Decomposition Techniques for Large-Scale Optimization in the Supply Chain

by

Lindsay Sanneman

B.S., Massachusetts Institute of Technology (2014)

Submitted to the Department of Mechanical Engineering in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

© Massachusetts Institute of Technology 2018. All rights reserved.
Abstract

Integrated supply chain models provide an opportunity to optimize costs and production times in the supply chain while taking into consideration the many steps in the production and delivery process and the many constraints on time, shared resources, and throughput capabilities. In this work, mixed integer linear programming (MILP) models are developed to describe the manufacturing plant, consolidation transport, and distribution center components of the supply chain. Initial optimization results are obtained for each of these models. Additionally, an integrated model including a single plant, multiple consolidation transport vehicles, and a single distribution center is formulated and initial results are obtained. All models are implemented and optimized for their given objectives using a standard MILP solver.

Initial optimization results suggest that it is intractable to solve problems of relevant scale using standard MILP solvers. The natural hierarchical structure in the supply chain problem lends itself well to application of decomposition techniques intended to speed up solution time. Exact techniques, such as Benders decomposition, are explored as a baseline. Classical Benders decomposition is applied to the manufacturing plant model, and results indicate that Benders decomposition on its own will not improve solve times for the manufacturing plant problem and instead leads to longer solve times for the problems that are solved. This is likely due to the large number of discrete variables in manufacturing plant model.

To improve upon solve times for the manufacturing plant model, an approximate decomposition technique is developed, applied to the plant model, and evaluated. The approximate algorithm developed in this work decomposes the problem into a three-level hierarchical structure and integrates a heuristic approach at two of the three levels in order to solve abstracted versions of the larger problem and guide towards high-quality solutions. Results indicate that the approximate technique solves problems faster than those solved by the standard MILP solver and all solutions are within approximately 20% of the true optimal solutions. Additionally, the approximate technique can solve problems twice the size of those solved by the standard MILP solver within a one hour timeframe.
Acknowledgments

I would first like to thank my advisor, Professor Julie Shah, for all she has done to support me through my last few years of research. Her support and caring really go above and beyond, and her positivity and encouragement have helped foster in me an excitement about research and graduate school. Her encouraging words have made such a difference to me, especially during some of the harder times. She has taught me so much, and I feel very lucky to have her as my advisor.

I would also like to thank Steelcase for funding my research and particularly Jennifer Tyler and Ed Vanderbilt for their guidance as I worked towards the Masters thesis.

My labmates in IRG have made the last few years some of the best years of my life. Everyone is so intelligent and supportive, and each person has each contributed to this thesis or to the other things I have done in graduate school in some way. I enjoy laughing with all of you and the coffee runs we take. You are all treasured friends of mine and make me excited to come into lab every morning.

I would also like to thank Professor Daniela Rus for her mentorship and guidance in the year before I began graduate school. I learned a lot about how to be an effective researcher from her and from my time in the Distributed Robotics Lab. I also learned a lot and laughed a lot with my DRL labmates, Joseph DelPreto, Ankur Mehta, and Robert Katzschmann, and I am grateful to them for all that they’ve taught me and all the fun we’ve had.

Joining the Lutheran Episcopal Ministry at the beginning of grad school was one of the happiest accidents of my life. I enjoy the conversations and deep questions we share in LEM and the humility with which everyone approaches faith. I would like to thank all of my LEM friends for their thoughtfulness and compassion. I very much look forward to taking a break with all of them every Wednesday, and I learn so much from each one of them. I especially want to thank the chaplains, Kari Jo Verhulst and Thea Keith-Lucas. They have been listening ears and pillars of support for me through my first years of grad school, and I cannot thank them enough for
their kindness and caring.

I would also like to thank everyone at the University Lutheran Church. I have truly found a home at UniLu, and I am so grateful to be surrounded by such a thoughtful group of people who care deeply about social justice issues and the wellbeing of all people. Each person at UniLu inspires me to strive to love others more and to love others in new ways every day.

I would also like to thank our Sanctuary guests at UniLu, a mom and her two wonderful daughters. The mom has taught me more about courage and strength than she knows, and the younger two bring me such joy every Friday with their energy, smiles, and playful spirits. I have been so blessed to watch the younger ones learn and grow over the last year, and I feel very lucky to know all three.

I would also like to thank all of the MIT pole vaulters past and present. They all have brought me many laughs over the years, and cheering each of them on through ups and downs has been one of the great joys of my time in graduate school so far. You guys are amazing, and I know each and every one of you will accomplish big things. Never forget your worth and how incredibly capable you are!

I especially want to thank Kathleen Brandes, who has been there for me through both some of the more difficult parts of grad school (including the qualification exams) and the fun parts. Her care and compassion and our shared sense of humor keep me going and inspire me every day.

I also want to thank wonderful, quirky, and witty 1W in East Campus. They brighten my days and really make our hall feel like a family. And beyond that, they have supported me and Patrick and taught us so much, even as we hope to support and teach them.

And I would like to thank the EC house team members (especially Rob Miller and Sandy Alexandre), who are all immensely passionate about student support and care so much about the wellbeing of our wonderful community (sorry,Yonadav!). Each and every person’s care for the students goes well beyond the call and is part of what I love about MIT. I aspire to be a little more like each of you every day.

I would also like to thank Henna Jethani, who has been there for me through so
much. She was there on my first day as a member of IRG and has been there through all the ups and downs of graduate school since. She has always been a cheerleader in my life, and I am so lucky to call her my friend.

I would also like to thank all of my other friends, including Katherine Evans, Rachel Luo, the Sharpes, the Coles, the Balls, and everyone from Arizona. Katherine, especially has been a dear friend of mine and has picked me up when I’ve been down and has been the source of many adventures over the years. And Rachel has been a great travel buddy and friend from our very first days at MIT. I also want to thank all of my other mentors over the years, including teachers from Rancho Solano and Dan, Pam, and Vasko from Arizona Sunrays.

Finally, I would like to thank my family, who has always loved and supported me. My parents have done so much for me and have always made a point to be there for me for both the big moments in life and the smaller ones. I would also like to thank Elise for her love and all of my grandparents for their love. They have all been so invested in me throughout my life, and they mean the world to me. I would like to thank my Grandma Betty in a special way. She taught me nearly everything I know about love, and I am forever grateful for all the time I had to learn from her. One of my favorite memories is being with her during my undergraduate graduation, and she will be greatly missed at this one. And I would lastly like to thank Patrick who has loved and supported me throughout grad school and who has taught me so much about supporting others. I would not have made it this far without him.
## Contents

1 **Introduction** 17  
1.1 Motivation ........................................ 18  
1.2 Supply Chain Modeling ............................. 18  
1.3 Exact Techniques ................................. 20  
1.4 Approximate Techniques .......................... 21  
1.5 Conclusions and Future Work ..................... 22  

2 **Supply Chain Modeling** 25  
2.1 Related Work .................................... 26  
2.2 Manufacturing Plant ............................... 28  
2.2.1 Manufacturing Plant Inputs, Outputs, Objective, and Constraints 29  
2.2.2 Manufacturing Plant MILP Formulation .................... 30  
2.2.3 Manufacturing Plant Results and Discussion .................. 33  
2.2.4 Limitations of the Manufacturing Plant Model ................ 35  
2.3 Consolidation Transport .......................... 36  
2.3.1 Consolidation Transport Inputs, Outputs, Objective, and Constraints 37  
2.3.2 Consolidation Transport MILP Formulation ................ 37  
2.3.3 Consolidation Transport Results and Discussion ............. 39  
2.3.4 Limitations of the Consolidation Transport Model ............ 40  
2.4 Distribution Center ............................... 41  
2.4.1 Distribution Center Inputs, Outputs, Objective, and Constraints 42  
2.4.2 Distribution Center MILP Formulation .................... 43
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4.3 Distribution Center Results and Discussion</td>
<td>45</td>
</tr>
<tr>
<td>2.4.4 Limitations of the Distribution Center Model</td>
<td>46</td>
</tr>
<tr>
<td>2.5 Integrated Model</td>
<td>47</td>
</tr>
<tr>
<td>2.5.1 Integrated Model Results and Discussion</td>
<td>48</td>
</tr>
<tr>
<td>2.5.2 Limitations of the Integrated Model</td>
<td>50</td>
</tr>
<tr>
<td>2.6 Future Modeling and Integration Work</td>
<td>51</td>
</tr>
<tr>
<td>3 Exact Techniques</td>
<td>53</td>
</tr>
<tr>
<td>3.1 Related Work</td>
<td>53</td>
</tr>
<tr>
<td>3.2 Benders Decomposition</td>
<td>55</td>
</tr>
<tr>
<td>3.2.1 Overview</td>
<td>55</td>
</tr>
<tr>
<td>3.2.2 Benders Decomposition Applied to the Manufacturing Plant</td>
<td>60</td>
</tr>
<tr>
<td>3.3 Results and Discussion</td>
<td>65</td>
</tr>
<tr>
<td>3.4 Future Work</td>
<td>67</td>
</tr>
<tr>
<td>4 Approximate Techniques</td>
<td>69</td>
</tr>
<tr>
<td>4.1 Related Work</td>
<td>69</td>
</tr>
<tr>
<td>4.2 Hierarchical Approximate Technique</td>
<td>73</td>
</tr>
<tr>
<td>4.2.1 Overview</td>
<td>73</td>
</tr>
<tr>
<td>4.2.2 Top Level: Work Allocation to Days</td>
<td>76</td>
</tr>
<tr>
<td>4.2.3 Middle Level: Labor Allocation to Workcenters</td>
<td>78</td>
</tr>
<tr>
<td>4.2.4 Bottom Level: Scheduling</td>
<td>80</td>
</tr>
<tr>
<td>4.2.5 Incompleteness of Approximate Technique</td>
<td>84</td>
</tr>
<tr>
<td>4.3 Results and Discussion</td>
<td>84</td>
</tr>
<tr>
<td>4.4 Future Work</td>
<td>89</td>
</tr>
<tr>
<td>5 Conclusion</td>
<td>91</td>
</tr>
<tr>
<td>5.1 Summary of Results</td>
<td>91</td>
</tr>
<tr>
<td>5.2 Future Work</td>
<td>94</td>
</tr>
</tbody>
</table>
## A Benders Decomposition Formulations for the Manufacturing Plant

