still split into a set of segments, however jobs may not overlap onto adjacent segments. For both
problem variants, (Coban and Hooker 2011) have a master problem assign each job to a segment and then a sub-model for each time segment schedules jobs within the time segment. Computational results demonstrate that the Benders approach is more robust than either a pure mixed integer linear program (MILP) or a pure constraint program (CP) approach. For simpler instances, CP was found to be the fastest solution method, however, for more challenging instances, the Benders approach was faster and more likely to achieve a solution within a ten-minute time limit than either MILP or CP.

2.6 Dantzig-Wolfe Reformulation
(Dantzig and Wolfe 1960) suggested that one might reformulate linear programs according to their extreme points. Particularly, (Dantzig and Wolfe 1960) considers problems with block-angular structure such as this problem in original or standard form:

$$\min c_1x_1 + c_2x_2 + ... + c_tx_t$$

Subject to:

$$d_1x_1 + d_2x_2 + ... + d_tx_t = b_0 \quad (1)$$

$$f_i x_i = b_i \forall i \in I \quad (2)$$

$$x_i \geq 0 \forall i \in I = \{1,2,\ldots,t\} \quad (3)$$

The set of constraints defined by (1) are often called coupling constraints.

Assuming the feasible region is a bounded polyhedron, a linear program with this form can be reformulated as a convex combination of the extreme points of the $t$ feasible regions described by the $t$ constraint sets of (2). These feasible regions can be written:

$$P_i = \{x_i \geq 0 \mid f_i x_i = b_i\}$$

letting:
be the extreme points of the t feasible regions.

$$
\min \sum_i \sum_j \lambda_{ij}^j c_j w_{ij}^j
$$

Subject to:

$$
\sum_i \sum_j \lambda_{ij}^j d_j w_{ij}^j = b_0
$$

$$
\sum_j \lambda_{ij}^j = 1 \forall i \in I
$$

Constraint set (1) transforms the coupling constraints. Constraints in set (2) are known as convexity constraints. These force the optimization to take a linear combination of the extreme points.

The reformulation above is known as the extensive formulation (Lubbecke and Desrosiers 2005). The extensive formulation typically reduces the number of constraints in the problem substantially. However, the number of variables has changed and is usually much larger than the original number of variables in the straightforward formulation. Since each extreme point is represented by a variable and since, in general, there are an exponential number of extreme points, the problem may now have a very large number of variables (Bertsimas and Tsitsiklis 1997). However, it is not necessary to include all extreme points in the model. In practice, one solves a Dantzig-Wolfe decomposition problem iteratively and the full set of extreme points is considered only implicitly. Starting with a subset of extreme points that are collectively feasible to the
restricted master problem (RMP), one computes the simplex basis matrix $B$ and associated dual prices $p$.

\[ p' = c'B^{-1} \]

where:

- $p'$: Transpose of vector of dual prices
- $c'$: Transpose of cost vector, $(c_1 \ldots c_t)$

Using the dual prices, sub-problems for each $i \in I = \{1, 2, \ldots, t\}$, are solved to optimality. So, for each $i$, there is a linear sub-problem. Letting $q$ be the vector $p$ without the dual price associated with the convexity constraint and let $r$ be the dual price associated with the convexity constraint (i.e. $p = (qr)$)

\[
\min(c_i - q'd_i)x_i - r
\]

subject to:

\[
x_i \in P_i
\]

(1)

If the solution of a sub-problem is negative, indicating that this new extreme point has the potential to reduce the objective function value of the reformulation (this is a minimization), then a new extreme point is added to the RMP. The column vector of constraint coefficients for this new $\lambda$ (variable representing an extreme point) must also be generated. This is known as delayed column generation and is the central idea behind the Dantzig-Wolfe decomposition (Bertsimas and Tsitsiklis 1997). With the new variables and their columns added to the model, the master is solved until one can prove that any new column would not improve the linear programming solution value; thus, one
proves optimality by proving that no column not already generated could improve the solution value. Thus, whenever columns are added, the linear program is re-solved providing new dual prices. A pricing problem – via optimization – seeks to generate a column whose reduced cost is most negative. If no new column (variable) is found, the LP-relaxation phase of the algorithm terminates. As a restriction of the full linear program, any solution to the RMP is an upper bound on the optimal objective function solution value. Once we have an upper bound on the master problem objective function value, we can identify lower bounds on the master problem objective function value at each iteration. If $z_{\text{RMP}}$ is the integer feasible upper bound for the RMP, $z_{\text{RMP}}^*$ is the (unknown) optimal objective function value of the RMP, and $z_k$ is the optimal objective function value for each pricing problem, then $z_{\text{RMP}} + \sum_k z_k \leq z_{\text{RMP}}^* \leq z_{\text{RMP}}$ (Bertsimas and Tsitsiklas 1997).

2.7 Integer Programming Column Generation

For a linear or continuous model, the sub-problems are themselves linear programs. However, starting with its use for solving a classic problem known as the cutting stock problem in (Gilroy and Gomory 1961) and (Gilroy and Gomory 1963), apply column generation to solving integer and combinatorial problems. Integer and combinatorial optimization problems can be also be decomposed and reformulated using a Dantzig-Wolfe scheme. As for the linear case, delayed column generation is a key aspect of its use, however it is not generally possible to solve the sub-problems using linear programming (i.e. simplex-based) methods. Column-generation has been applied to many important integer and combinatorial problems. A recent survey work on column
generation, (Lubbecke and Desrosiers 2005), identifies well over twenty distinct applications. For example (Desrosiers, Soumis and Desrosiers 1984) applies the method to solve shortest path routing problems with time window constraints. (Hoffman and Padberg 1994) and (Desrosiers and Soumis 1989) apply column generation to different types of crew scheduling.

When using reformulation and column generation for integer and combinatorial problems, algorithms begin as they do for the linear case. Once no column can be found with negative cost, if the master problem satisfies integrality constraints, the solution found is optimal. Otherwise, one must employ a tree-search (enumeration) algorithm. Combinatorial problems rely on binary partitioning of the solution space by fixing a single variable at each level of a search tree. This approach is the well-known branch and bound. Unfortunately, fixing reformulation variables has undesirable effects, often destroying sub-problem structure and leading to unbalanced search trees (Barnhart et.al. 1994). One therefore applies an alternate branching scheme known as “row-branching” and generates negatively priced columns at each node of the search tree. This approach, branching within a column generation algorithm, is known as branch and price. One alternate branching scheme is to branch based on the variables in the standard formulation space. In cases where several sub-problems are very similar, symmetry may lead to very inefficient branching and poor computational performance. Much recent research provides problem-specific branching schemes, including (Vance et.al. 1994) and (Vanderbeck 2000). (Vanderbeck 2011) provides a generic scheme where the sub-problem variables can be bounded.
A work more directly relevant to this research is (Savelsbergh 1997). In this study, the authors apply Dantzig-Wolfe decomposition algorithm to the Generalized Assignment Problem (GAP). From (Savelsbergh 1997), it is reasonable to conclude that extensive reformulation of MWP (or any PSP) should provide bounds that are at least as strong as those provided by the linear program relaxation of the simple formulation (Savelsbergh 1997). Stronger bounds, if available, can be leveraged to drive more efficient branching schemes, hopefully resulting in faster solution time.

A discussion of Dantzig-Wolfe Decomposition would probably be incomplete without mention of the related technique of Lagrangian Relaxation. When using this technique, one dualizes the complicating constraints, or adds a penalty to the objective function for violating the complicating constraints. There are several algorithms tailored to optimize instances of the GAP based on Lagrangian Relaxation within the literature, including (Fisher et.al. 1986), (Guignard and Rosenwein 1989), and (Karabakal et.al. 1992). (Savelsbergh 1997), however, observes worse computational performance for Lagrangian Relaxation based techniques versus column generation techniques despite theoretical equivalence of the Lagrangian Relaxation bound and Dantzig-Wolfe bound. (Savelsbergh 1997) posits that the better convergence properties of the simplex method applied to the extensively formulated master problem vis-à-vis the subgradient and dual ascent methods necessary for solving the Lagrangian Dual are to blame.

2.8 Sifting
A technique, which has come to be known as sifting, is similar to column generation in that variables not in a restricted master problem are priced to determine
those that should be added to the full problem. We found a good discussion of this technique in (Bixby et.al. 1991). This work attributes the idea to (Forrest 1989), however it is likely that the idea is older. (Crowder and Padberg 1980) use subsets of the arcs in the Travelling Salesman Problem (TSP) and use reduced costs and variable fixing to establish lower bounds on their problem instances. The idea of sifting is to start a problem with a set of variables that can give a feasible solution to the full linear program. (Bixby et.al 1991) identifies this set as the “working set”. Then, using optimal dual variable values for restricted problem, one prices each variable of the full problem, that is not in the working set. Any variable with a negative price can potentially reduce the objective function value of the restricted problem. One adds each of these negatively priced variables to a candidate list. Once each variable in the full variable set but not in the working set has been priced, some of those in the candidate list (i.e. those that priced negatively) are added to the working variables. One approach is to add some number of the most negatively priced variables. (Bixby et.al. 1991) suggests a related approach called lambda pricing.

After adding the selected candidate list variables, one resolves the restricted master problem to obtain new dual prices. One then repeats the process by again pricing each variable not in the current working set again. Once one can demonstrate that there are no variables outside of the working set that have negative prices, the optimal objective function value of the restricted problem is the same as the optimal objective function value of the full sized problem.
Modern commercial solvers, such as Gurobi, have the capability to perform sifting. However, in order to use this capability one must first load the full variable set into the model and memory.

2.9 Relationship to the Literature
This research considers the same data sets considered in (Kotkin et.al. 2011) and also will use an implementation comparable to the SWF formulation as one means of solving instances of the MWP. The second, decomposition based implementation will build on the minimum cost planning and scheduling formulation developed in (Hooker 2007). Problem instances considered in this research do not occupy the experimental design space considered by (Hooker 2007) and are also based on real rather than randomly generated data. For this reason, it will be necessary to develop new logic-based cuts to support a workable decomposition based implementation. The method devised in (Hooker 2007) will also require modification to permit system tardiness for some instances. Implementation is somewhat less straightforward when individual facilities are allowed tardiness that must, in aggregate for the system, be less than some specified amount. As mentioned previously, the efficacy of Logic Based Benders’ Decomposition has been questioned in (Heinz and Beck 2012) while (Cire, Coban and Hooker 2013) disputes the findings of (Heinz and Beck 2012). This research will consider data not considered by either paper and provide independent and relevant evidence.

The final column generation based implementation will build on the work in column generation including (Savelsburgh 1997) and (Hoffman and Padberg 1993). The
use of a column generation approach for a PSP as represented by the MWP will represent a novel approach to a PSP as far as can be determined.

This research will also build upon that presented in (Kotkin et.al. 2011) in the representation of the solution space as an efficient frontier rather than a single value. Elsewhere in the literature, such as (Hooker 2007), the PSP is a minimum cost problem with fixed due dates or a minimum tardiness problem without consideration given to costs. While (Kotkin et.al. 2011) suggests a weighted objective function, the approach anticipated here is closer to that of preemptive optimization (i.e. minimizing cost for a fixed level of acceptable tardiness)
In this section, we consider alternative formulations of the Maintenance Workload Problem (MWP). The original work, (Kotkin et. al 2011) was an effort by Army Materiel Systems Analysis Activity (AMSAA) to assign items to repair facilities and schedule the items at each repair facility. Due to the very large size of some instances of the MWP, the prior work (Kotkin et. al. 2011), grouped items requiring similar repairs and with similar origin and destination into item-batches in a preprocessing step in order to reduce the total number of binary variables in their optimization model. Our formulations organize items into similar groups, which we identify as item arrival groups (IAGs). Unlike batches, the items in IAGs are not implicitly constrained to be assigned and scheduled together. This means that our formulations do not require that items within an IAG process at the same facility nor that all items within the IAG start at the same time. We also take a different approach than (Kotkin et.al. 2011) to the multi-objective aspect of the MWP. Our objective function considers only the cost of item transport and assignment. Each model constrains tardiness to a maximum specified amount. (Kotkin et.al. 2011) on the other hand, minimizes a weighted sum of cost and tardiness. We feel that the simplified objective function is superior to the weighted sum function in that unlike units (cost and tardiness) are not combined. Finally, we do not generalize the model to permit multiple resource types to constrain facility processing. We consider a
single resource, direct labor hours, at each time period at each facility. Although (Kotkin et.al. 2011) includes set $W$ to permit this generalization, it was not used in the prior work and we have no data to support its use.