<table>
<thead>
<tr>
<th>Problem</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1 Benders Optimality Cut Formulation</td>
<td>98</td>
</tr>
<tr>
<td>A.2 Feasibility Cut Formulation</td>
<td>99</td>
</tr>
<tr>
<td>A.3 Full Formulation of Benders Dual Subproblem</td>
<td>100</td>
</tr>
</tbody>
</table>
List of Figures

1-1 Supply Chain Integrated Model Layout .................................................. 20

2-1 Log Median Manufacturing Plant Model Runtimes for Three Workcenters, 10 Day Time Horizon ......................................................... 36

2-2 Log Median Consolidation Transport Model Runtimes ......................... 41

2-3 Log Median Distribution Center Model Runtimes ................................... 46

2-4 Log Median Integrated Model Model Runtimes ..................................... 50

4-1 Approximate Algorithm Hierarchy .......................................................... 76

4-2 Log Median Solve Times for Original MILP and Approximate Algorithm 88
# List of Tables

3.1 Feasibility Cuts, Optimality Cuts, and Solve Times for Two Days, Three Products ........................................... 66

4.1 Number of Problems Solved in under One Hour for Varying Problem Sizes .................................................. 86

4.2 Average Percent Differences between Approximate Solutions and MILP Solutions ......................................... 89

4.3 Percentage of Problems in which Initial MILP Solution is Better than Approximate Solution ............................. 89
Chapter 1

Introduction

The supply chain is a rich domain involving many different processes and handoffs that must integrate smoothly in order to take products from a material acquisition stage all the way to delivery to the final customer. Due to the complex nature of this problem and the costs associated with lost time in supply chain processes, it is of value to model and optimize these processes such that resources are used in a cost effective manner and deadlines are met. There are many different types of supply chains, and each supply chain might involve a number of stages. In this thesis, the specific case of a furniture manufacturing supply chain is explored, and models and assumptions are based on discussions with Steelcase, a furniture manufacturing company, about its supply chain processes. Three primary processes in the Steelcase supply chain include product manufacturing, distribution center operations, and transportation planning, and these three processes are the focus of this thesis. Models developed for supply chain components must balance both sufficient generality to describe the structure of the possibly-varying aspects of the supply chain processes as well as sufficient specificity to optimize the processes well. It is also important to consider the transitions between the different steps in the supply chain and to incorporate information about how a product will move downstream when scheduling upstream processes. To this end, the main contributions of this thesis include the following:

- The development of models for a manufacturing plant, a distribution center, and
consolidation transport and an integrated model that incorporates the processes involved in all three.

- The application of an exact decomposition technique, Benders decomposition, to the manufacturing plant model.

- The development of a hierarchical approximate algorithm and its application to the manufacturing plant problem.

1.1 Motivation

While researchers and companies have previously developed models for the different supply chain components, typically when each step in the process is optimized independently, it is likely that the overall flow through all steps in the supply chain process for any given product is suboptimal. Because of this, integrating the optimization of the numerous steps in the supply chain process into a single model is of interest. However, in the case of the furniture manufacturing supply chain explored in this work, thousands of products need to be scheduled through dozens of processes each week, so an integrated model will become very large for real-world-sized problems. As orders come in and need to be scheduled through all processes, optimization techniques need to be able to optimize schedules and resources quickly enough to keep up with demand and operations, likely within in sub-hour timeframes. Instead of searching for exact optimal solutions, approximate techniques could provide a way to solve such problems more quickly while outputting near-optimal solutions.

1.2 Supply Chain Modeling

Chapter 2 details the models that were developed for each component of the supply chain as part of this work and the assumptions that were made in developing these models. All components were developed as Mixed Integer Linear Programming (MILP) models, and the Gurobi optimizer for Java was used to solve them.
Three separate models were developed, including a manufacturing plant model, a consolidation transport model, and a distribution center model. The manufacturing plant model considers labor allocations to workcenters along the manufacturing line and scheduling of products through the required workcenters. The consolidation transport model is a vehicle packing and scheduling problem, taking into consideration vehicle capacity constraints and scheduled deadlines for products that are packed onto the vehicles. Finally, the distribution center model considers capacity constraints over time as products move into and out of the distribution centers. Chapter 2 also details an integrated model that was developed. The integrated model includes one manufacturing plant, one distribution center, and one set of consolidation transport vehicles. Figure 1-1 shows how products flow through each of the supply chain components in the case of the Steelcase supply chain: from the manufacturing plant to a factory distribution center, then onto consolidation transport vehicles in order to be shipped to a regional distribution center, and then finally onto outbound transport vehicles for delivery to the final customer. The boxed components are components that are incorporated into the integrated model. Both the regional distribution center and factory distribution center are modeled in the same way, and development and integration of an outbound transportation model is left as future work. In chapter 2, specifics of assumptions made for each of the models are given along with details about experiments run to test each model. Results for the three separate supply chain component models and the integrated model are included. The primary findings in chapter 2 include the following:

- The Gurobi optimizer was only able to solve problems an order of magnitude smaller than those of relevant size in the real-world supply chain for the manufacturing plant MILP model, the distribution center MILP model, and the integrated model. Larger consolidation transport MILP problems were solved using Gurobi, but the problems that were solved were still not as large as those that are dealt with in the real-world supply chain.

- The size of problem that can be solved for each of the models developed and
tested in chapter 2 is likely limited by the increase in number of decision variables and constraints with the increase in problem size.

1.3 Exact Techniques

While it is convenient to model different components of the supply chain in the MILP framework, solving these problems is in general NP-hard (Bertsimas and Weismantel [5]). A number of exact decomposition techniques have been developed to solve problems with the MILP structure more quickly by exploiting their decomposable structures. In particular, MILP problems can be split into a master problem and one or multiple subproblems with all binary- and integer-valued variables in the master problem and all continuous decision variables in the subproblem(s). Chapter 3 discusses such techniques in more detail, and one technique in particular, Benders decomposition, is detailed further and applied to the manufacturing plant problem. Implementation details for Benders decomposition applied to the manufacturing plant problem and corresponding results are given. The primary findings in chapter 3 include the following:

- It was not possible to solve even the smallest problems that were solved with the manufacturing plant MILP model in chapter 2 with the Benders decomposition implementation for the manufacturing plant problem. This suggests that
Benders decomposition will not decrease the solution time as will be necessary for solving real-world scale problems and instead increases solution time.

- The primary bottleneck in solving the manufacturing plant problem using Benders decomposition is that in general, many more feasibility cuts are added than optimality cuts, causing slow convergence. This is likely a byproduct of a relatively large number of binary- and integer-valued decision variables as compared with the number of continuous decision variables in the problem. Specific details of the feasibility and optimality cuts within the Benders decomposition framework are provided in chapter 2.

1.4 Approximate Techniques

Though exact decomposition techniques have been shown to provide runtime improvements for certain types of models with a decomposable structure, in general they are not sufficient to provide the necessary improvements in runtimes to solve realistic supply chain-sized problems in relevant timeframes. In chapter 4, approximate techniques are explored as a way of achieving high quality solutions in less time than exact techniques take to find the true optimal solution. An approximate technique is developed and is applied to the manufacturing plant problem. Results are given. The primary findings in chapter 4 include the following:

- The approximate technique solved manufacturing plant problems faster than the Gurobi optimizer solved the corresponding MILP formulations in all cases in which both techniques solved the problem before a one hour timeout.

- On average, the approximate technique arrived at its final solution before the Gurobi optimizer found its first solution to the MILP formulation.

- The approximate technique could solve problems twice as large as those solved with the Gurobi MILP formulation within a one hour time horizon.
Due to a bottleneck caused by the lowest level MILP in the approximate technique, the size of problem that can be solved using the approximate technique will be limited.

Solution quality for the approximate technique declined as problem size increased, although all approximate solutions were within 20.35% of the solution outputted by Gurobi in cases in which both techniques solved the problem.

1.5 Conclusions and Future Work

Chapter 5 details a summary of the results from this thesis and future work as it relates to the results. MILP models were developed for three components of the supply chain including the manufacturing plant, consolidation transport, and the distribution center, and an integrated model was also created and evaluated. Solving these models using the Gurobi optimizer suggested that for real-world-sized problems involving thousands of products to be scheduled in sub-hour timeframes, alternate techniques will be necessary. An exact decomposition technique, Benders decomposition, was applied to the manufacturing plant problem to serve as a baseline for exploration of decomposition techniques for problems like the ones developed in this work. Due to the structure of the manufacturing plant problem, Benders decomposition showed no improvement over solving the original MILP formulation using Gurobi. Exploring modified exact techniques, such as Benders decomposition with callbacks, could improve computational performance on problems with a large number of binary and integer decision variables, like the manufacturing plant model developed in this work. An approximate decomposition technique was developed and applied to the manufacturing plant problem, and computational improvements were achieved using this technique as compared with solving the original MILP formulation using Gurobi. A bottleneck remains in the lowest level scheduling problem of the approximate technique, since this is modeled as a MILP. Exploration of ways to reduce the impact of this bottleneck and thereby solve larger problems is left as future work. Some interesting additional future directions include:
• The expansion of the integrated model to account for additional complexity in the supply chain and other supply chain components that are not modeled in this thesis.

• Consideration of stochasticity in the supply chain models.

• An exhaustive study of different frameworks that could be used to model the supply chain problem.

• Application of machine learning techniques to better model different aspects of the supply chain and to allow models to generalize better.
Chapter 2

Supply Chain Modeling

The model of the supply chain proposed in this work includes three generalized supply chain components and an integrated model incorporating the three components and the interactions between them. These models were developed through discussions with an industry partner to ensure that they represent a real world problem well. The three components included in modeling efforts are the manufacturing plant, consolidation transport, and the distribution center. An outbound transport and delivery model will be added and integrated as part of future work.

Figure 1-1 in chapter 1 shows the layout of the supply chain components used for the integrated model in this work. The ordering of the components in the figure and implemented in the integrated model is used to demonstrate how the supply chain components developed as part of this work can be integrated to optimize over a larger portion of the supply chain as opposed to individual pieces. However, each component presented here functions as an individual piece that can be integrated into supply chain models according to many different configurations.

In this supply chain layout, orders are first received and scheduled for production at a manufacturing plant. After they are produced at the manufacturing plant, they move to a distribution center at the manufacturing plant where they are collected to be assigned to vehicles, called consolidation transport vehicles, that transport them to regional distribution centers. At regional distribution centers, products are collected from various manufacturing plants such that they can be shipped to the
final customer destinations in a single shipment. Finally, products are grouped onto outbound transport vehicles at the regional distribution center, and the outbound transport vehicles are routed to customer destinations. Each of the implemented model components is detailed further in the following sections, and the outbound transport component will developed as future work.

Each model was formulated as a mixed integer linear program (MILP) since the MILP framework lends itself well to a wealth of decomposition techniques that can be used to solve problems more quickly. All MILP models are formulated and solved using Gurobi for Java [16]. Some of the decomposition techniques that can be applied to MILP models are explored further in chapters 3 and 4.

2.1 Related Work

The supply chain with its various components has been modeled in the literature using the MILP framework as well as other frameworks, such as constraint programming (CP), and the focuses of the different models that have been developed vary. For example, Castro and Grossmann [9] propose a multistage, multi-product model of the manufacturing plant and formulate the problem as a continuous-time MILP. This model accounts for multiple workcenters and multiple products, but it assumes that all workcenters are required for all products, so it does not cover cases in which products only require a subset of the workcenters. The model developed is compared with a discrete-time MILP formulation and a CP formulation for three different objectives including minimizing cost, minimizing earliness, and minimizing makespan. The CP formulation is shown to perform well for the makespan minimization problem, the discrete-time formulation is shown is perform well for the earliness minimization problem, and the continuous-time formulation performs best with the cost minimization problem. The model they propose does not consider multi-objective problems. Floudas and Lin [14] perform a review of both discrete-time and continuous-time MILP models describing scheduling of manufacturing plant processes. They note that discrete-time models can be limited in the fact that many decisions variables
are necessary to achieve required accuracy for some production scheduling problems, and continuous-time approaches can reduce the number of required decision variables by employing event variables for things like start times and end times of production processes.

In addition to manufacturing plant models, other components of the supply chain have also been modeled using the MILP framework. For instance, Fanti, Stecco, and Ukovich [13] propose two MILP models to describe distribution center operations with the objective of minimizing operation time. While the models they propose account for unloading and loading processes at the distribution center, constraint capacities are eliminated in order to make the problems tractable to solve, and information about facility volumetric capacity is lost. Additionally, Maheut and Garcia-Sabater [20] model a transportation planning problem with the objective of minimizing vehicle usage using the MILP framework. In their model, vehicles are loaded while accounting for stock levels over time. However, they do not consider the flow of specific products through the supply chain and instead consider only inventories. In the motivating supply chain example considered in this work, specific products need to be tracked and scheduled through manufacturing processes, distribution centers, and transportation processes.

Integrated supply chain models have also been proposed in the literature. You, Grossmann, and Wassick [30] formulate a MILP model for simultaneous capacity, production, and distribution planning for a multi-site system and explore bi-level and Lagrangian decomposition techniques for solving the large-scale MILP models. Their model considers production trains at multiple production facilities and transit to distribution centers in multiple locations downstream. It does not consider specifics of vehicle packing and routing between locations and additionally does not track individual products through the supply chain, but rather considers high level capacities and flows. Application of bi-level and Lagrangian decomposition techniques showed solve time improvements for the examples given, and the bi-level technique showed greater improvement than the Lagrangian decomposition technique in all cases in which improvements were achieved. Sitek and Wikarek [25] propose a model integrating man-
ufacturing plants, distributors, and customers and formulate it as a hybrid CP/MILP model. In their model, details of production processes at the manufacturing plant are abstracted away. They solve a pure MILP formulation of the problem in addition to the hybrid model both with and without constraint propagation. The greatest improvement over the pure MILP formulation is achieved by the hybrid model incorporating constraint propagation. Other integrated supply chain models explore other aspects of the supply chain such as retailers, suppliers, and transportation considerations (Masoud [21], Pujari [22], Sitek and Wikarek [26]). While a wealth of models have been developed describing different components of the supply chain and the integrated supply chain, many of these models abstract away details that could contribute to overall more optimal end-to-end scheduling of production, transportation, and distribution of products through the supply chain. The work in this thesis aims to model and integrate some of these details, particularly at the manufacturing plant level. Since the integrated models that have been developed in prior literature have many decision variables and constraints for the real-world-sized problems they aim to optimize, decomposition techniques have been explored for solving a number of the models that have been developed. A second goal of the work in this thesis is to further analyze decomposition techniques that could be used to solve real-world-sized problems.

2.2 Manufacturing Plant

The manufacturing plant problem formulated in this work is a labor allocation and scheduling problem. Each manufacturing plant has a set of workcenters that can each accommodate a minimum and maximum amount of labor. A workcenter is a single station in the assembly line that performs one unit of the total required work for the manufacturing process. Different amounts of labor correspond to different cycle times for production at a given workcenter. A cycle time is the amount of time a workcenter takes to perform one unit of work on one product. Items that need to be produced at each manufacturing plant might require a subset or all of
the workcenters at the plant for production. Precedence constraints exist between workcenters, and for all products, workcenters are visited in the same order, but not all workcenters are necessarily required for all products. The manufacturing plant model assumes deterministic cycle times for given labor allocations, that there is no absolute maximum on the available labor across the problem time horizon, and that workcenters accommodate production of one product at a time. It is also assumed that each day in the time horizon has a constant length in hours, and all work can be scheduled at any time within that number of hours for a given day.

2.2.1 Manufacturing Plant Inputs, Outputs, Objective, and Constraints

The inputs of the manufacturing plant problem are the cycle times corresponding to the different labor allocations at each workcenter, labor allocation minimums and maximums for each workcenter, the problem time horizon, the list of products that need to be produced and their required workcenters, and each product’s deadline. The outputs of the manufacturing plant problem are the labor allocation to the workcenters across the problem time horizon and production schedule for each product being processed at the manufacturing plant.

The objective of the manufacturing plant problem is to minimize the total labor requirement, thereby minimizing cost, and simultaneously to schedule the production of each product as close to its deadline as possible. This will minimize the time it sits in the distribution center at the manufacturing plant to prevent overcrowding of the space. Currently, each term in the multi-objective problem is weighted evenly.

At a high level, the constraints in the manufacturing plant problem include the following:

1. The labor assigned to each workcenter for each day falls between the minimum and maximum for that workcenter.

2. Each workcenter only works on one product at a time.
3. Each product is only being produced at one workcenter at a time.