Our first new formulation, Standard Formulation (SF) in section 3.1.1, revises the “Sliding Windows Formulation” (SWF) found in (Kotkin et.al. 2011) to minimize repair cost for full-sized instances of the MWP. Although we organize the items for repair into IAGs very much like the item-batches created for SWF, we permit the model to select the number of items from each IAG to assign and schedule using integer variables. This increases the model’s state space, but not necessarily the number of variables; in most cases, we were actually able to reduce the number of variables. In section 3.1.2, we define a restriction of the MWP, differing from SWF, which we identify as restricted standard formulation (RSF). Using a subset of variables from the full problem, we can solve the RSF to identify an upper bound (i.e. a feasible solution). These bounds prove useful for large problems and can be used in conjunction with a method that gives a valid lower bound on the MWP to give solutions of known quality.

Our third formulation, Benders Decomposition Formulation (BDF) described in sections 3.2.1-3.2.5, is the first of two formulations employing a decomposed structure that we solve iteratively. BDF consists of a master problem to assign items to facilities and multiple sub-problems to model the scheduling of item repairs at each facility. BDF gives a lower bound for the MWP at each iteration. Once the model arrives at a feasible assignment of all items, optimality is proven. We also outline a method to employ BDF
in conjunction with RSF to obtain solutions of known quality for all experimental instances in section 3.2.6.

Our fourth formulation described in section 3.3 employs a Dantzig-Wolfe reformulation in order to use a column generation approach. This formulation is also solved iteratively and consists of a restricted master problem (RMP) and a set of facility pricing problems. In brief, we solve the linear program relaxation (LPR) of the RMP to obtain dual price information. We then use the facility pricing problems to identify new variables (columns) for inclusion in the RMP that have the potential to reduce the LPR objective function value. The algorithm generates an upper bound when we can identify an integer feasible solution to the RMP. The formulation yields a lower bound when there are no variables to add to the RMP that can possibly reduce the LPR.

Finally, after noting some success with the restricted standard formulation (RSF), we give a formulation that implements a sifting approach in section 3.5. We start by identifying a feasible solution (i.e. upper bound) using the RSF and then transition to pricing each variable not originally included in the RSF. Columns that price favorably (negatively) are added to the model. We can use this method to identify both a lower bound and potentially improve our initial solution.

3.1 Standard Formulations

We formulate two models in what we are calling standard formulation (SF). Other authors (e.g. Barnhart et.al. 1997) have identified the “basic” formulation as compact formulation. Standard formulation for an integer program is a complete and contained mathematical formulation that might be sent to a solver and optimized by means of the
linear programming simplex method and an enumeration algorithm (e.g. branch and bound). We note that modern commercial grade solvers include a number of enhancements that often speed solution, such as heuristics and cutting planes that give bounds and reduce the work that the enumerative algorithm must perform.

3.1.1 Full Standard Formulation (SF)

Our first formulation is a complete representation of the problem in a single model. Our model seeks to assign items for repair to facilities at minimum cost. The model must also schedule the items on the facilities to which they are assigned within some total system tardiness constraint for all jobs. We implemented two important modifications that permit the consideration of full-sized instances without increasing the number of variables. We define integer variables rather than binary variables for each IAG rather than item-batches. This permits items within each IAG to start independently of one another and even at different repair facilities. In addition to the change of variable type, this change requires some modification of the constraints. The work in (Kotkin et. al 2011) and most other schedule optimization research that we have reviewed uses binary variables to represent jobs or items for processing. Accordingly, in early versions, we used a binary variable to represent each item of each IAG, however the number of variables for moderate sized instances proved too large for our models to be able to load into memory, thereby preventing solution. The change to integer representation proved crucial for solving large instances. The change to integer variables requires some care when implementing constraints that will be indicated in the formulation. We carefully
designed the formulation to create as few variables as possible using sets. We define the following sets for model SF:

- **I**: set of IAGs (Item Arrival Groups) for repair where each IAG consists of one or more similar equipment items for repair with identical origin, destination, release date and due date,
- **K**: set of repair facilities,
- **I_k**: set of IAGs whose items may be scheduled on facility $k$ (i.e. facility $k$ has the capability to repair items of IAGs in this set),
- **K_i**: set of facilities with the capability to perform the repair of items in IAG $i$,
- **S**: set of time periods.

We define the following parameters for model SF:

- $a_i$: item quantity (amount) associated with IAG $i$,
- $\sigma^1_{ik}$: cost per item to ship a carcass (un-repaired item) from IAG $i$ to facility $k$,
- $\sigma^2_{ik}$: cost per item to ship a repaired item from IAG $i$ to facility $k$,
- $p_{ik}$: cost per item to perform repair an item in IAG $i$ at facility $k$,
- $r_i$: release date of IAG $i$ at its origin,
- $d_i$: Due date of IAG $i$ at its destination,
- $l_{ik}$: Time required to perform repair of a single item in IAG $i$ at facility $k$,
- $\tau^1_{ik}$: Time to ship an item as carcass from IAG $i$ to facility $k$,
- $\tau^2_{ik}$: Time to ship a finished item from IAG $i$ to facility $k$,
Resource (labor hours) available at facility $k$ during time period $s$,

Amount of resource (labor hours) an item in IAG $i$ requires during each time period of its repair time ($l_{ik}$) if it is repaired at facility $k$,

Maximum total tardiness for the instance,

Total cost to transport and repair an item in IAG $i$ at facility $k$.

We define the following parameterized sets for model SF:

$U_{iks}$ set of time periods during which an item in item-group $i$ that began repair at time $s$ at facility $k$ will still be processing;

$S_{iks}$ set of time periods that items from item-group $i$ may feasibly start repair at facility $k$; $S_{iks} = \{s \mid s \geq r_i + \tau_{ik}^1\} \forall i \in I, \forall k \in K$,;

$T_{iks}$ set of time periods where starting items from item-group $i$ at facility $k$ will result in one or more units of tardiness (this will be a subset of $S_{iks}$);

We define the following variables for model SF:

$x_{iks}$ positive integer variables that give the number of items of IAG $i$ that begin processing at facility $k$ at time period $s$.

**Objective:**
\[
\min \sum_{I} \sum_{K} \sum_{S} c_{ik}x_{iks}
\]

Subject to:

\[
\sum_{K} \sum_{S} x_{iks} = a_i \forall i \in I
\]  \hspace{1cm} (1)

\[
\sum_{I} \sum_{K} \sum_{T_{ik}} \sum_{s} (s + l_{ik} + \tau_{ik}^2 - d_i)x_{iks} \leq b
\]  \hspace{1cm} (2)

\[
\sum_{I} \sum_{K} \sum_{T_{ik}} \sum_{s \in J_{iks}} \mu_{ik}x_{iks} \leq m_k \forall k \in K, s \in S_{ik}
\]  \hspace{1cm} (3)

\[
x_{iks} \in \mathbb{Z}^+ \forall i \in I, k \in K, s \in S_{ik}
\]  \hspace{1cm} (4)

Constraint set (1) requires that each item in an IAG be scheduled. Constraint set (2) computes the total tardiness of all items and constrains it to be less than the maximum allowed tardiness. For each IAG, we include integer IAG variables only for time periods that would result in tardiness. For this integer formulation, where some IAG items might accrue tardiness while others might not, we must adopt the constraint set described by (2). Constraint set (3) enforces the labor resource constraint for each time period at each facility by “looking back” to determine if IAG items have begun and are within their active period of processing defined by \( l_{ik} \). Constraint set (4) defines integer variables on the appropriate sets. Defining the variables on set \( K \) creates \( x_{iks} \) variables only where a facility has the capability to perform the necessary repairs. Defining the variables on set \( S_{ik} \) creates \( x_{iks} \) variables only where the time period \( s \) equals or exceeds the IAG’s release time and transportation time to the repair facility \( k \).
3.1.2 Restricted Standard Formulation

The full standard formulation can result in the creation of a very large number of variables. We must create an integer variable for each time period that items within an IAG might start at each facility that they might start. For our largest instances, this exceeds seventy-million variables. The vast majority of these variables will have a value of zero in any feasible solution. Even with the change to integer state-space, we found it difficult to create models of the required size within a reasonable timeframe. We found that we could instead create a restricted model that includes a subset of our decision variables to identify upper bounds on the objective function value. The lower bounds identified by these models are not valid for the full-sized problem. However, any feasible solution found is a proper upper bound on the full-sized problem’s objective function value. A useful restricted formulation we found was to create variables only for some set of least expensive facilities and also for some set of fastest facilities for each IAG. So, for each IAG, we evaluate the cost to perform repair at each facility $c_{ik} = \sigma_{ik} + p_{ik} + \sigma_{ik}$.

Sorting on this value, we identify the least expensive $n$ facilities. For each job, call this set of least expensive facilities $E_i(n)$. We also consider the necessary transportation time and duration of each repair operation at each facility, $\tau_{ik} + l_{ik} + \tau_{ik}$, we identify the $p$ ‘fastest’ facilities for each job and include these facilities in set $F_i(p)$. Then, for each IAG, we create variables for only the union of these sets, $K_i(n,p) = E_i(n) \cup F_i(p)$. So, (4) takes the form $x_{iks} \in Z^+ \forall i \in I, k \in K_i(n,p), s \in S_{ik}$ for the RSF.
3.2 Benders’ Decomposition
(Hooker 2007) and subsequent research has shown that a technique called logic-based Benders Decomposition (LBBD) is an effective solution technique for certain minimum cost planning and scheduling problems (PSP-MC). The MWP effectively generalizes the PSP by minimizing cost while permitting some amount of tardiness. To determine whether this form of decomposition might prove useful in identifying or quantifying the quality of solutions to the MWP, we formulated the MWP according to a Benders Decomposition approach. We identify this formulation as the Benders Decomposition Formulation (BDF). We have a master, or coordinating model that seeks to minimize the cost of assigning all IAG items to a set of facilities. We then have a set of sub-models, representing facilities, which seek to schedule the jobs that the master has assigned according to some objective function. We solve the problem iteratively. First we solve the master to obtain variable values to model facilities. Next, using these variable values, we solve sub-problems for each facility. Based on the results of our facility modeling, we may formulate constraints to place into the master problem to help guide its search for an optimal and feasible solution.

3.2.1 An Initial Benders’ Master Problem
For development and exposition, we provide first an initial master problem. The initial master problem is similar to the generalized assignment problem (GAP) discussed in chapter 2. We define the following sets and indices for the BDF master problem (BDF-M):
set of IAGs for repair where each IAG consists of one or more similar equipment items for repair with identical origin, destination, release date and due date,

$I_k$ set of IAGs where items may be feasibly assigned to facility $k$ (i.e. facility $k$ has the capability to repair the items in IAGs in $I_k$),

$Q_i$ set of items in IAG $i$,

$K$ set of repair facilities,

$K_i$ set of facilities with the capability to perform the repair of items in IAG $i$,

$H$ set of iterations of master problem solution,

$J_{kh}$ set of items assigned to facility $k$ at iteration $h$ where set members take the form $(i,q)$, with $i \in I$ and $q \in Q_i$.

We define the following parameters for BDF-M:

$\sigma_{ik}^1$ cost per item to ship a carcass (un-repaired item) from IAG $i$ from its origin location to facility $k$,

$\sigma_{ik}^2$ cost to ship a repaired item from IAG $i$ from its origin location to facility $k$,

$p_{ik}$ cost per item to repair an item from IAG $i$ at facility $k$,

$b$ maximum total tardiness (tardiness budget) for the instance,

$c_{ik}$ total cost to transport and repair an item in IAG $i$ at facility $k$

$$c_{ik} = \sigma_{ik}^1 + p_{ik} + \sigma_{ik}^2.$$
We define the following variables for BDF-M:

- $x_{iqk}$: binary variables equal to 1 if item $q$ from IAG $i$ is assigned to facility $k$ for repair and 0 otherwise,

- $f_k$: positive continuous variables representing the amount of total permitted tardiness allocated to facility $k$ (e.g. the items assigned to facility $k$ may accrue no more than this amount of tardiness).

**Objective:**

$$\min \sum_k \sum_{i} \sum_{Q_i} c_{ik} x_{iqk}$$

**Formulation (Subject to):**

$$\sum_k x_{iqk} = 1 \forall i \in I, q \in Q_i$$  \hspace{1cm} (1)

$$\sum_k f_k \leq b$$  \hspace{1cm} (2)

$$x_{ik} \in \{0,1\} \quad \forall k \in K, i \in I_k, q \in Q_i$$  \hspace{1cm} (3)

Constraint set (1) requires that each item in each IAG be assigned to a facility for repair.

Constraint (2) requires that the total tardiness allocated to all facilities be less than the total allowed tardiness. Constraint set (3) defines a binary variable for each item in each IAG and for each facility that has the capability to repair items in the IAG.

Solving the master problem formulated above will assign each item to a facility with the capability (but perhaps not the capacity) to repair the item at minimum cost. The master objective function value (or more exactly, its lower bound) is a valid lower bound on the objective function value of the MWP. The master problem will also allocate
tardiness to facilities, although the initial allocation will be arbitrary since there is
initially no information to guide this decision. Given the assignments determined by the
master and the tardiness allocated to each facility, each facility may or may not be able to
actually perform all the work assigned to it given its labor available and allowed
tardiness.

3.2.2 The Benders’ Facility Sub-Problem
To evaluate each facility’s capacity to perform its assigned set of item repairs, we
define a sub-problem for each facility in $K$. We identify these as Benders decomposition
formulation-sub-problems (BDF-S). We define the following sets and indices for BDF-S.

- $H$: set of iterations of master problem solution,
- $J_{kh}$: set of items assigned for repair to facility $k$ at iteration $h$. Items in this
  set have the form $(i, q)$ where $i$ identifies the item’s IAG and $q$
  identifies the particular item within the IAG,
- $S$: set of time periods,

We define the following data elements for BDF-S:

- $r_i$: release date of all items in IAG $i$ at its origin,
- $d_i$: due date of all items in IAG $i$ at its destination,
- $l_{ik}$: time required to repair an item in IAG $i$ at facility $k$,
- $\tau_{ik}^1$: time to ship an item from IAG $i$ as a carcass from its origin to facility $k$,
- $\tau_{ik}^2$: time to ship a finished item from IAG $i$ from facility $k$ to its destination,
- $m_{ks}$: resource (labor hours) available at facility $k$ during time period $s$,
\( \mu_{ik} \) amount of resource (labor hours) each item in IAG \( i \) requires during each
time period of its repair time \( (l_{ik}) \) if it is repaired at facility \( k \),

\( f_h \) amount of tardiness allocated to facility \( k \) at iteration \( h \).

We define the following parameterized sets:

\( U_{iks} \) set of time periods during which an item in item-group \( i \) that began repair
at time \( s \) at facility \( k \) will still be processing;

\[ U_{iks} = \{s' \mid s - l_{ik} < s' \leq s\} \forall i \in I, \forall k \in K, \]

\( S_{ik} \) set of time periods that items from item-group \( i \) may feasibly start repair
at facility \( k \); \( S_{ik} = \{s \mid s \geq r_i + \tau_{ik}\} \forall i \in I, \forall k \in K, \)

\( T_{ik} \) set of time periods where starting items from item-group \( i \) at facility \( k \)
will result in one or more units of tardiness;

\[ T_{ik} = \{s \mid s \geq 1 + d_i - l_{ik} - \tau_{ik}^2\} \forall i \in I, \forall k \in K. \]

Variables:

\( y_{isq} \) binary variables equal to 1 if item \( q \) from IAG \( i \) begins repair processing at
the beginning of time period \( s \) and 0 otherwise,

\( t_{sq} \) positive continuous variables representing the tardiness due to item \( q \) in
IAG \( i \).

Objective:

\[ Z_{SP(k,h)} = \max \sum_{J_{is}} \sum_s y_{isq} \]
Subject to:

\[ \sum_{s} y_{iqs} \leq 1 \forall (i,q) \in J_{kh} \] (1)

\[ \sum_{J_{ik}} \sum_{s \in U_{ik}} \mu_{ik} y_{iqs} \leq m_{ks} \forall s \in S \] (2)

\[ t_{iq} = \sum_{T_{iq}} (s + l_{ik} + \tau_{ik}^{2} - d_{ij}) y_{iqs} \forall (i,q) \in J_{kh} \] (3)

\[ \sum_{J_{ik}} t_{iq} \leq f_{k} \] (4)

\[ y_{iqs} \in \{0,1\} \forall (i,q) \in J_{kh}, s \in S_{ik} \] (5)

\[ t_{iq} \in \mathbb{R}^{+} \forall (i,q) \in J_{kh} \] (6)

Constraint set (1) permits each assigned item to be repaired at most once. Constraint set (2) requires that all items processing during a time unit not exceed the facility’s labor available. Constraint set (3) sets the tardiness for each item assigned to the facility. Constraint (4) requires that the total tardiness from all assigned items be less that the amount of tardiness allocated to the facility. Constraint set (5) defines binary variables for each IAG item assigned to the facility and for each time period during which it might begin repair. We define these variables only for time periods on and after an item’s release date plus its transportation time to the facility represented. Constraint set (6) defines all tardiness variables as positive real numbers.

At the outset, we know neither the optimal assignment of items to facilities (for repair), nor an optimal tardiness allocation to facilities in order to accomplish this. We must attempt to discover both. We do this by using the linear programming relaxation (LPR) of the sub-problems. The LPR is formulated exactly as the above sub-problem;
with the exception that each $y_{qs}$ can take on any value on the closed interval [0,1]. For a given allocation of tardiness, $f_k \forall k \in K$, solving the LPR will give an upper bound on the number of items within a given assignment set, $J_{kh}$, that may be feasibly repaired at the facility. The LPR has the advantage that it can be solved using the simplex algorithm without resorting to branch-and-bound enumeration as binary and integer programs often require. Solving the LPR will ordinarily be much faster than solving the associated integer or binary program. Another important factor for our purposes is that we can extract dual information from the solution of a linear program that is not available when solving a binary or integer program.

### 3.2.3 Benders’ Cuts

Assume now that we solve the master problem yielding both a tardiness allocation and a set of items assigned to each facility. It is not yet known whether each facility can accomplish its given assignment within its tardiness allocation (i.e. budget). We solve the LPR of the sub-problem representing the first facility and find that the facility can perform all assigned repairs (on assigned items) without violating its tardiness budget. In this case, nothing need be done and we consider the next facility. We solve the LPR representing this facility and find that the facility is unable to perform all assigned repairs within its allocated tardiness budget. This means that $Z_{SP(k,h)-LPR}$, the objective function value of the sub-problem for facility $k$ at iteration $h$ is less than $|J_{kh}|$, the number of jobs assigned to facility $k$ for repair. If our tardiness allocation were fixed (it is not), we might formulate the following cut for the master problem.
This cut indicates that of the jobs in the assignment to facility $k$, no more than the number successfully scheduled may be assigned in subsequent assignments (or iterations of master problem solution). As it is not the case that our tardiness allocation is fixed, we can make use of the dual price of tardiness from the sub-problem’s LPR. If $\rho$ is the dual price of constraint (4) in the sub-problem, this indicates the number of additional items in the assignment (or fractional items) that the facility could repair given an additional unit of tardiness in its budget $f$. We can use this to formulate a new cut for the master problem.

$$\sum_{(i,q) \in J_{kh}} x_{iqk} \leq Z_{SP(k,h)} - LPR + \rho f_k$$

In the master problem, this constraint gives an upper bound on the maximum number of items from a particular assignment that can be repaired at a particular facility. Given an allocation of tardiness to the facility, it increases the constraint’s right hand side and permits more items to be assigned to the facility. Unfortunately, it is not valid to adjust the constraint to account for the amount of tardiness already allocated to the facility due the non-linear properties of the scheduling sub-problem. We also note that it is sometimes the case that a valid constraint using the tardiness dual price will not cut off the present master problem solution, as is required for continuation of the Benders algorithm. If the number of items successfully scheduled added to the product of the dual price and the tardiness presently allocated exceeds the number of jobs in the assignment, we may choose to not add the constraint to the master problem.
Unlike the standard formulation, we used binary variables for the Benders Decomposition formulation, each binary variable representing an individual item of equipment within an IAG. We attempted to modify the Benders formulation to use integer variables to represent the number of items from a particular IAG rather than represent each item individually. While both BDF-M and BDF-S could easily be formulated using integer variables, we were unable to create a scheme to generate valid communicating cuts that also (usually) cut off infeasible assignments.

3.2.4 Final Benders’ Master Problem
In this section, we give a second description of the master problem with cuts from the sub-problems.

We define the following sets and indices for BDF-M:

\( I \) set of IAGs for repair where each IAG consists of one or more similar equipment items for repair with identical origin, destination, release date and due date,

\( I_k \) set of IAGs where items may be feasibly assigned to facility \( k \) (i.e. facility \( k \) has the capability to repair the items in IAGs in \( I_k \)),

\( Q_i \) set of items in IAG \( i \),

\( K \) set of repair facilities,

\( H \) iterations of master problem solution,

\( J_{kh} \) set of items assigned to facility \( k \) at iteration \( h \); set members are unique items identified as \((i, q)\),
\( \overline{J}_{kh} \) set of items assigned to facility \( k \) at iteration \( h \) and \( Z_{SP(h,k)\rightarrow LPR} < \vert \overline{J}_{kh} \vert \).

We define the following data elements for BDF-M.

\( \sigma_{ik}^1 \) cost per item to ship a carcass (un-repaired item) from IAG \( i \) from its origin location to facility \( k \),

\( \sigma_{ik}^2 \) cost to ship a repaired item from IAG \( i \) from its origin location to facility \( k \),

\( p_{ik} \) cost per item to repair an item from IAG \( i \) at facility \( k \),

\( T_{\text{max}} \) maximum total tardiness for the instance.

\( c_{ik} \) total cost to transport and repair an item in IAG \( i \) at facility \( k \)

\[
    c_{ik} = \sigma_{ik}^1 + p_{ik} + \sigma_{ik}^2.
\]

\( \rho_{kh} \) Dual price of sub-problem tardiness constraint from sub-problem \( k \) at iteration \( h \).

**Variables:**

\( x_{iqk} \) binary variables equal to 1 if item \( q \) from IAG \( i \) is assigned to facility \( k \) for repair,

\( f_k \) positive continuous variables representing the amount of total permitted tardiness allocated to facility \( k \).