4. Each product should only be scheduled at each of its required workcenters once in the total time horizon.

5. Each workcenter should completely finish a unit of work before the end of the day (non-preemption).

### 2.2.2 Manufacturing Plant MILP Formulation

The formal manufacturing plant MILP formulation is as follows:

\[
\begin{align*}
\min & \quad \sum_{i \in I, j \in J, l \in L} Y_{ijl} + \sum_{k \in K} Q_k \quad (2.1) \\
A_{ij} & \leq maxLabor_j \quad \forall i \in I, j \in J \quad (2.2) \\
A_{ij} & \geq minLabor_j \quad \forall i \in I, j \in J \quad (2.3) \\
A_{ij} &= \sum_{l \in L} l \cdot Y_{ijl} \quad \forall i \in I, j \in J \quad (2.4) \\
\sum_{l \in L} Y_{ijl} &= 1.0 \quad \forall i \in I, j \in J \quad (2.5) \\
K_{ijkl} &\leq Y_{ijl} \quad \forall i \in I, j \in J, k \in K, l \in L \quad (2.6) \\
K_{ijkl} &\leq F_{ijk} \quad \forall i \in I, j \in J, k \in K, l \in L \quad (2.7) \\
K_{ijkl} &\geq Y_{ijl} + F_{ijk} - 1 \quad \forall i \in I, j \in J, k \in K, l \in L \quad (2.8) \\
S_{ijk} + \sum_{l \in L} cycleTimes_{jl} \cdot workcenters_{kj} \cdot K_{ijkl} &\leq S_{i,j+1,k} \quad \forall i \in I, j \in J, k \in K \quad (2.9)
\end{align*}
\]
\[ S_{ijk} + \sum_{l \in L} \text{cycleTimes}_{jl} \cdot \text{workcenters}_{kj} \cdot K_{ijkl} \geq S_{i,j+1,k} - M_1 \cdot \text{workcenters}_{k,j+1} \quad \forall i \in I, j \in J, k \in K \quad (2.10) \]

\[ \sum_{i \in I} F_{ijk} = 1.0 \quad \forall i \in I, j \in J, k \in K \quad (2.11) \]

\[ G_{ijk} \leq \text{hoursPerShift} \cdot F_{ijk} \quad \forall i \in I, j \in J, lk \in K \quad (2.12) \]

\[ G_{ijk} \leq S_{ijk} \quad \forall i \in I, j \in J, k \in K \quad (2.13) \]

\[ \text{hoursPerShift} \cdot F_{ijk} + S_{ijk} \leq G_{ijk} + \text{hoursPerShift} \quad \forall i \in I, j \in J, k \in K \quad (2.14) \]

\[ G_{ijk} \geq 0 \quad \forall i \in I, j \in J, k \in K \quad (2.15) \]

\[ \sum_{l \in L} \text{cycleTimes}_{jl} \cdot \text{workcenters}_{kj} \cdot Y_{ijl} + \text{workcenters}_{kj} \cdot G_{ijk} \leq \text{hoursPerShift} \quad \forall i \in I, j \in J, k \in K \quad (2.16) \]

\[ S_{ijk} + L_{ijkm} \geq S_{ijm} \quad \forall i \in I, j \in J, k, m \in K \quad (2.17) \]

\[ S_{ijk} + L_{ijkm} \leq S_{ijm} + 2 \cdot \text{hoursPerShift} \cdot D_{2ijklm} \quad \forall i \in I, j \in J, k, m \in K \quad (2.18) \]

\[ S_{ijm} + L_{ijkm} \geq S_{ijk} \quad \forall i \in I, j \in J, k, m \in K \quad (2.19) \]

\[ S_{ijm} + L_{ijkm} \leq S_{ijk} + 2 \cdot \text{hoursPerShift} \cdot D_{1ijklm} \quad \forall i \in I, j \in J, k, m \in K \quad (2.20) \]

\[ D_{1ijklm} + D_{2ijklm} = 1.0 \quad \forall i \in I, j \in J, k, m \in K \quad (2.21) \]

\[ M_2 \cdot \text{workcenters}_{k,j} \cdot F_{ijk} + M_2 \cdot \text{workcenters}_{m,j} \cdot F_{ijm} + \sum_{l \in L} \text{cycleTimes}_{jl} \cdot \text{workcenters}_{kj} \cdot \text{workcenters}_{mj} \cdot Y_{ijl} \leq L_{ijkm} + 2 \cdot M_2 \quad \forall i \in I, j \in J, k, m \in K \quad (2.22) \]

\[ \sum_{i \in I} \text{hoursPerShift} \cdot i \cdot F_{ijk} + \sum_{i \in I} G_{ijk} = P_{jk} \quad \forall j \in J, k \in K \quad (2.23) \]
\begin{align}
F_{i j k} - F_{i, j+1, k} & \leq M_2 \cdot \text{workcenters}_{k,j+1} \quad \forall i \in I, j \in J, k \in K \tag{2.24} \\
W_k &= \sum_{i \in I, l \in L} \text{cycleTimes}_{last,l} \cdot \text{workcenters}_{k, last} \cdot K_{i, last, k,l} + P_{last,k} \quad \forall k \in K \tag{2.25} \\
Q_k + W_k & \leq \text{deadlines}_k + 2 \cdot \text{hoursPerShift} \cdot \text{timeHorizon} \cdot T_1_k \quad \forall k \in K \tag{2.26} \\
Q_k + W_k & \geq \text{deadlines}_k \quad \forall k \in K \tag{2.27} \\
Q_k + \text{deadlines}_k & \leq W_k + 2 \cdot \text{hoursPerShift} \cdot \text{timeHorizon} \cdot T_2_k \quad \forall k \in K \tag{2.28} \\
Q_k + \text{deadlines}_k & \geq W_k \quad \forall k \in K \tag{2.29} \\
T_1_k + T_2_k &= 1.0 \quad \forall k \in K \tag{2.30}
\end{align}

Here, $i \in I$ is a day in total set of days in the problem time horizon $I$, $j \in J$ is a workcenter in the set of all workcenters $J$, $k, m \in K$ are products in the set of all products $K$, and $l \in L$ is labor assignment (number of people assigned) in the set of all possible labor assignments $L$. $A_{i j} \in \mathbb{Z}$ is an integer decision variable describing the amount of labor (number of people) assigned to workcenter $j$ on day $i$. $F_{i j k} \in \{0, 1\}$ is a binary decision variable that takes the value of one if product $k$ is assigned to workcenter $j$ on day $i$, and zero otherwise. $S_{i j k} \in [0, \text{hoursPerShift}]$ is the time product $k$ is scheduled to begin work at workcenter $j$ on day $i$. $P_{j k} \in \mathbb{R}$ the absolute time in the total time horizon that product $k$ is scheduled to begin production at work center $j$, $W_k \in \mathbb{R}$ is the absolute time in the total time horizon that product $k$ is scheduled to complete work at its last workcenter, and $Q_k \in \mathbb{R}$ is the total lateness, or the absolute value of the difference between the deadline time and the absolute end time of product $k$. $K_{i j k} \in \{0, 1\}$, $Y_{i j l} \in \{0, \text{hoursPerShift}\}$, $D_{1 i j k m} \in \{0, 1\}$, $D_{2 i j k m} \in \{0, 1\}$, $L_{i j k m} \in [0, \text{hoursPerShift}]$, $G_{i j k} \in [0, \text{hoursPerShift}]$, $T_1_k \in \{0, 1\}$, and $T_2_k \in \{0, 1\}$ are decision variables used for linearizing constraints. The array $\text{maxLabor}_j$ represents the maximum labor allowed at each workcenter $j$, $\text{minLabor}_j$ is the minimum labor allowed at each workcenter $j$, $\text{cycleTimes}_{j l}$ represents the cycle time that corresponds to a labor allocation of $l$ people to workcenter $j$, $\text{deadlines}_k$ represents the deadlines for each of the products $k$, and $\text{workcenters}_{k,j}$ is a binary-valued input.
representing whether product \( k \) requires workcenter \( j \) for production. \( M_1 \) and \( M_2 \) are large positive integers with \( M_1 < M_2 \). We set \( cycleTimes_{j,0} = M_1 \) for all \( j \), and cycle times corresponding to disallowed labor allocation values for a given workcenter are set to \( M_2 \). Finally, \( hoursPerShift \) is the total number of hours in a shift and \( timeHorizon \) is the total number of days in the time horizon.

Equation 2.1 is the problem objective, minimizing the total labor required across the time horizon and the total lateness across all products. Equations 2.2 and 2.3 impose minimum and maximum labor constraints for each workcenter for each day as in constraint 1. Equation 2.4 is a linearizing constraint that sets \( A_{ij} \) to the correct integer value for given the labor assignment defined by \( Y_{ijl} \). Equation 2.5 ensures that there is only one value assigned for a labor assignment to a workcenter for all days in the time horizon. Equations 2.6-2.8 are linearizing equations that set \( K_{ijkl} = Y_{ijl} \cdot F_{ijk} \). Equations 2.9 and 2.10 ensure that a product is not scheduled at the next workcenter until it is completed at the last workcenter if both workcenters are scheduled to process that product on the same day (enforces constraint 3). Equation 2.11 makes sure each product is only scheduled at each workcenter once in the total time horizon, enforcing constraint 4. Equations 2.12-2.15 are linearizing equations that set \( G_{ijk} = F_{ijk} \cdot S_{ijk} \). Equation 2.16 ensures constraint 5 is met. Equations 2.17-2.21 are linearizing equations setting \( L_{ijkm} = |S_{ijk} - S_{ijm}| \). Equation 2.22 enforces constraint 2. Equation 2.23 sets the absolute start times of each product at each workcenter. Equation 2.24 ensures that workcenters that are not required for a product are assigned to the same day as the last required workcenter for that product. This ensures that the optimal solution can be found with precedence constraints met. Finally, equations 2.25-2.30 set the lateness for each product to the absolute value of the difference between the deadline and the absolute end time for the product.

### 2.2.3 Manufacturing Plant Results and Discussion

The manufacturing plant model was evaluated using a state-of-the-art optimizer, Gurobi, in order to serve as a baseline in the exploration of solve time improvements achieved by exact and approximate decomposition techniques. The model was
tested holding the number of workcenters at the plant, the maximum amount of labor possible at any one workcenter, and the problem time horizon constant at three workcenters, five people per workcenter, and 10 days, respectively. These numbers are based on an example of a small manufacturing line in a manufacturing plant in the Steelcase supply chain. This example consists of three workcenters, although some of their manufacturing lines involve up to 10 workcenters. Workcenters in this manufacturing plant generally accommodate a maximum of three to five workers apiece, and planning is generally done one to two weeks ahead of production of items, or five to 10 business days.