**Objective:**

\[
    \min \sum_{k} \sum_{i} \sum_{q} c_{ik} x_{iqk}
\]
Formulation (Subject to):

\[ \sum_{k} x_{iqk} = 1 \forall i \in I, q \in Q \quad (1) \]

\[ \sum_{k} f_k \leq T_{max} \quad (2) \]

\[ x_{ik} \in \{0,1\} \forall k \in K, i \in I_k, q \in Q \quad (3) \]

\[ \sum_{(i,q) \in F_{in}} x_{iqk} \leq Z_{SP(k,h)-LPR} + \rho_{kh} f_k \quad (4) \]

Constraint set (1) requires that each item in each IAG be assigned to a facility for repair. Constraint (2) requires that the total tardiness allocated to all facilities be less than the total allowed tardiness. Constraint set (3) defines a binary variable for each item in each IAG and for each facility that has the capability to repair items in the IAG. Constraint set (4) identifies cuts that are added by each sub-problem when the objective function of its LPR is less than the number of items assigned to it.

There are certain constraints we may include in the master problem before we begin a Benders iterative algorithm that can help the model to converge more quickly by eliminating certain assignments. A significant complication introduced by splitting IAGs into individual items represented by binary variables is symmetry. Every item in an IAG will have identical costs and processing requirements on the same facility. When using our given cutting scheme, or a “no-good” cutting scheme, as in (Hooker 2007), we may be forced to rule out many essentially identical solutions. Consider a simple case where items \( x_1, ..., x_5 \) come from the same IAG and so, have identical costs and processing...
requirements at each facility. If, for example, we assign $x_1, x_2, x_3$ to a facility $a$ at some iteration, and find that this assignment is infeasible, we could deliver the cut 

$$x_{1a} + x_{2a} + x_{3a} \leq 2$$

to the master problem. Since our IAG items are symmetric, the next assignment to facility $k$ will almost certainly include some combination of jobs from $x_1, ..., x_5$ other than the one just ruled out (e.g. $x_1, x_2, x_3$). In all likelihood, our algorithm will be forced to create cuts for all combinations of three from the set $x_1, ..., x_5$ in order to rule out all necessary combinations. Obviously, the problems with symmetry worsen with increasing IAG sizes. As a partial solution, we can include symmetry breaking constraints in both our master and sub-problem formulations. For the master problem, these take the following form, $x_{i p k} \leq x_{i (q-1) k}$. This indicates that an item will not be repaired at a facility unless the preceding item in the item’s IAG is repaired at the facility. Obviously we would not define these for every facility. This would require that all items be assigned to facilities as complete IAGs. The approach we take is to generate these constraints only for each IAG’s lowest cost facility. For each item group, we can define these types of constraints for two facilities. We can require that one facility be loaded in order of increasing $q$ as per the constraint above. We can also force one facility to be loaded in order of decreasing $q$. These constraints are of the form, $x_{i p k} \geq x_{i (q-1) k}$. We create these constraints for the two least expensive facilities. We note that this does not completely resolve problems that may arise due to symmetry. Other more comprehensive symmetry breaking strategies are complicated significantly by the non-linear character of the scheduling problem.
An often-cited use of decomposition as an optimization technique is to expose problem structure. Indeed, decomposing the MWP opens a number of other avenues to create new constraints to help guide the master to an optimal solution. Although we did not find these approaches helpful for all instances, we note several options here. For instance, using our facility sub-problems, we can solve a single linear program relaxation for each facility to obtain an upper bound on the maximum number of jobs that can be scheduled out of the set of all possible jobs that a facility might be assigned. If we are willing to solve several linear program relaxations for each facility, it is possible to adapt the windowing strategy from (Hooker 2007) to the MWP to identify an upper bound for the maximum number of jobs that a facility might complete for various arriving groups of items. Finally, in certain cases, we might obtain a tighter upper bound on the number of jobs a facility might successfully schedule by solving the (non-relaxed) integer program of the facility sub-problem for an identified bottleneck facility, maximizing the number of scheduled jobs of all possible jobs. The last option tends to be computationally intensive, and so would probably only make sense for certain difficult cases. Each upper bound obtained, whether from a linear program relaxation or from an integer program can be used as a constraint to guide the master problem to a solution of the MWP. In some cases, the effort required to implement these approaches was not economical, meaning that overall solution time was adversely affected.

3.2.5 Benders’ Algorithm Upper Bounds

In our described algorithm, the relaxed sub-problem solution may not actually give a binary/integer feasible solution to the sub-problem, the relaxation gives only a
bound on the number of items of a collection that can be feasibly scheduled. Accordingly, we may terminate our algorithm once we have finished generating cuts on the LPR sub-problems. In this case, we may only have a lower bound on the optimal objective function value for the MWP. If we do not have an integer/binary feasible solution for each facility, we can elect to continue searching for such a solution and obtain an upper bound on the objective function value for the MWP. Perhaps the easiest way to continue the search is to fix the tardiness allocation at its final value (as we will be able to generate no further information to inform reallocation) and then deliver cuts to the master problem from a non-relaxed (integer/binary) sub-problem. These will not provide duality information and will only provide a fixed upper bound for the number of jobs from a given assignment that may be successfully repaired given the fixed tardiness allocation. Cuts for the master problem in the search phase may take the following form.

$$\sum_{(i,q) \in J_{ik}} x_{iqk} \leq Z_{SP(h,k)}$$

The tardiness allocation obtained at the conclusion of LPR based cutting is not guaranteed to give integer feasible sub-problem solutions and so these cuts may (improperly) remove feasible (and potentially optimal) solutions from the search region. We can use these only to carry out a search for a feasible solution (upper bound) and may not use the master problem objective function value (or its bound) as a proper lower bound on the MWP once we have fixed the tardiness allocation.

**3.2.6 A Hybrid Benders’ Decomposition Method**

We’ve hinted at a significant drawback of the Benders approach. We do not obtain an upper bound (feasible solution) until we achieve optimality. In practice, this can
lead to very long solution times. But, we do obtain a lower bound at each iteration of master problem solution. In our discussion of the restricted standard formulation (3.1.2), we noted that the method does not give any useable lower bound, just upper bounding feasible solutions. An obvious hybrid approach is to use the two methods together for challenging, large instances of the MWP. We can obtain an upper bound using the RSF and using this bound in our Benders’ algorithm terminating upon achieving a particular optimality gap or time limit. A drawback to this approach is that once we have identified a feasible solution using the RSF, we do not continue search, by inclusion of more variables, for example.

3.3 Column Generation Formulation
Dantzig-Wolfe reformulation with column generation decomposes the MWP into a restricted master problem (RMP) and a set of pricing problems, in this case, one per facility. The RMP considers binary variables that represent extreme points of the feasible space. For the MWP, we have integer variables representing the assignment of some number of items from each IAG to a particular facility. The cost associated with a column (i.e. to give it a binary value of one) is the cost of both transporting the items and repairing all of the items in the column at the given facility. To use the column generation algorithm, we then create the linear programming relaxation of the RMP (RMP-LPR) by allowing binary variables to take on any value on the interval [0,1]. This is necessary so that we can obtain dual prices to inform column generation at the facility sub-models.

We cannot reasonably include a variable representing every feasible schedule in the RMP, so we start column generation with a set of variables that will give a feasible
solution to the RMP-LPR. We can obtain starting variables using heuristics or interim solutions obtained from other optimization methods. Another option is to include a variable with a large cost that performs all items within each IAG, as if there were a high cost facility with infinite capacity.

To execute the algorithm, we first solve the RMP-LPR to obtain dual prices for each constraint in the RMP. We can then use these dual prices to identify new assignments (i.e. variables) to include in the master by solving the pricing problem associated with each facility informed by the dual prices. The first phase of the algorithm terminates when we can show that there are no remaining columns that could improve the objective function of the RMP-LPR. During column generation, any solution to the RMP-LPR that is integer is a feasible solution to the RMP and also an upper bound on the value of the RMP objective function value. Once we show that no remaining columns could improve the objective function, the optimal value of the RMP-LPR objective function represents a lower bound on the value of the RMP objective function.

Once we have shown that no further column could possibly improve the master problem objective function, if the RMP-LPR yields a non-integer solution, we branch based on sub-problem (IAG item quantity) variables. We can enforce these branches through constraints in the sub-problem. At each node of the resulting branching tree, we must attempt to generate columns (assignments) that have the potential to reduce the objective function.
3.3.1 Column Generation Master Problem (RMP)

We define the following sets and indices for the first column generation formulation restricted master problem (RMP)

$I$ set of IAGs of items for repair where each IAG consists of one or more similar equipment items for repair with identical origin, destination, release date and due date,

$K$ set of repair facilities,

$H$ set of iterations of restricted master problem solution,

$M_k$ set of feasible assignments for facility $k$.

We define the following data elements for the RMP

$a_i$ item quantity associated with IAG $i$,

$\sigma_{ik}^1$ cost per item to ship a carcass (un-repaired item) from IAG $i$ to facility $k$,

$\sigma_{ik}^2$ cost per item to ship a repaired item from IAG $i$ to facility $k$,

$p_{ik}$ cost per item to perform repair of an item in IAG $i$ at facility $k$,

$b$ maximum total tardiness for the instance,

$t_{km}$ total tardiness incurred by assigning jobs contained in assignment $m$ to facility $k$,

$g_{im}$ number of items from IAG $i$ in assignment $m$ for facility $k$,

$c_{km}$ cost to execute assignment $m$ on facility $k$

\[ c_{km} = \sum_i (\sigma_{ik}^1 + p_{ik} + \sigma_{ik}^2) g_{im}, \]

Variables:
$x_{km}$  binary variables equal to 1 if feasible assignment $m$ is selected for facility $k$.  

**Objective:**

$$z_{RMP} = \min \sum_{k} \sum_{M_k} c_{km} x_{km}$$

**Subject to:**

$$\sum_{k} \sum_{M_k} g_{ikm} x_{km} \geq a_i \forall i \in I$$  \hspace{1cm} (1)

$$\sum_{k} \sum_{M_k} t_{km} x_{km} \leq b$$ \hspace{1cm} (2)

$$\sum_{M_k} x_{km} = 1 \forall k \in K$$  \hspace{1cm} (3)

$$x_{km} \in \{0,1\} \forall k \in K, m \in M_k$$ \hspace{1cm} (4)

Constraints in set (1) require that the selected variables (i.e. facility assignments) account for each item in each IAG. The single constraint (2) requires that selected assignments have less than the total allowed tardiness. Constraints in set (3) are known as convexity constraints. These require that each facility have exactly one assignment. Constraints in set (4) define binary variables.

**3.3.2 Column Generation Pricing Problem**

The role of the pricing problem is to give new columns that have the potential to reduce the objective function value of the relaxed master problem. If the minimization of the pricing problem gives a negative value, a new column is generated. If, for example, we are generating a column for the $p$-th assignment to the $k$-th facility, the new variable
in the master problem will be \( x_{kp} \). Constraint coefficients for the job assignment constraints in the RMP will take on values given by the decision variables \( \theta_{kp} = w_i \forall i \in I \).

The value of tardiness is \( t_{kp} = \sum_{l} v_{il} \) for the new column. Finally, a value of 1 as the coefficient for the \( k \)-th convexity constraint (with a value of 0 for all others) will form the remainder of the column. It is not strictly necessary to find an optimal solution to the pricing problems, but any new columns generated for the master problem should give a negative objective function value in the pricing problem. We define the following sets and indices for the column generation formulation pricing problem (CG-P):

- \( I \) — set of IAGs for repair. Each IAG consists of one or more similar equipment items for repair with identical origin, destination, release date and due date,
- \( K \) — single member set that is the particular facility, represented by the pricing problem. (i.e. \(|K| = 1\)),
- \( S \) — set of time periods.