While the number of workcenters, maximum labor, and time horizon parameters were held constant, the number of products was varied between five and 50 since 50 products was the largest problem size for which a majority of problems tested finished within a one hour timeout. Problem sizes of five, 10, 15, 20, 25, 30, 40, and 50 products were tested. For each of these cases, 20 sets of problem parameters were randomly generated as follows: an integer value representing the minimum labor for a workcenter was sampled from a uniform distribution between zero and five. An integer value representing the maximum labor for a workcenter was sampled from a uniform distribution between that workcenter’s minimum labor and five to ensure that is was larger than the minimum value. Cycle times (in hours) corresponding to each possible labor amount for each workcenter were sampled from a uniform distribution between zero and one in order to represent real-world cycle times that are on the order of minutes. Binary values representing which workcenters were required for each product (zero if the workcenter was not required for the product and one if it was) were randomly, uniformly selected. Finally, deadline days for each product were sampled from a uniform distribution between zero and the total number of days in the time horizon (10 days), and deadline times for each product on its deadline day were sampled from a uniform distribution between zero and the number of hours per work day (eight hours).

The problems were tested with a timeout of one hour. The MILP models were solved using Gurobi version 7.5.1 with Java version 1.8.0 on a 2.9 GHz Intel core i5.
processor. The maximum number of products that the model was evaluated on was 50 products, and the results can be seen in figure 2-1. The figure shows the median solve times for each problem size on a log scale with error bars covering the 25%-75% quartile ranges. All 20 problems generated through a size of 40 products were solved in under one hour, and 10 of the 20 problems tested with a size of 50 products were solved in under one hour. The median solve time remains relatively low, under one minute, through 25 products and grows substantially after that. This is likely due to the large increase in the number of binary- and integer-valued decision variables as number of products is increased. Each problem has \((ij + 2ijk + ijl + 3ijk^2)\) binary- and integer-valued variables and \((jk + 2ijk)\) continuous variables. Note that increasing the number of products results in a quadratic increase in the number of decision variables. Considering these results, since it is necessary to schedule production of on the order of thousands of products in the motivating supply chain example, it is only possible to optimize problems approximately two orders of magnitude smaller than necessary for a real-world example with this MILP formulation.

### 2.2.4 Limitations of the Manufacturing Plant Model

While the manufacturing plant modeled in this work describes varying labor and corresponding cycle times at workcenters as well as scheduling of products through each of their required workcenters, there are a number of aspects of the real-world manufacturing plant that are not modeled in this work. To this end, one primary limitation of this model is that variability of cycle times and uncertainty in schedule execution are not considered. Additionally, although product dwell time between workcenters is accounted for in this model, volumetric dwell capacity is not considered. Extending the model to account for these limitations would an interesting direction to explore in the future.
2.3 Consolidation Transport

The consolidation transport model is a vehicle assignment problem. Consolidation transport vehicles are used to ship items from various manufacturing plants and consolidate them at regional distribution centers for eventual delivery to customers in a single shipment. From the distribution center at the manufacturing plant, available items are grouped onto consolidation transport vehicles according to volumetric and weight constraints and shipped to a regional distribution center. Currently, the model accounts for products being shipped from one manufacturing plant to one regional distribution center, although an eventual integrated supply chain model will account for products being shipped from multiple manufacturing plants to multiple regional distribution centers. The consolidation transport model assumes that vehicles leave at three possible departure times in a single day and that products arrive in the distribution center at the manufacturing plant at a fixed time, which in an integrated model corresponds to their completion time at the final workcenter at the manu-
facturing plant. Additionally, it is assumed that products can be always be packed onto a vehicle if volumetric constraints are met and that all consolidation transport vehicles have the same volumetric and weight capacity.

2.3.1 Consolidation Transport Inputs, Outputs, Objective, and Constraints

The inputs to the consolidation transport problem include the problem time horizon, consolidation transport vehicle weight and volume capacities, product sizes and weights, the three daily vehicle departure times, and the product arrival times at the distribution center at the manufacturing plant. The output of the problem is an assignment of products to vehicles across the time horizon. The objective of the consolidation transport problem is to minimize the total number of transport vehicles required, thereby minimizing cost.

At a high level, the constraints in the consolidation transport vehicle problem are the following:

1. A product must be assigned to a transport vehicle that departs the manufacturing plant distribution center after it arrives at that distribution center.

2. Each product is only assigned to one vehicle.

3. Vehicle volume and weight capacities are not exceeded.

2.3.2 Consolidation Transport MILP Formulation

The formal consolidation transport problem MILP formulation is as follows:

$$\min \sum_{i \in I, n \in N, o \in O} V_{ino} \quad (2.31)$$
\[(vehicleDepartureTimes_n - arrivalTimes_{ik}) \cdot B_{inok} \geq 0 \quad \forall i \in I, n \in N, o \in O, k \in K \quad (2.32)\]

\[\sum_{k \in K} \text{productVolumes}_k \cdot B_{inok} \leq vehicleVolumeCapacity \quad \forall i \in I, n \in N, o \in O \quad (2.33)\]

\[\sum_{k \in K} \text{productWeights}_k \cdot B_{inok} \leq vehicleWeightCapacity \quad \forall i \in I, n \in N, o \in O \quad (2.34)\]

\[\sum_{i \in I, n \in N, o \in O} B_{inok} = 1.0 \quad \forall k \in K \quad (2.35)\]

\[M_3 \cdot V_{ino} \geq \sum_{k \in K} B_{inok} \quad \forall i \in I, n \in N, o \in O \quad (2.36)\]

\[V_{ino} + \sum_{l \in K} B_{inok} \leq 0 \quad \forall i \in I, n \in N, o \in O \quad (2.37)\]

Here, \(i \in I\) is a day in total set of days in the problem time horizon \(I\), \(n \in N\) is one of the three departure times \(N\) for a day, \(o \in O\) is a specific vehicle at a departure time of all the vehicles \(O\) that are available at that time, and \(k \in K\) is a product in the total set of products \(K\). \(B_{inok} \in \{0,1\}\) is a binary decision variable representing whether product \(k\) is assigned to vehicle \(o\) at departure time \(n\) on day \(i\). \(V_{ino} \in \{0,1\}\) is a binary decision variable that represents whether vehicle \(o\) at departure time \(n\) on day \(i\) is used or not. The three product departure times within the day are represented by the \(departureTimes_n\) array. The product arrival times at the manufacturing plant distribution facility, which are the times they are available to be shipped on a consolidation transport vehicle, are represented by \(arrivalTimes_{ik}\). The \(productVolumes_k\) and \(productWeights_k\) arrays contain each of the \(k\) products’ weights and volumes. The vehicle volume and weight capacities are the \(vehicleVolumeCapacity\) and \(vehicleWeightCapacity\) values. Finally, \(M_3\) is
a large positive integer.

Equation 2.31 is the objective and minimizes the total number of vehicles used. Equation 2.32 enforces constraint 1. Equations 2.33 and 2.34 ensure constraint 3 is met. Equation 2.35 enforces constraint 2. Finally, equations 2.36 and 2.37 set $V_{ino}$ to zero if no products are assigned to vehicle $o$ at departure time $n$ on day $i$ and one otherwise.

### 2.3.3 Consolidation Transport Results and Discussion

The consolidation transport model was evaluated using a state-of-the-art optimizer, Gurobi, to serve as a step towards developing an integrated model and evaluating where primary bottlenecks are in an integrated model. The model was evaluated for problems with a set time horizon of 10 days (as with the manufacturing plant model), a constant three possible departure times per day with five possible vehicles per departure time. These numbers are true to the motivating supply chain example explored in this work. Additionally, all vehicles were assumed to have the same constant size of 50 cubes (a volumetric unit used in the motivating supply chain example). This number was chosen in conjunction with the range from which product sizes were generated to represent a ratio true to the ratio of vehicle size to product size and subsequent average number of items packed onto a vehicle in the motivating problem. The number of products to be grouped onto vehicles was varied between 50 and 800 since 800 products was the largest problem size for which the majority of problems run finished within a one hour timeout. Problem sizes of 50, 100, 200, 300, 400, 500, 600, 700 and 800 products were tested. For each of these cases, 20 sets of problem parameters were randomly generated as follows: product volumetric sizes were randomly sampled from a uniform distribution between zero and five (with units of cubes). Plant completion days and times for each product, which map to the earliest available times that each product can depart on a consolidation transport vehicle, were sampled from a uniform distribution between zero and the size of the time horizon, 10 days, and from a uniform distribution between zero and the number of hours per shift in a day, eight hours, respectively.
The problems were tested with a timeout of one hour. The MILP models were solved using Gurobi version 7.5.1 with Java version 1.8.0 on a 2.9 GHz Intel core i5 processor. Results can be seen in figure 2-2. The figure shows the median solve times for each problem size on a log scale with error bars covering the 25%-75% quartile ranges. Of the 20 problems generated for each of the problem sizes, all problems with sizes of up to 100 products were solved in under one hour, 18 with a size of 300 products were solved, 17 with a size of 400 products were solved, 19 with a size of 500 products were solved, 15 with a size of 600 products were solved, 12 with a size of 700 products were solved, and 17 with a size of 800 products were solved. The variability in the number of problems solved in under one hour for each problem size is due to the variability in problem parameters generated, leading to more and less constrained problems. There are \((\text{inok} + \text{ino})\) binary decision variables in the consolidation transport model, so solve time increases less dramatically than in the manufacturing plant or distribution center problems. Although it is possible to solve problems involving many more products than the manufacturing plant model with the consolidation transport model, since it will be necessary to schedule production of on the order of thousands of products in the motivating supply chain example, the size of problem that can be solved here is still approximately one order of magnitude smaller than necessary for the real-world supply chain problem.

2.3.4 Limitations of the Consolidation Transport Model

While the consolidation transport model accounts for the vehicle packing and scheduling aspects of the consolidation transport process, there are a number of limitations in this model as compared with the real-world consolidation transport process. Much like the manufacturing plant model, the consolidation transport model is limited in its ability to manage uncertainty in execution, in this case caused by varying departure times. This model also only considers one vehicle size, while in the real-world consolidation transport problem, there are two possible vehicle sizes available for each departure time on each day. In the real-world problem, the number of each type of vehicle to use is chosen based on daily shipping needs. Finally, during particularly
busy weeks, the consolidation transport process occasionally involves a step in which vehicles are packed and then parked at the distribution centers that they are departing from in advance of their actual departure times. This step is not considered in this model.

2.4 Distribution Center

The distribution center problem is a capacity constraint problem with a temporal element. As products flow into and out of the distribution centers over time, facility capacity constraints must be met across the time horizon. There are two types of distribution center including manufacturing plant distribution centers, where items that have completed production are collected to be grouped onto consolidation transport vehicles, and regional distribution centers, where items are collected from consolidation transport vehicles to be shipped to the final customer destination in a single shipment. The distribution center model assumes that if volumetric constraints are
met, items can be packed to fit in the distribution center and that there are no labor constraints associated with the number of items that can be handled in a distribution center. Currently, the model of the distribution center also assumes that there are no assigned arrival times or necessary departure deadlines, and these are instead decision variables in the problem. In the integrated supply chain model, the distribution center arrival times and departure times for each product will correspond to other variables in the supply chain process. The consolidation transport vehicle arrival time will become a product’s arrival time at a regional distribution center and the outbound transport vehicle departure time will be its departure time from the center. The manufacturing plant production completion time will become a product’s arrival time at the manufacturing plant distribution center and the consolidation transport vehicle departure time will be its departure time from that facility. In the absence of these other variables, a minimum dwell time is imposed on products flowing through the distribution center.

2.4.1 Distribution Center Inputs, Outputs, Objective, and Constraints

The inputs to the distribution center problem include the list of products that will pass through the distribution center and their volumetric sizes and the volumetric capacity of the facility. The outputs of the problem are the product arrival and departure times at the distribution center. The objective is to minimize the sum of the wait times at the distribution center for all products. This objective is a stand-in objective for proof of concept, and in the integrated model, deadlines will be taken into consideration, and flow through the distribution center is simply a feasibility problem.

At a high level, the constraints in the distribution center model include the following:

1. The distribution center facility capacity must not be exceeded when a product arrives at the facility.
2. Each product must spend a specified minimum amount of time at the distribution center facility.

Constraint 1 ensures that facility capacity is never exceeded by making sure that capacity constraints are met each time a product arrives at the facility. Although both arrival times and departure times of each product are decision variables in the distribution center model, capacity could only ever be exceeded when a new product arrives at the facility and it was not there before, so this constraint applies only when products arrive. Constraint 2 is a stand-in constraint in the absence of the product manufacturing plant completion times and consolidation transport vehicle departure times that are present in the integrated problem. In the integrated model, product manufacturing plant completion times act as distribution center arrival times and consolidation transport departure times act as distribution center departure times for each product. Constraint 2 ensures that the arrival and departure times are not scheduled at the same time for each product in the absence of these incoming and outgoing constraints, which forces the products to remain at the distribution center for a nonzero amount of time and allows for the proof of concept for ensuring capacity constraints are met over time.

### 2.4.2 Distribution Center MILP Formulation

The formal distribution center problem MILP formulation is as follows:

\[
\min \sum_{k \in K} (U_k - T_k) \tag{2.38}
\]

\[
Z_{km} \leq M_4 \cdot X_{km} \quad \forall \, k, m \in K \tag{2.39}
\]

\[
Z_{km} \leq T_m \quad \forall \, k, m \in K \tag{2.40}
\]

\[
T_m + M_4 \cdot X_{km} - Z_{km} \leq M_4 \quad \forall \in K \tag{2.41}
\]

\[
Z_{km} \geq 0 \quad \forall \, k, m \in K \tag{2.42}
\]

\[
R_{km} - M_4 \cdot X_{km} \leq 0 \quad \forall \, k, m \in K \tag{2.43}
\]
\[ R_{km} - U_m \leq 0 \quad \forall k, m \in K \] (2.44)

\[ U_m + M_4 \cdot X_{km} - R_{km} \leq M_4 \quad \forall k, m \in K \] (2.45)

\[ R_{km} \geq 0 \quad \forall k, m \in K \] (2.46)

\[ Z_{km} - T_k \leq 0 \quad \forall k, m \in K \] (2.47)

\[ M_4 \cdot X_{km} + T_k - R_{km} \leq M_4 \quad \forall k, m \in K \] (2.48)

\[ M_4 \cdot H_{km} - T_k + T_m \leq M_{\text{Ambox}} \quad \forall k, m \in K \] (2.49)

\[ M_4 \cdot H_{km} - T_k + T_m \geq \epsilon \quad \forall k, m \in K \] (2.50)

\[ M_4 \cdot J_{km} - U_m + T_k \leq M_4 \quad \forall k, m \in K \] (2.51)

\[ M_4 \cdot J_{km} - U_m + T_k \geq \epsilon \quad \forall k, m \in K \] (2.52)

\[ H_{km} + J_{km} - X_{km} \leq 1 \quad \forall k, m \in K \] (2.53)

\[ X_{km} \leq 1.0 \quad \forall k, m \in K \] (2.54)

\[ \sum_{m \in K} \text{productSizes}_m \cdot X_{km} \leq \text{facilitySize} \quad \forall k \in K \] (2.55)

\[ U_m - T_m \geq \text{minTime} \quad \forall m \in K \] (2.56)

Here, \( k, m \in K \) are products in the total set of products \( K \). \( T_k \in [0, \text{hoursPerShift}] \) is a decision variable that represents the arrival time of product \( k \) at the distribution center. \( U_k \in [0, \text{hoursPerShift}] \) is a decision variable representing the departure time of product \( k \) at the distribution center. \( X_{km} \in \{0, 1\} \) is a linearizing binary decision variable that takes the value of one if product \( m \) is at the distribution center when product \( k \) arrives and zero otherwise. \( R_{km} \in \mathbb{Z}, \ Z_{km} \in \mathbb{Z}, \ H_{km} \in \{0, 1\}, \) and \( J_{km} \in \{0, 1\} \) are linearizing decision variables used to linearize constraints.

Equation 2.38 is the objective and minimizes the total product dwell time at the distribution center across all products. Equations 2.39-2.41 are linearizing equations to set \( Z_{km} = X_{km} \cdot T_m \). Equations 2.42-2.46 are linearizing equations to set \( R_{km} = X_{km} \cdot U_m \). Equation 2.47 ensures that \( X_{km} = 0 \) if product \( k \) arrives at the distribution
center before product $m$. Equation 2.48 ensures that $X_{km} = 0$ if product $m$ leaves the
distribution center before product $k$ arrives. Equations 2.49-2.54 ensure that $X_{km} = 1$
if product $m$ is at the distribution center when product $k$ arrives and product $m$ leaves
the distribution center after product $k$ arrives. Equation 2.55 enforces constraint 1,
and equation 2.56 enforces constraint 2.

2.4.3 Distribution Center Results and Discussion

Like the consolidation transport model, the distribution center model was evaluated
using a state-of-the-art optimizer, Gurobi, to serve as a step towards developing
an integrated model and evaluating where primary bottlenecks are in an integrated
model. The distribution center model was evaluated with a constant volumetric size
of 50 cubes (the unit used in the motivating supply chain example) while varying the
number of products from one through the maximum for which a majority of problems
could be solved in a time limit of one hour, which was 11. The volumetric size for the
distribution center is set such that the ratio of facility size to product size and the
number of products that fit in the facility at any given time is lower than is true to
the motivating supply chain example. This was such that some small problems, like
the ones included in the results for this section, could be run and completed before
a one hour timeout and the trends of the results could be obtained. Problem sizes
of two, five, eight, nine, 10, and 11 products were tested. For each of these cases, 20
sets of problem parameters were randomly generated including product sizes sampled
from a uniform distribution between zero and five (with units of cubes).

The problems were tested with a timeout of one hour. The MILP models were
solved using Gurobi version 7.5.1 with Java version 1.8.0 on a 2.9 GHz Intel core i5
processor. Results can be seen in figure 2-3. The figure shows the median solve times
for each problem size on a log scale with error bars covering the 25%-75% quartile
ranges. Of the 20 problems generated for each problem size, all problems with sizes
of up to nine products were solved in under one hour, 18 problems with a size of
10 products were solved, and 16 problems with a size of 11 products were solved.
This problem has $3k^2$ binary decision variables and $(2k + 2k^2)$ continuous decision
variables. The quadratic increase in number of decision variables likely corresponds to the dramatic increase in solve time as number of products is increased. Here, the maximum problem size that is solvable within an hour timeframe is 11 products, two orders of magnitude smaller than the real-world problem sizes in the motivating supply chain example. Since this problem size is smaller than the problems solved by both the manufacturing plant and consolidation transport models, the distribution center aspect of the integrated model is likely the bottleneck in the size of problem that can be solved with that model.