Parameters for CG-P include:

- \( r_i \) — release date of all items in IAG \( i \) at their origin,
- \( d_i \) — due date of all items in IAG \( i \) at their destination,
- \( l_{ik} \) — time required to perform repair of a single item in IAG \( i \) at facility \( k \),
- \( \tau_{ik}^1 \) — time to ship an item as carcass from IAG \( i \) to facility \( k \),
- \( \tau_{ik}^2 \) — time to ship a finished item from IAG \( i \) to facility \( k \),
resource (labor hours) available at facility $k$ during time period $s$,

amount of resource (labor hours) an item in IAG $i$ requires during each
time period of its repair time ($l_{ik}$) if it is repaired at facility $k$,

total costs to execute a repair of an item in IAG $i$ on facility $k$
$c_i = \sigma_{ik}^1 + p_{ik} + \sigma_{ik}^2$; Note that $k$ is fixed for the pricing problem,

dual variable value associated with $i$-th job assignment constraint from LP
relaxation of dual problem,

dual variable value associated with tardiness constraint from LP relaxation
of dual problem,

dual variable value associated with $k$-th convexity constraint from LP
relaxation of dual problem,

maximum total tardiness for the instance.

We define also the following parameterized set:

$U_{is}$ for the represented facility, the set time periods during which an item in
IAG $i$ will be processing if the job begins at time $s$;

$U_{is} = \{s' | s - l_{ik} < s' \leq s\}$.

We define the following variables for CG-P:

positive integer variables representing the number of items from IAG $i$
repaired as part of new column,
\( y_{is} \)
positive integer variables representing the number of items from IAG \( i \)
that start at time period \( s \),

\( \nu_i \)
positive continuous variables representing tardiness due to items in IAG \( i \).

Objective:

\[
z_k = \min \sum_i (c_i - \pi_i)w_i - \rho \sum_i \nu_i - \theta_k
\]

Subject to:

\[
\sum_S y_{is} \geq w_i \forall i \in I \tag{1}
\]

\[
\sum_I \sum_{s'} \mu_{is} y_{is'} \leq m_{ks} \forall s \in S \tag{2}
\]

\[
\sum_{s|s_0 < s < s+1} y_{is} = 0 \forall i \in I \tag{3}
\]

\[
\nu_i = \sum_K \sum_S (s + l_{ik} + \tau_{ik}^2 - d_i) y_{is} \forall i \in I, \forall s \in S|s \geq 1 + d_{ik} - l_{ik} - \tau_{ik}^2 \tag{4}
\]

\[
\sum_I \nu_i \leq b \tag{5}
\]

\[
y_{is}, w_i \in Z^+ \forall i \in I, s \in S \tag{6}
\]

Constraint set (1) requires that any items repaired as part of the new column be scheduled on the facility using the \( y_{is} \) decision variables. Constraint set (2) requires that all IAG items active during a time unit consume less than the available resource (i.e. labor available) at that time period by “looking back” to determine if IAG items have begun and are within their active period of processing defined by \( l_{ik} \). Constraint set (3) prevents IAG items from starting before they are released and can be transported to the facility.
Constraint set (4) computes tardiness for each IAG. For each IAG, we include integer IAG variables only for time periods that would result in tardiness. For this integer formulation, where some IAG items might accrue tardiness while others might not, we must adopt the constraint described by (4). Constraint set (5) requires that the total tardiness resulting from the new column be less than the maximum allowed tardiness. Constraint set (6) defines binary variables.

### 3.4 Sifting Formulation

Without additional work, the quality of any feasible solution found with the Restricted Standard Formulation (RSF) is not known, since all columns (variables) are not seen by the solver. As we have seen, Benders Decomposition Formulation is one means to determine a lower bound that may be used to estimate the quality of a feasible solution to an instance of the MWP. The Linear Programming Relaxation (LPR) is another way to obtain a lower bound. Typically, the LPR of an integer program is quick to solve relative to its integer program. However, for the MWP, some instances are so large that all variables do not load into memory within reasonable times. For these problems, we cannot directly obtain the LPR lower bound. In order to obtain the LPR we use the concept of candidate lists and sifting. Sifting is discussed in (Bixby et.al. 1991) where it is attributed to (Forrest 1989). However, the technique is older, having been used in (Crowder and Padberg 1980) to establish lower bounds for a travelling salesman problem. We note that modern commercial solvers, such as Gurobi, can perform sifting as we describe, but only after variables have been loaded into the model. Our method provides this capability outside the model and without loading variables into the model.
The sifting formulation restricted master problem (Sift-RMP) begins with the Restricted Standard Formulation (RSF) described in section 3.1.2. Recall that we create only a subset of variables from the full Standard Formulation (SF). For each IAG, we evaluate the cost to perform repair at each facility \( c_{ik} = \sigma_{ik}^1 + p_{ik} + \sigma_{ik}^2 \) and identify the least expensive \( n \) facilities. For each IAG, call this set of least expensive facilities \( E_i(n) \).

We also consider the necessary transportation time and duration of each repair operation at each facility, \( \tau_{ik}^1 + l_{ik} + \tau_{ik}^2 \), we identify the \( m \) ‘fastest’ facilities for each IAG and include these facilities in set \( F_i(m) \). Then, for each IAG, we create variables for only the union of these sets, \( E_i(n) \cup F_i(m) \). These variables form our initial working variables

\[
W = \bigcup_i E_i(n) \cup F_i(m).
\]

We solve the linear program relaxation of this problem (Sift-RMP-LPR) to obtain dual variable values. Recall, from section 3.2 that we have four constraint sets (the remaining constraint set defines variables):

\[
\pi_i : \sum_{K_i} \sum_S x_{iks} \geq q_i \forall i \in I \tag{1}
\]

\[
\rho : \sum_l \sum_{K_i} \sum_{SI} \left( s + l_{ik} + \tau_{ik}^2 - d_i \right) x_{ikr} \leq T_{\max} \forall i \in I \tag{2}
\]

\[
\phi_i : \sum_{K_i} \sum_{SI} \left( s + l_{ik} + \tau_{ik}^2 - d_i \right) x_{iks} \leq T_{\max} \forall i \in I \tag{3}
\]

\[
\theta_{ks} : \sum_{I_k} \sum x_{iks} \leq m_{ks} \forall k \in K, s \in S \tag{4}
\]
We can identify dual variables for sets (1)-(4) as shown to the left of each equation. Note that the values of the variables associated with (1) will be non-negative and the values of those associated with (2)-(4) will be non-positive.

We then consider each variable not included in Sift-RMP from the Standard Formulation (SF). If we take $X$ as the complete set of variables from the Standard Formulation, this is $X \setminus W$. We add to our candidate list variables that price favorably (i.e. negatively). The pricing problem for this computation uses the dual variable values computed when we solve Sift-RMP-LPR. We identify $t_{iks} = \max(0, s + l_{ik} + \tau_{ik}^2 - d_i)$ as the tardiness resulting if an item in batch $i$ starts processing at facility $k$ at time period $s$. We can then evaluate each possible variable that we have not included in Sift-RMP-LPR (i.e. $X \setminus W$ ). The pricing problem is the evaluation of the equation

$$Z_{\text{Sift-RMP}}(i,k,s) = c_{i,k} - \pi_i + \phi_i + \sum_{s' \in U_i} \theta_{ks} \mu_{ik} + \rho t_{iks}$$

If this quantity is less than zero then the variable $x_{iks}$ may reduce the objective function value of Sift-RMP-LPR. Those that price negatively, we place into the candidate list. For smaller problems, it may be practical to add every variable that prices negatively, however for larger problems, this approach is not practical, so once we have considered all variables $X \setminus W$, then we add the most negatively priced (i.e. smallest) $q$ as variables to Sift-RMP. We selected $q$ as equal to a half million. This means that as many as five hundred thousand variables may be added in an iteration. The most negative are added first. Naturally, when the number of variables in the candidate list is less than $q$, we simply add all variables in the candidate list.
After adding the most negatively priced variables, we then resolve the master problem (Sift-RMP-LPR) to obtain new dual variable values and repeat the pricing or all variables in \(X \setminus W\). When all variables in the standard formulation have non-negative reduced cost, we have proven that the objective function value of Sift-RMP-LPR gives a lower bound on the MWP and is equal to the LPR of the full standard formulation. This value can be used to estimate the quality of any known feasible solution.

Once we have this proven lower bound, we solve Sift-RMP as an integer program (not relaxed) to obtain a feasible integer solution. It may be that the proven quality of the identified solution is not sufficient. As with column generation, it is possible to implement a branching scheme to partition and search the feasible region of the Sift-RMP to identify and more closely bound the solution. At each node of the branching scheme (partition of the feasible region), one must search for variables that might reduce the LPR objective function value to obtain the proper relaxation bound. We did not find it necessary to resort to this approach in order to identify provably near optimal solutions.
IV-EXPERIMENTAL RESULTS

Considering both prior work in (Kotkin et.al. 2011) and our newly developed formulations, we perform computational experiments to evaluate the implemented efficacy of each formulation at generating provably near-optimal solutions to the MWP. In the first section, we repeat work performed by the Army Materiel Systems Analysis Activity (AMSAA) in (Kotkin et.al. 2011) and implement their sliding windows formulation (SWF). Using the prior results, we define experimental instances according to equipment group, maximum allowed tardiness and maximum allowed runtime. In section 4.2, we report results for our standard formulation (SF) that allows the model to determine batch-size rather than treating batches as fixed. In section 4.3, we report results for our Benders Decomposition formulation (BDF). In section 4.4, we report results for our column generation formulation (CGF). In sections 4.5-4.6, we report results for our “hybrid” algorithms. Section 4.5 gives results for the use of restricted standard formulation (RSF) in conjunction with Bender’s Decomposition (RSF-BDF). Finally, section 4.6 gives results for our sifting formulation (SIFT) that builds from RSF. We implemented each formulation using the Python programming language and the Gurobi commercial solver. Instance runs were conducted on a Linux (Ubuntu 13.10) with 32GB RAM and a 3.0 GHz Intel i5 processor (quad core).
### 4.1 Sliding Windows Formulation

We first implemented the Sliding Windows Formulation from (Kotkin et al. 2011). We used the same weighted objective function and data to run each prior instance. The objective function weights, from the prior work, can be found in Table 4. We compared our results to those obtained by AMSAA in that prior research. As prior work, the formulation for SWF can be found in Chapter 2. In Table 3, we give the computational resources and implementation characteristics for both the prior research and our research.

<table>
<thead>
<tr>
<th>1. Equipment Group / Weight Setting</th>
<th>2. SWF Operating Cost from (Kotkin et al. 2011) (USD)</th>
<th>3. SWF Tardiness from (Kotkin et al. 2011) (wks)</th>
<th>4. Runtime Reported in (Kotkin et al. 2011) (s)</th>
<th>5. SWF Operating Cost (USD)</th>
<th>6. SWF Tardiness (wks)</th>
<th>7. Runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/M</td>
<td>67339.72</td>
<td>0</td>
<td>2</td>
<td>67339.72</td>
<td>0</td>
<td>0.7</td>
</tr>
<tr>
<td>2/M</td>
<td>1309658</td>
<td>0</td>
<td>5</td>
<td>1309658</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td>3/T</td>
<td>2563741</td>
<td>229</td>
<td>14</td>
<td>2563700</td>
<td>229</td>
<td>6</td>
</tr>
<tr>
<td>3/M</td>
<td>2531260</td>
<td>235</td>
<td>14</td>
<td>2531260</td>
<td>235</td>
<td>6</td>
</tr>
<tr>
<td>3/C</td>
<td>2323977</td>
<td>1317</td>
<td>15</td>
<td>2323977</td>
<td>1197</td>
<td>7</td>
</tr>
<tr>
<td>4/M</td>
<td>1324432</td>
<td>0</td>
<td>28</td>
<td>1300205</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>4/C</td>
<td>1162923</td>
<td>3137</td>
<td>57</td>
<td>1156834</td>
<td>2871</td>
<td>27</td>
</tr>
<tr>
<td>5/M</td>
<td>2392397</td>
<td>1736</td>
<td>4514</td>
<td>2392397</td>
<td>1409</td>
<td>4514</td>
</tr>
<tr>
<td>6/M</td>
<td>3751779</td>
<td>0</td>
<td>33</td>
<td>3751779</td>
<td>0</td>
<td>31</td>
</tr>
<tr>
<td>7/M</td>
<td>16051777</td>
<td>0</td>
<td>180</td>
<td>16051710</td>
<td>0</td>
<td>117</td>
</tr>
<tr>
<td>8/T</td>
<td>18857438</td>
<td>602</td>
<td>369</td>
<td>18857438</td>
<td>602</td>
<td>208</td>
</tr>
<tr>
<td>8/C</td>
<td>18856734</td>
<td>608</td>
<td>842</td>
<td>18856734</td>
<td>608</td>
<td>208</td>
</tr>
<tr>
<td>10/M</td>
<td>145474000</td>
<td>360</td>
<td>57706</td>
<td>NA</td>
<td>NA</td>
<td>57706</td>
</tr>
<tr>
<td>11/M</td>
<td>50183986</td>
<td>840</td>
<td>68460</td>
<td>NA</td>
<td>NA</td>
<td>68460</td>
</tr>
<tr>
<td>12/M</td>
<td>260209631</td>
<td>2547</td>
<td>71473</td>
<td>NA</td>
<td>NA</td>
<td>71473</td>
</tr>
</tbody>
</table>
In the table above, we give the solution reported in (Kotkin et.al. 2011). We also report the best solution we obtained for the given equipment group and weight setting. In solving our formulation, we set termination conditions for the solver to solve to a relative gap of 0.0001 (or better), but allowed no more time than that reported by AMSAA for the instance. Note that the meaning of the relative gap here is not the same as for the remainder of our results. Recall that SWF effectively over-constrains the full problem by batching items. Any lower bound identified with SWF is a lower bound particular to the over-constrained (i.e. restricted) SWF problem and is not valid for the full-sized problem. For the most part, our results using the SWF parallel those of AMSAA in (Kotkin, et.al. 2011). We obtained better results for three instances (4/M, 4/C and 7/M), however, we were unable to solve the final three instances (10/M, 11/M and 12/M) in any reasonable timeframe. The prior research was likely able to solve these, albeit at exceptionally long runtimes, only due to the availability of substantially more RAM (128 GB vs. our 32 GB). Our computer architecture does not allow the addition of more than 32 GB of RAM, so we attempted to remedy this using increased virtual memory (Linux swap), but still found load time in excess of twenty hours. These results do not include Equipment Group 9 because this was found to be infeasible when formulated with batched jobs in (Kotkin et.al. 2011). Our work confirms this fact. Further we find, using later models, that Equipment Group 9 is infeasible for the full-sized (unbatched items) problem for a two-year planning horizon. It is also infeasible for a three-year planning horizon.

We used the results of the prior work (Kotkin et.al. 2011) to define experimental instances for our subsequent work in this research. After initial testing, we split the
equipment groups into low difficulty, medium difficulty and high difficulty sub-groups. We established maximum runtime for the sub-groups as 1 minute, 10 minutes and one hour respectively. Table 6 defines experimental instances, each with a maximum tardiness and a maximum allowed runtime.

4.2 Standard Formulation Results

We next implemented the MWP using our Standard Formulation (SF) from section 3.1. Recall that the SF seeks to minimize cost while constraining total tardiness to some maximum allowed amount defined by the particular experimental instance. Additionally, our implementation of SF effectively permits the model to select batch sizes by modeling each IAG with an integer variable rather than a binary variable for each batch. Although we are increasing the number of values each variable might take by using integer variables, we were actually able to reduce the variable-space (number of variables necessary) as we were able to combine some item-batches of similar items with identical origin, destination and repair type into a single IAG. Recall that for SWF, item-batches are limited in size so the item-batch’s processing requirement (labor required per item times number of items in the batch) does not completely exhaust the ability of any facility with the capability to repair the items within the item-batch. No such preprocessing is necessary in SF, since the model itself will determine the number of items within the IAG to process at each time period with an integer valued variable.
For this experiment, the termination criteria are a relative gap of less than 0.0001 between the upper and lower bounds or a run-time in excess of that defined for the instance.

Using the SF with increased state space (integer variables), we were able to identify upper and lower bounds for all but the two largest instances of the MWP.

4.3 Benders Decomposition Formulation (BDF) Results

We performed a similar computational experiment using our Benders Decomposition Formulation. Recall that BDF is implemented using a binary variable to represent each item. This is because we found it difficult to create valid and useful integer-variable-based cuts to communicate information about sub-problem infeasibility to the master problem. While this would seem to create an exceptionally large problem, one of the advantages of decomposition is that we do not need to retain sub-problems in

---

Table 4: Full Standard Formulation Results

<table>
<thead>
<tr>
<th>Equipment Group</th>
<th>Inst Num</th>
<th>Best Solution / Upper Bound (USD)</th>
<th>Lower Bound (USD)</th>
<th>Best Solution Observed Tardiness (wks)</th>
<th>Time of Last bound update</th>
<th>Max Runtime</th>
<th>Relative Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>67339.72</td>
<td>67339.72</td>
<td>0</td>
<td>0.06</td>
<td>60</td>
<td>0.00000</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1309657.98</td>
<td>1309657.98</td>
<td>0</td>
<td>0.2</td>
<td>60</td>
<td>0.00000</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2563700.95</td>
<td>2563700.95</td>
<td>229</td>
<td>21</td>
<td>60</td>
<td>0.00000</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2531259.77</td>
<td>2531259.77</td>
<td>235</td>
<td>11</td>
<td>60</td>
<td>0.00000</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2323977.29</td>
<td>2323977.29</td>
<td>1299</td>
<td>19.5</td>
<td>60</td>
<td>0.00000</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1300157.44</td>
<td>1299522.29</td>
<td>0</td>
<td>2.8</td>
<td>60</td>
<td>0.00049</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1158712.32</td>
<td>1151684.84</td>
<td>3104</td>
<td>27.4</td>
<td>60</td>
<td>0.00610</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2364501.55</td>
<td>2360139.78</td>
<td>1735</td>
<td>233</td>
<td>600</td>
<td>0.00185</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3751779.89</td>
<td>3751779.89</td>
<td>0</td>
<td>3</td>
<td>60</td>
<td>0.00000</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>16051650.65</td>
<td>16051598.36</td>
<td>0</td>
<td>14.6</td>
<td>60</td>
<td>0.00000</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>18857437.84</td>
<td>18857437.84</td>
<td>602</td>
<td>66.9</td>
<td>600</td>
<td>0.00000</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>18856734.04</td>
<td>18856734.04</td>
<td>608</td>
<td>79.8</td>
<td>600</td>
<td>0.00000</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>145461189.72</td>
<td>14545146.40</td>
<td>360</td>
<td>429.8</td>
<td>3600</td>
<td>0.00005</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>NA</td>
<td>0.00</td>
<td>NA</td>
<td>NA</td>
<td>3600</td>
<td>NA</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>NA</td>
<td>0.00</td>
<td>NA</td>
<td>NA</td>
<td>3600</td>
<td>NA</td>
</tr>
</tbody>
</table>
memory. We found that we could create and destroy sub-problem models quickly. Only the master problem resides in memory for the whole algorithm. As with the Standard Formulation, we establish termination criteria as a relative gap of 0.0001 between upper and lower bounds or a runtime in excess of the maximum allowed for the instance.

**Table 5: Benders’ Decomposition Results**

<table>
<thead>
<tr>
<th>Equipment Group</th>
<th>Max Tardiness ((b))</th>
<th>Best Solution / Upper Bound (USD)</th>
<th>Lower Bound (USD)</th>
<th>Best Solution Observed Tardiness (wks)</th>
<th>Time of last bound update</th>
<th>Max Runtime</th>
<th>Relative Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>67340</td>
<td>67340</td>
<td>0</td>
<td>0.08</td>
<td>60</td>
<td>0.00000</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1309658</td>
<td>1309658</td>
<td>0</td>
<td>0.7</td>
<td>60</td>
<td>0.00000</td>
</tr>
<tr>
<td>3</td>
<td>229</td>
<td>NA</td>
<td>2520860</td>
<td>NA</td>
<td>1.6</td>
<td>60</td>
<td>NA</td>
</tr>
<tr>
<td>3</td>
<td>235</td>
<td>NA</td>
<td>2491637</td>
<td>NA</td>
<td>1.5</td>
<td>60</td>
<td>NA</td>
</tr>
<tr>
<td>3</td>
<td>1317</td>
<td>NA</td>
<td>2323977</td>
<td>NA</td>
<td>0.2</td>
<td>60</td>
<td>NA</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>NA</td>
<td>1263635</td>
<td>NA</td>
<td>58.6</td>
<td>60</td>
<td>NA</td>
</tr>
<tr>
<td>4</td>
<td>3137</td>
<td>NA</td>
<td>1147459</td>
<td>NA</td>
<td>58.2</td>
<td>60</td>
<td>NA</td>
</tr>
<tr>
<td>5</td>
<td>1736</td>
<td>NA</td>
<td>2344502</td>
<td>NA</td>
<td>599.9</td>
<td>600</td>
<td>NA</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>3751780</td>
<td>3751484</td>
<td>0</td>
<td>18.7</td>
<td>60</td>
<td>0.00008</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>NA</td>
<td>16050539</td>
<td>NA</td>
<td>38.7</td>
<td>60</td>
<td>NA</td>
</tr>
<tr>
<td>8</td>
<td>602</td>
<td>NA</td>
<td>18856734</td>
<td>NA</td>
<td>9.3</td>
<td>600</td>
<td>NA</td>
</tr>
<tr>
<td>8</td>
<td>608</td>
<td>NA</td>
<td>18856734</td>
<td>NA</td>
<td>11</td>
<td>600</td>
<td>NA</td>
</tr>
<tr>
<td>10</td>
<td>360</td>
<td>NA</td>
<td>144895982</td>
<td>NA</td>
<td>2584.3</td>
<td>3600</td>
<td>NA</td>
</tr>
<tr>
<td>11</td>
<td>840</td>
<td>NA</td>
<td>50021853</td>
<td>NA</td>
<td>3561.2</td>
<td>3600</td>
<td>NA</td>
</tr>
<tr>
<td>12</td>
<td>2547</td>
<td>NA</td>
<td>259592784</td>
<td>NA</td>
<td>2523.3</td>
<td>3600</td>
<td>NA</td>
</tr>
</tbody>
</table>

Recall that the Benders’ Decomposition Formulation works by generating successive lower bounds for the problem. Although this formulation was not generally able to converge to an upper bound quickly, it could arrive at a good lower bound for every instance regardless of problem size. In cases where we do not arrive at an upper bound, the time given in the column labeled “Time of last bound update” gives the time
that the given lower bound was achieved, although instances were run for the complete allowed time.

Since we are able to compute a lower bound for each instance using the Benders Formulation, we can now use these results to lower bound the solutions obtained by AMSAA’s Sliding Windows Formulation (SWF). These are shown in Table 10.