### 2.4.4 Limitations of the Distribution Center Model

The distribution center model accounts for capacity constraints over time as products move into and out of distribution center facilities. As with the previously discussed models, the distribution center model is limited in its ability to account for uncertainty in execution times as products arrive and depart facilities. It is also limited in the fact that it does not account for the labor requirements associated with moving products.
into and out of distribution centers, involving the unloading and reloading of products onto transport vehicles. In the real-world, labor requirements could prove to be a bottleneck in addition to facility capacity.

### 2.5 Integrated Model

The integrated model integrates one manufacturing plant, one plant distribution center, and one set of consolidation transport vehicles scheduled from the plant distribution center to one regional distribution center. There are constraints that connect each of the models together, setting the completion time at the manufacturing plant to distribution center arrival time and the consolidation transport vehicle departure time to the distribution center departure time for each product. In the integrated model, there is no longer a constraint on minimum stay time at the distribution center for each product since arrival and departure times are determined by the other steps in the process previously mentioned. Therefore, equation 2.56 is not present in the integrated model.

Similar to the manufacturing plant model, the objective in the integrated model is to minimize the lateness, or the time before or past the deadline that the product is completed, and total labor required across the time horizon. In this case, the deadline for each product corresponds to a deadline for departing the distribution center on a consolidation transport vehicle, as opposed to it being a manufacturing plant completion time as was the case in the manufacturing plant model. The objective can be expressed as:

\[
\min \sum_{i \in I, j \in J, l \in L} Y_{ijl} + \sum_{k \in K} E_k
\]

The linking constraints that are added include:

\[
\sum_{i \in I, n \in N, o \in O} \left(\text{hoursPerShift} \cdot \left\lfloor \frac{n}{3} \right\rfloor + \text{departureTimes}_n \mod 3 \right) \cdot B_{inok} - C_k = 0
\]

\[\forall k \in K (2.58)\]
\[ E_k + C_k \leq \text{deadlines}_k + 2 \cdot \text{hoursPerShift} \cdot \text{timeHorizon} \cdot T_{1k} \quad \forall k \in K \quad (2.59) \]

\[ E_k + C_k \geq \text{deadlines}_k \quad \forall k \in K \quad (2.60) \]

\[ E_k + \text{deadlines}_k \leq C_k + 2 \cdot \text{hoursPerShift} \cdot \text{timeHorizon} \cdot T_{2k} \quad \forall k \in K \quad (2.61) \]

\[ E_k + \text{deadlines}_k \geq C_k \quad \forall k \in K \quad (2.62) \]

\[ T_{1k} + T_{2k} = 1.0 \quad \forall k \in K \quad (2.63) \]

As in the manufacturing plant model, \( W_k \in \mathbb{R} \) is the absolute time in the total time horizon that product \( k \) is completed at the manufacturing plant. \( W_k \) replaces \( T_k \) from the distribution center model in the integrated model, since the completion time at the manufacturing plant is equivalent to the arrival time at the distribution center. \( C_k \in \mathbb{R} \) is the absolute time in the total time horizon that product \( k \) leaves the distribution center at the manufacturing plant when the consolidation transport vehicle that it is assigned to departs. \( E_k \in \mathbb{R} \) is the absolute value of the difference between the deadline time for each product \( k \) and the time that the product leaves the distribution center. Constraint 2.58 sets \( C_k \) to the correct value, and constraints 2.59-2.63 set \( E_k \) to the correct lateness value. Note that these constraints replace constraints 2.26-2.30 in the manufacturing plant model.

### 2.5.1 Integrated Model Results and Discussion

The integrated model was evaluated using a state-of-the-art optimizer, Gurobi, in order to serve as a baseline in exploration of solve time improvements achieved by exact and approximate decomposition techniques. It was evaluated holding the number of workcenters at the manufacturing plant constant at three workcenters, the maximum labor at a workcenter constant at five people, the time horizon constant at 10 days, the sizes of both the consolidation transport vehicles and the distribution center constant at 50 cubes, and the time per shift in a day constant at eight hours. The reasons for these numbers are specified in the previous sections describing the individual models. The number of products was varied from five to the maximum
number for which the majority of problems tested finished within a one hour timeout, which was 15. Problem sizes of five, nine, 10, 11, 12, 13, 14, and 15 products were tested. As with the models detailed in previous sections, 20 sets of problem parameters including minimum and maximum labor at each workcenter, cycle times for the workcenters based on different labor allocations, required workcenters for each product, product sizes, and product deadline days and times were randomly generated for each case. These parameters were sampled from the same distributions stated in the previous sections.

The problems were tested with a timeout of one hour. The MILP models were solved using Gurobi version 7.5.1 with Java version 1.8.0 on a 2.9 GHz Intel core i5 processor. Results can be seen in figure 2-4. The figure shows the median solve times for each problem size on a log scale with error bars covering the 25%-75% quartile ranges. Of the 20 problems generated for each problem size, all problems with sizes of up to 10 products were solved in under one hour, 16 problems with a size of 11 products were solved, 18 problems with a size of 12 products were solved, 14 problems with a size of 13 products were solved, 17 problems with a size of 14 products were solved, and 14 problems with a problem size of 15 products were solved. Like with the consolidation transport model, variability in the number of problems solved for each problem size is a function of the problem parameters that were generated. As with the manufacturing plant model and distribution center model, the number of binary- and integer-valued decision variables in the integrated model varies quadratically with $k$ and as with the distribution center model, the number of continuous decision variables also varies quadratically with $k$. Note that due to the added complexity of the integrated model, smaller problems can be solved within one hour than those solved with the manufacturing plant and consolidation transport models. However, slightly larger problems can be solved with the integrated model than with the distribution center model. This is likely due to the fact that aspects of product flow through the distribution center are more constrained by product completion times at the manufacturing plant on one end and the departure times of the consolidation transport vehicles on the other, reducing the search space somewhat. The maximum problem
size that can consistently be solved with the integrated model, 15 products, remains two orders of magnitude smaller than those that need to be solved in the real-world motivating supply chain example.

### 2.5.2 Limitations of the Integrated Model

Since the integrated model combines the three supply chain component models discussed previously, it exhibits the same limitations as its constituent models. Additionally, it is limited by the fact that here, there is only one plant, one distribution center, and one set of consolidation transport vehicles considered. In the real-world supply chain problem, multiple manufacturing plants and distribution centers are connected by multiple sets of consolidation transport vehicles. Finally, without the outbound transport component integrated, the ability to optimize the supply chain with end-to-end information is inherently limited.
2.6 Future Modeling and Integration Work

So far, the manufacturing plant, consolidation transport, and distribution center models have been developed and integrated. As part of future work, an outbound transport model will be developed. This model will involve grouping products onto delivery vehicles according to volumetric and weight constraints and routing the vehicles to customer destinations. Additionally, all items being shipped to a single customer should be grouped onto a single vehicle, and items from different orders (being shipped to different customer destinations) should be grouped onto vehicle to minimize lateness in delivery time. Once the outbound transport model is complete, it will be integrated into the overall supply chain model and tested. Finally, integrated models with varying numbers of manufacturing plants and regional distribution centers will be developed to explore the entire supply chain problem.
Chapter 3

Exact Techniques

Exact techniques are applied to problems modeled using the MILP framework in order to solve for their true optimal solutions to some tolerance. Gurobi, the optimizer used in this work, solves the MILP models detailed in chapter 2 to a MIP gap tolerance of $10^{-4}$. In particular, Gurobi uses a branch-and-bound algorithm in combination with either the simplex method or interior point methods in order to attain the optimal solution [3]. There exist a number of other exact decomposition techniques that can be used to solve large-scale models implemented using the MILP framework more quickly than the techniques currently used by standard solvers, such as Gurobi, if the models have a special structure. These techniques draw on the decomposable structure of various MILP problem formulations. A few such techniques are detailed in the following sections, and one technique is implemented with the manufacturing plant model to serve as a baseline in investigating whether such techniques can improve solve times for this formulation of the supply chain problem.

3.1 Related Work

Two exact decomposition techniques that have been applied to large-scale problems modeled using the MILP framework are Benders decomposition (Benders [4]) and Dantzig-Wolfe decomposition (Dantzig and Wolfe [12]). Benders decomposition is a technique that decomposes MILP variables into two groups, creating a master prob-
lem and one or multiple subproblems. Through the solving of the subproblem(s), feasibility and optimality cuts are iteratively added to the master problem until the algorithm converges to an optimal solution. Feasibility cuts are added to the master problem to eliminate from the search space master problem solutions that have resulted in infeasible subproblems, indicating overall problem infeasibility. Optimality cuts are added to ensure that as feasible subproblem solutions are found, no worse solution is attempted at the master problem level in future iterations. Benders decomposition is also known as “row generation” since new constraints are added to the master problem at each step. Dantzig-Wolfe decomposition is similar but is essentially the dual of the Benders decomposition technique in that instead, the master problem retains constraints involving many to most of the problem decision variables, and solutions to the subproblem(s) contribute proposals involving combinations of assigned decision variable values to the master problem. Dantzig-Wolfe decomposition is also known as “column generation” since new decision variables representing each proposal are added to the problem at every step.

Accordingly, Benders decomposition typically performs better for problems that have many “complicating variables”, variables that appear in many of the original problem’s constraints, and Dantzig-Wolfe decomposition performs better in problems with many “complicating constraints”, constraints that contain many of the original problem’s variables (Rahmaniani et al. [23], Taşkin [27]). By decomposing the problem into a master and subproblem in each case, the computational burden is reduced.