<table>
<thead>
<tr>
<th>Equipment Group</th>
<th>SWF Obj Weights</th>
<th>SWF Observed Tardiness</th>
<th>SWF-1 Op Cost UB (AMSAA)</th>
<th>Lower bound (Benders' Decomposition)</th>
<th>Relative Gap (AMSAA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>M</td>
<td>0</td>
<td>67340</td>
<td>67340</td>
<td>0.00000</td>
</tr>
<tr>
<td>2</td>
<td>M</td>
<td>0</td>
<td>1309658</td>
<td>1309658</td>
<td>0.00000</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>229</td>
<td>2563741</td>
<td>2520860</td>
<td>0.01701</td>
</tr>
<tr>
<td>3</td>
<td>M</td>
<td>235</td>
<td>2531260</td>
<td>2491637</td>
<td>0.01590</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>1317</td>
<td>2323977</td>
<td>2323977</td>
<td>0.00000</td>
</tr>
<tr>
<td>4</td>
<td>M</td>
<td>0</td>
<td>1324432</td>
<td>1263635</td>
<td>0.04811</td>
</tr>
<tr>
<td>4</td>
<td>C</td>
<td>3137</td>
<td>1162923</td>
<td>1147459</td>
<td>0.01348</td>
</tr>
<tr>
<td>5</td>
<td>M</td>
<td>1736</td>
<td>2392397</td>
<td>2344502</td>
<td>0.02043</td>
</tr>
<tr>
<td>6</td>
<td>M</td>
<td>0</td>
<td>3751779</td>
<td>3751484</td>
<td>0.00008</td>
</tr>
<tr>
<td>7</td>
<td>M</td>
<td>0</td>
<td>16051777</td>
<td>16050539</td>
<td>0.00008</td>
</tr>
<tr>
<td>8</td>
<td>M</td>
<td>602</td>
<td>18857438</td>
<td>18856734</td>
<td>0.00004</td>
</tr>
<tr>
<td>8</td>
<td>C</td>
<td>608</td>
<td>18856734</td>
<td>18856734</td>
<td>0.00000</td>
</tr>
<tr>
<td>10</td>
<td>M</td>
<td>360</td>
<td>145474000</td>
<td>144895982</td>
<td>0.00399</td>
</tr>
<tr>
<td>11</td>
<td>M</td>
<td>840</td>
<td>50183986</td>
<td>50021853</td>
<td>0.00324</td>
</tr>
<tr>
<td>12</td>
<td>M</td>
<td>2547</td>
<td>260209631</td>
<td>259592784</td>
<td>0.00238</td>
</tr>
</tbody>
</table>

4.4 Column Generation Formulation (CG) Results

Our column generation formulation (CG) operates by use of a restricted master problem (RMP) where each binary variable represents a particular assignment of items to some facility. The RMP selects a single variable for each facility. With its complete
selection of variables, the RMP must satisfy the assignment constraints requiring that all items be assigned to a facility. Further, the tardiness resulting at all facilities must sum the less than that allowed for the instance. The tardiness resulting from an assignment to a facility is part of that variable’s constraint-coefficient column. To run the algorithm, we solve the linear program relaxation (LPR) of the RMP. Of note, we do require a feasible solution to the LPR to start the algorithm. We ensure that a solution exists by creating a single ‘phantom’ facility that has the ability to repair every item, but at a very high cost to the objective function. We create a sub-problem for each facility, and using the dual prices from the RMP-LPR, we can identify new assignments to that facility that have the potential to reduce the objective function of the RMP-LPR. Often one checks for integrality of the RMP-LPR solution to determine whether a solution feasible to the RMP has been found. We found it more efficient to solve the RMP as an integer program after the generation of columns on the sub-problems. Any feasible integer solution of the RMP is a valid upper bound to the problem. We do not obtain a lower bound for the problem until we can demonstrate through sub-problem pricing that there are no additional columns, not already part of the RMP, which might reduce the objective function value of the RMP-LPR. However, we can establish a valid lower bound quickly by determining the lowest possible repair cost for each IAG and summing this quantity for the problem. This is similar to the way in which Bender’s Decomposition Formulation arrives at its initial lower bound.
We establish the same set of termination criteria as for SF and BDF. We terminate when we can demonstrate an optimality gap of less than 0.0001 or we exceed the time allowed for the instance.

Table 7: Column Generation Results

<table>
<thead>
<tr>
<th>Equipment Group</th>
<th>Max Tardiness (b)</th>
<th>Best Solution / Upper Bound (USD)</th>
<th>Lower Bound (USD)</th>
<th>Best Solution Observed Tardiness (wks)</th>
<th>Time of last bound update</th>
<th>Max Runtime</th>
<th>Relative Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>67340</td>
<td>67340</td>
<td>0</td>
<td>0.1</td>
<td>60</td>
<td>0.00000</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1309658</td>
<td>1309658</td>
<td>0</td>
<td>0.7</td>
<td>60</td>
<td>0.00000</td>
</tr>
<tr>
<td>3</td>
<td>229</td>
<td>2999438</td>
<td>2323977</td>
<td>229</td>
<td>56.0</td>
<td>60</td>
<td>0.29065</td>
</tr>
<tr>
<td>3</td>
<td>235</td>
<td>3269945</td>
<td>2329377</td>
<td>235</td>
<td>55.3</td>
<td>60</td>
<td>0.40379</td>
</tr>
<tr>
<td>3</td>
<td>1317</td>
<td>3662639</td>
<td>2323977</td>
<td>1317</td>
<td>56.7</td>
<td>60</td>
<td>0.57602</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1784934</td>
<td>1136780</td>
<td>0</td>
<td>53.2</td>
<td>60</td>
<td>0.57017</td>
</tr>
<tr>
<td>4</td>
<td>3137</td>
<td>NA</td>
<td>1136780</td>
<td>NA</td>
<td>3.4</td>
<td>60</td>
<td>NA</td>
</tr>
<tr>
<td>5</td>
<td>1736</td>
<td>NA</td>
<td>2311051</td>
<td>NA</td>
<td>26.2</td>
<td>60</td>
<td>NA</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>NA</td>
<td>3744983</td>
<td>NA</td>
<td>20.7</td>
<td>60</td>
<td>NA</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>NA</td>
<td>16041159</td>
<td>NA</td>
<td>5.9</td>
<td>60</td>
<td>NA</td>
</tr>
<tr>
<td>8</td>
<td>602</td>
<td>NA</td>
<td>18856734</td>
<td>NA</td>
<td>11.8</td>
<td>600</td>
<td>NA</td>
</tr>
<tr>
<td>8</td>
<td>608</td>
<td>NA</td>
<td>18856734</td>
<td>NA</td>
<td>11.0</td>
<td>600</td>
<td>NA</td>
</tr>
<tr>
<td>10</td>
<td>360</td>
<td>NA</td>
<td>144647734</td>
<td>NA</td>
<td>54.3</td>
<td>3600</td>
<td>NA</td>
</tr>
<tr>
<td>11</td>
<td>840</td>
<td>NA</td>
<td>49878069</td>
<td>NA</td>
<td>307.3</td>
<td>3600</td>
<td>NA</td>
</tr>
<tr>
<td>12</td>
<td>2547</td>
<td>NA</td>
<td>259608143</td>
<td>NA</td>
<td>2523.3</td>
<td>3600</td>
<td>NA</td>
</tr>
</tbody>
</table>

4.5 Intermediate Comparison of Methods

Had any of the preceding formulations given a comprehensive means of solving all instances of the problem to a provably near optimal solution, no further development would be necessary. Since this is not the case, we consider the results of the decomposed methods and the results of the Standard Formulation (with integer variables). In particular, we are interested in lower bounds. Figure 4 shows the lower bound proven by each of the preceding methods for instances where all methods could determine a lower
bound and where the lower bound proven differed among methods. We also determined the bound proven by the linear program relaxation of the standard formulation and plot that value on the sub-charts in Figure 4.

Figure 1: Solution Method Lower Bounds by Instance
From this we observe that for those cases where we can obtain the linear program relaxation lower bound (we could not reasonably load instances 11/1 or 12/1), the linear program relaxation bound is always at least as tight as the bound that we obtained by Benders’ Decomposition. We acknowledge that Benders’ Decomposition may have obtained better bounds had more time been allocated. This intermediate finding drove the development of the Sifting Formulation given in paragraph 3.4.

4.6 Restricted Standard Formulation (RSF) bounded by Benders Decomposition Formulation (BDF) Results

We observed with the Benders Decomposition Formulation that we were able to achieve lower bounds relatively quickly for each instance. Unfortunately, the algorithm did not often converge to a feasible (and therefore optimal) solution. We found it possible to create a much smaller restricted version of the standard formulation by creating variables for only the least expensive and fastest facilities. This formulation is described in Chapter 3. For this method, we first generate the Restricted Standard Formulation (RSF) by selecting \( n = m = 0.1 \lceil K \rceil \). That is, we create variables for a fraction of the least expensive and fastest facilities as we describe in section 3.1.2. We then solve this in an attempt to arrive at an upper bound. If the initial restriction is infeasible, we note the time to prove infeasibility and increment \( n \) and \( m \). We re-formulate the RSF and re-solve to obtain an upper bound cumulating all computation time. Once we obtain an upper bound from the RSF, we use the Benders Decomposition Formulation (BDF) to arrive at a lower bound. We attempt to solve each instance to a relative gap of 0.0001 or better using no more than the maximum allowed time for the instance for all necessary computation (RSF and BDF, to include infeasible RSF).
This approach gave upper and lower bounds for every experimental instance. Instances 3/1, 3/2 and 5/1 all required an increase in size of the RSF. That is, the initial set of variables in the restriction proved infeasible, so as described above, the number of facilities included for each IAG was increased. The subsequent re-solve proved feasible for all three instances.

### 4.7 Results: Sifting Formulation

Our final formulation attempts to improve both the lower bound and the upper bound (best feasible solution) through the use of column generation. As in section 4.5, we formulate the RSF using \( n = m = 0.1[ K ] \). We solve the RSF to obtain an initial upper bound and feasible solution to the linear program relaxation of the RSF. Now, for the

<table>
<thead>
<tr>
<th>Equipment Group</th>
<th>Max Tardiness ((b))</th>
<th>Best Solution / Upper Bound (USD)</th>
<th>Lower Bound (USD)</th>
<th>Best Solution Observed Tardiness (wks)</th>
<th>Time of last bound update</th>
<th>Max Runtime</th>
<th>Relative Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>67340</td>
<td>67340</td>
<td>0</td>
<td>0.7</td>
<td>60</td>
<td>0.00000</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1309658</td>
<td>1309658</td>
<td>0</td>
<td>0.3</td>
<td>60</td>
<td>0.00000</td>
</tr>
<tr>
<td>3</td>
<td>229</td>
<td>2563701</td>
<td>2520860</td>
<td>229</td>
<td>14.0</td>
<td>60</td>
<td>0.01699</td>
</tr>
<tr>
<td>3</td>
<td>235</td>
<td>2531280</td>
<td>2491637</td>
<td>235</td>
<td>14.0</td>
<td>60</td>
<td>0.01590</td>
</tr>
<tr>
<td>3</td>
<td>1317</td>
<td>2323977</td>
<td>2323977</td>
<td>1315</td>
<td>2.5</td>
<td>60</td>
<td>0.00000</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1311353</td>
<td>1263635</td>
<td>0</td>
<td>28.0</td>
<td>60</td>
<td>0.03776</td>
</tr>
<tr>
<td>4</td>
<td>3137</td>
<td>1155049</td>
<td>1147459</td>
<td>3086</td>
<td>11.7</td>
<td>60</td>
<td>0.00661</td>
</tr>
<tr>
<td>5</td>
<td>1736</td>
<td>2372740</td>
<td>2344502</td>
<td>1734</td>
<td>76.3</td>
<td>600</td>
<td>0.01204</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>3751780</td>
<td>3751484</td>
<td>0</td>
<td>3.3</td>
<td>60</td>
<td>0.00008</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>16052040</td>
<td>16050539</td>
<td>0</td>
<td>22.0</td>
<td>60</td>
<td>0.00009</td>
</tr>
<tr>
<td>8</td>
<td>602</td>
<td>18857438</td>
<td>18856734</td>
<td>602</td>
<td>94.3</td>
<td>600</td>
<td>0.00004</td>
</tr>
<tr>
<td>8</td>
<td>608</td>
<td>18856734</td>
<td>18856734</td>
<td>608</td>
<td>92.9</td>
<td>600</td>
<td>0.00000</td>
</tr>
<tr>
<td>10</td>
<td>360</td>
<td>145503268</td>
<td>144895982</td>
<td>360</td>
<td>277.2</td>
<td>3600</td>
<td>0.00419</td>
</tr>
<tr>
<td>11</td>
<td>840</td>
<td>50163440</td>
<td>50021853</td>
<td>839</td>
<td>332.8</td>
<td>3600</td>
<td>0.00283</td>
</tr>
<tr>
<td>12</td>
<td>2547</td>
<td>259777305</td>
<td>259578959</td>
<td>2547</td>
<td>689.4</td>
<td>3600</td>
<td>0.00076</td>
</tr>
</tbody>
</table>
sifting formulation (SIFT), we consider each variable from the full standard formulation as a candidate to enter the restricted formulation. Using the dual prices from the LP relaxation of the RSF, we price each variable not included in the restriction. Those that price favorably (negatively), indicating the possibility that including the variable may reduce the objective function value of the RSF-LPR, we add to the RSF. Once we have priced every possible variable, we re-solve the LPR of the RSF, with the new (negatively priced) variables. Armed with new dual prices, we again consider every variable not included in the RSF, adding those that price favorably. Once we can demonstrate that no variable, if added to the RSF, could reduce the RSF-LPR objective function, then the objective function value of the RSF-LPR is a lower bound on the objective function value for the full-sized problem. Further, we can re-solve the RSF with the added variables and now require the solver to solve the problem again as an integer program. This often yields a better upper bound for the problem.
The Sifting Formulation was successful at generating good solutions for all instances. It was also possible to generate a valid lower bound by pricing variables. We observe that the linear programming bound found by the Sifting Formulation is generally stronger than the bound identified by Benders’ Decomposition in comparable time.

### 4.7 Summary of Results

To facilitate comparison of methods and identification of relative strengths, we now summarize our results in the succeeding charts. Our initial chart depicts upper bounds identified for each instance and by each method. We give a Cleveland dot plot for each instance for visual comparison of the upper bounding capability of each method. The results shown as SWF-2 are results we obtained from our implementation of Sliding Windows Formulation (SWF)
Figure 2: Upper Bound Found by Solution Method

We observe that Sifting Formulation upper bounds are better than those obtained using the RSF-BDF method for the largest two instances. We also observe that for smaller cases where it is possible to obtain an upper bound using the Standard Formulation, the Sifting Formulation still performs comparably well the that method, in some cases even marginally outperforming the Standard Formulation.
We now provide a Cleveland dot plot for the lower bound proven for each instance by each method.

![Cleveland dot plot](image)

**Figure 3: Lower Bound Proven by Solution Method**
From these plots, we observe the continued superior lower bounds proven by the Standard Formulation method that are derived from the linear program relaxation. Although we cannot expect the Sifting Formulation to provide a better bound than the Standard Formulation lower bound (where that bound can be found), we observe that lower bounds provided by the Sifting Formulation are superior to those proven by Benders Decomposition.

As a final comparison of methods, we give a Cleveland dot plot of the relative gap for the solution that each method provides. In order to better visualize, we actually plot the square root of the relative gap.
As a means of providing a provably near optimal solution to all instances of this problem, outperforms RSF-BDF. Even for instances where we could use the Standard Formulation, the Sifting Formulation remains competitive with that formulation.
4.8 Tardiness and Cost Trade-Off

Using our model output, it is possible to outline a Pareto curve that indicates the trade-off between cost and tardiness. In the figures below, the dot indicates the feasible solution identified for the specified (x-axis) amount of allowed tardiness. We give charts for Equipment Categories 3 and 11. The charts include an absolute lower bound for cost as the lower (red) horizontal line. This is obtained by summing the lowest repair cost for each item, or assuming that all facilities have unlimited capacity.
The Pareto Chart for Equipment Category 3 required about five minutes of computation time (295 seconds) to obtain the points shown. At the left, a minimum of 229 weeks of total tardiness must be allowed in order to schedule all items during the time horizon. With 236 items in the dataset, this translates into an average of just under 1 week of tardiness per item. At the far right, by permitting 1200 weeks of total tardiness,
we can achieve the minimum possible operating cost for the system. This translates into 5.1 weeks of tardiness per item. We may also consider the actual distribution of tardiness for the equipment category when we allow different amount of tardiness.

Figu 6: Item Tardiness Distribution for Equipment Category 3
Equipment Category 11 contains 20623 items. At the far left, we require 827 weeks of allowed tardiness, translating into 0.04 weeks of tardiness per item. At the far right, 71702 weeks to total tardiness gives 3.5 weeks of tardiness per item in Equipment Category 11.

Figure 7: Equipment Category 11 Pareto Chart
These charts demonstrate the overall feasibility of our method. We can use our formulation, in particular the Sifting Formulation, to obtain solutions along the efficient frontier for these problems. Even for an instance too large to load and solve in a reasonable time (i.e. Equipment Category 11) using the Standard Formulation, we can obtain ten efficient solutions in 11075 seconds, or about three hours, using the Sifting Formulation.
V-INSIGHTS, CONCLUSIONS AND FURTHER RESEARCH

This work provides substantially improved methodologies for the identification of provably near optimal solutions to a military maintenance planning and scheduling problem. Using our improved optimization modeling for this type of problem, we have also outlined a means to explore the Pareto surface that trades off total tardiness for reduced total operating cost.

The work is directly relevant to the Military maintenance problem considered in this work. Beyond its direct application to military logistical scheduling, this research is relevant to large scale planning and scheduling that occurs in manufacturing and supply-chain applications. While this type of problem has been addressed in the literature, this research is differentiated in several important ways. First, our research provides insights concerning the development of good variable restricted models. Second, although the Benders’ Decomposition framework has already been applied to a similar problem (e.g. the PSP in (Hooker 2007), due to the possibility of tardiness, the formulations in this work cannot be directly applied to solve the MWP. Our Bender’s Decomposition, formulation employs an integer program sub-problem and the linear program relaxation of that integer sub-program in order to allocate tardiness to facilities. This permits the use of dual variable information to generate valid Benders’ cuts for the Benders’ master problem that help to allocate tardiness among the facilities. The integer program-
constraint program framework described in (Hooker 2007) is not suited to our problem because constraint programming generates no duality information that we could assist in allocating our total tardiness budget.

To our knowledge, our use of column generation to optimize a planning and scheduling type problem is unique. A further unique aspect of our work here is the use of two decomposition methods (i.e. Benders’ Decomposition and Dantzig-Wolfe Decomposition/Column Generation). The decomposition methods employed here have been in use since the 1960s, however there are relatively few examples of the use of both of these two alternative decomposition methods in the same research effort.

5.1 Insights

In the original work on the MWP, the decision to batch items and use binary variables to represent those item-batches rather than individual items successfully reduced the number of variables needed to represent the problem, but at the cost of eliminating many feasible solutions. The use of integer variables for Item Arrival Groups, as shown in our Standard Formulation can, in most cases, reduce the variable space, since we need not constrain the size of the IAG. Despite the substantial increase in state-space induced by the change to integer variables, solution times are not adversely affected. Indeed, many instances become easier to solve. In short, it is important to distinguish between one’s variable space and one’s state-space.

The use of item-batches and binary variables, as in SWF, is a proper restriction of the full-sized MWP. The SWF batching approach does allow many large problems to be solved to a near-optimal solution, even if the optimality is not proven as part of the
optimization. We found that a carefully designed variable restriction is more efficient in terms of variables created and overall solution time. We cannot overstate the importance of a well-designed restriction to the successful optimization of this large problem.

We might suppose that many large optimizations will have more than one true objective. In this case, we have both cost and total tardiness. We found it much more straightforward to constrain one objective and optimize for the other. We avoid the summation of entities with unlike units (i.e. we do not add dollars and weeks). Every objective of an optimization model need not be present in its objective function. We also note that it is possible to extract the true ‘cost of tardiness’ from a Pareto curve graph, like the one we produce rather than arbitrarily establishing a cost of tardiness at the outset of the optimization.

5.2 Conclusions

Relative to the prior work, SWF, we were unable to demonstrate the ability to consistently generate substantially better (i.e. lower cost) feasible solutions for particular levels of total tardiness. However, we find that we can now provide solutions with known near-optimal quality using either of two new methodologies. A restriction of the full-sized problem with Bender’s Decomposition providing a lower bound gives good known quality solutions to the MWP instances. Alternatively, using the restriction followed by sifting to bring in potentially improving variables and to generate the lower bound also provides a means to good known quality solutions. We also note that our new methods seem to be substantially faster despite our use of much less powerful computers.
In light of our results using variable restriction, a carefully designed restriction of the full problem does reliably yield good feasible solutions and upper bounds the MWP. We do not find that the application of Benders Decomposition to be a good means to identify feasible solutions for this application. Benders Decomposition is a good means to identify lower bounds on the optimal objective function value for our problem instances. These lower bounds can then be paired with feasible solutions obtained from the use of variable restriction to give solutions of known quality. The results of this research also prove that the Sliding Windows Formulation (SWF) did produce fairly good solutions to the true (full-sized) problem of interest. This quality could not be shown using only the SWF.

Likewise, the straightforward application of Dantzig-Wolfe decomposition and column generation failed to show utility with regard to the MWP. In this case, the idea of pricing variables was of utility. The idea of pricing variables into a formulation is the heart of the sifting formulation. We were surprised to find that the linear programming bound provided by the sifting approach was at least as strong as that provided by the Benders’ Decomposition. The possibility of improving one’s initial solution by including additional variables based on pricing often yields better (near-optimal) feasible solutions. These two observations lead us to conclude that this method is the most promising approach for the MWP.

5.3 Future Work
The use of variable restriction may be considered a heuristic method. There are other heuristic methods that might provide good solutions to this problem. Consideration
of other heuristic methods (i.e. genetic algorithms) may be in order, particularly since we have good methods for lower bounding optimal solutions.

Although we never found it necessary to resort to branch and bound in our sifting approach for the instances we have, some instances of the planning and scheduling problem or other problems similar to the MWP may not be bounded well by the linear program relaxation. For these cases, the development and research of custom branch and bound code may be of value.

Finally, further development of our methods may provide new methods for solving alternative scheduling problems. Within the literature, there are many variations of the scheduling problem to account for particular application characteristics. For example, we might wish expand our methods applicability to problems with equipment set-up times or set-up costs.
REFERENCES


BIOGRAPHY

Ryan R. Squires graduated from the United States Military Academy in 1995. He received a Masters of Science Degree in Engineering Management from Missouri University of Science and Technology in 1999 and a Masters of Science Degree in Operations Research from the Naval Postgraduate School in 2007. He has served in the United States Army as a commissioned officer in the United States Army since 1995 and presently holds the rank of Lieutenant Colonel. In addition to various stateside assignments, he has served in Saudi Arabia, Iraq, and Afghanistan.