The supply chain model components as defined in chapter 2 have both complicating variables and complicating constraints. Benders decomposition and its variants has been explored extensively in the literature and applied to a diverse set of problems in planning and scheduling (Canto [7], Hooker [17]), vehicle routing (Cordeau et al. [10], Corrêa, Langevin, and Rousseau [11]), and resource management (Cai et al. [6], Kim, Wu, and Huang [18]) among others. Therefore it is implemented in this work as a baseline technique in the exploration of algorithmic improvements achieved by decomposition techniques over standard MILP solvers. Dantzig-Wolfe decomposition
is left as part of future work.

3.2 Benders Decomposition

The following is an overview of classical Benders decomposition and specifics of the implementation for the manufacturing plant model as part of this work. Over the years, many variants of Benders decomposition have evolved for different types of problem structure. For example, logic-based Benders decomposition can be used when subproblems are easily formulated as constraint programs, nonlinear programs, or feasibility-checking problems, generalized Benders decomposition is applicable when subproblems are best formulated as mixed integer nonlinear programs, and other variants of Benders decomposition apply particularly to cases in which decomposing the problem with integer subproblems works best (Rahmaniani et al. [23]). Since the manufacturing plant problem is formulated as a MILP in chapter 2 and it is possible to decompose the MILP into an integer master problem and a continuous subproblem, classical Benders decomposition is implemented here to serve as a baseline for exploration of decomposition techniques.

3.2.1 Overview

In classical Benders decomposition, a full MILP problem formulation is decomposed into two primary groups: a master problem with binary- and integer-valued decision variables and the constraints containing only these variables and one or multiple linear program (LP) subproblems including the remainder of the variables and constraints. The master problem is solved, and an incumbent integer-feasible solution is found. The integer values assigned as part of this integer-feasible solution are set as constants for the solving of the subproblem, and one of three outcomes results. If the subproblem is infeasible, indicating that the integer solution that was found in the master problem is infeasible in the overall problem, a feasibility cut is added to the master problem such that the same integer solution will not be explored again. If the subproblem is unbounded, the overall problem is unbounded, and the algorithm terminates without
a solution. If the subproblem has an optimal solution and the algorithm has not yet converged, indicating that the overall optimal solution has not yet been found, an optimality cut, or Benders cut is added to the master problem such that no worse solution is explored in the future. If the master problem is found to be infeasible at any point, the overall problem is infeasible, and if it is found to be unbounded, the overall problem is unbounded.

In classical Benders decomposition, the subproblem is formulated as its dual, which gives an equivalent objective solution to the primal of the subproblem by the strong duality property of LP problems. This allows cuts to be added to the master problem as a function of the master problem’s decision variables, as will be demonstrated in the general formulation below. It is important to note that unboundedness of the subproblem is equivalently infeasibility of the dual subproblem, and infeasibility of the subproblem is equivalently unboundedness of the dual subproblem.

As the algorithm progresses, the master problem solution provides a lower bound for a minimization problem or an upper bound for a maximization problem. The subproblem solution augmented with the master problem objective value for the current integer variable assignment provides an upper bound for a minimization problem or a lower bound for a maximization problem. This falls from the fact that the master problem is a relaxed version of the overall problem, and every subproblem solution found, if feasible given the current integer decision variable assignment is suboptimal until the algorithm converges. The master and subproblems are iteratively solved until the master and augmented subproblem objectives are equivalent (the convergence criteria).

For Benders decomposition, the general formulation of a MILP minimization prob-
Problem can be stated as:

\[
\begin{align*}
\text{minimize } & \; c^T x + f^T y \\
\text{subject to } & \; A x + B y \geq b \\
& \; y \in Y \\
& \; x \geq 0
\end{align*}
\] (3.1)

Here, \( x \) and \( y \) are the problem decision variables, where \( x \) are the continuous decision variables and \( y \) are the integer decision variables. In this formulation, \( c, f, \) and \( b \) are positive constant vectors, and \( A \) and \( B \) are constant matrices. The master problem can be written:

\[
\begin{align*}
\text{minimize } & \; f^T y + \theta \\
\text{subject to } & \; y \in Y \\
& \; \theta \geq (b - B y)^T \bar{u}_o \quad \forall \; o \in O \quad (3.7) \\
& \; (b - B y)^T \bar{u}_f \leq 0 \quad \forall \; f \in F \quad (3.8)
\end{align*}
\]

In the master problem, \( \theta \) is the subproblem contribution to the overall objective. In the initialization step of the algorithm when the masters problem is solved without any Benders or feasibility cuts, we will have an initial \( \theta = 0 \). As the algorithm progresses, for a minimization problem, \( \theta \) is bounded from below by the dual subproblem values that give the current best dual objective in conjunction with the master problem variables that comprise the objective of the dual subproblem. Since in Benders decomposition we consider the dual of the subproblem, the best of all subproblem proposals (for a minimization primal subproblem and the equivalent maximization dual) will be the maximum of all dual objectives found so far. Therefore, \( \theta \) in the master problem has a lower bound of the best (highest) of all subproblem objectives so far, which eliminates any worse solution than those that have already been evaluated. Equation 3.7 is the set of optimality, or Benders cuts, that bound \( \theta \) from below as described. \( \bar{u}_o \) is the set optimal subproblem dual decision variable values for each
subproblem solution, and \( o \in O \) is the set of all dual vectors for all the optimality cuts added so far.

Equation 3.8 is the set of feasibility cuts, \( \bar{u}_f \) is the set of dual subproblem unbounded ray values from each infeasible subproblem (or equivalently, each unbounded dual subproblem), and \( f \in F \) is the set of all unbounded rays from all unbounded dual subproblems so far. The feasibility cut falls from the fact that \((b - By)^T \bar{u}_f\) is equivalent to the subproblem primal objective, \( c^T x \), by strong duality. Since \( c^T \) is a vector of positive constants and all values \( x \) must be positive according to equation 3.4, requiring \((b - By)^T \bar{u}_f\) to be positive ensures the same integer-feasible solution will not be explored again.

Note that when solving the subproblem directly using an optimizer and then extracting the dual values, the dual values are also known as pi attributes in solvers such as Gurobi. When solving the primal subproblem and relying on strong duality present in LP problems, the extreme ray values are also called Farkas duals in optimizers such as Gurobi.

The subproblem is expressed as:

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax \geq b - By \\
& \quad x \geq 0
\end{align*}
\]  

(3.9) (3.10) (3.11) (3.12)

In this work, we solve the primal subproblem and rely on strong duality to extract dual values rather than formulating and solving the dual problem directly. However, it is informative to formulate the dual as part of the general formulation of Benders decomposition so that it is clear where the feasibility and optimality cuts are derived.
The dual of the subproblem is written as:

$$\begin{align*}
\text{maximize} & \quad (b - B\bar{y})u \\
\text{subject to} & \quad A^T u \leq c \\
& \quad u \geq 0
\end{align*}$$

(3.13) \quad (3.14) \quad (3.15) \quad (3.16)

---

**Algorithm 1 Benders Decomposition**

1. **Inputs:** $A, B, b, c, f, \epsilon$
2. **Outputs:** $\bar{y}, \bar{\theta}$
3. Initialize $UB = \infty$
4. Initialize $LB = -\infty$
5. **Solve initial master problem, and set $\bar{y}$**
6. **while** $UB - LB > \epsilon$ **do**
7. **Solve subproblem**
8. **if** feasible **then**
9. \hspace{1em} $UB \leftarrow f(\bar{y}) + (b - B\bar{y})\bar{u}_o$
10. \hspace{1em} Add optimality cut $\theta \geq (b - By)^T\bar{u}_o$ to master problem
11. **else if** infeasible **then**
12. \hspace{1em} Add feasibility cut $(b - By)^T\bar{u}_f \leq 0$ to master problem
13. **else if** unbounded **then**
14. \hspace{1em} **break**
15. **Solve master problem**
16. **if** infeasible **then**
17. \hspace{1em} **break**
18. **else**
19. \hspace{1em} $LB \leftarrow f(\bar{y}) + \bar{\theta}$

The pseudocode for the classical Benders decomposition algorithm is in algorithm 1. The algorithm takes as input problem parameters $A, B, b, c, f$ as described in the Benders decomposition formulation in this section and a tolerance $\epsilon$ (line 1). It outputs the objective value comprised of the assigned $\bar{y}$ and $\bar{\theta}$ decision variable values (line 2). Lines 3 and 4 initialize the upper bound to infinity and lower bound to negative infinity, respectively. In line 5, the initial master problem is solved in order to set an initial integer-feasible solution. Lines 6-19 constitute the main loop of the algorithm and run until the upper and lower bounds are within a tolerance $\epsilon$ of each
other. In line 7 the subproblem is solved. As indicated in lines 8-10, if the primal subproblem is feasible, the upper bound is updated to the current overall objective given the master and subproblem solutions and an optimality cut is added to the master problem. In lines 11-12, if the primal subproblem is infeasible, a feasibility cut is added to the master problem. In lines 13-14, if the primal subproblem is unbounded, the overall problem is unbounded, so the algorithm terminates without a solution. In line 15, the master problem is called again with the added cuts. In lines 16-17, if the master problem is infeasible, then the overall problem is infeasible, and the algorithm terminates without a solution. Finally, in lines 18-19 the lower bound is updated based on the new master problem solution.

3.2.2 Benders Decomposition Applied to the Manufacturing Plant Problem

To study the merits of decomposition techniques like Benders decomposition for the supply chain problem, Benders decomposition was applied to the manufacturing plant problem, as detailed in chapter 2. The manufacturing plant model was used for inspection in this work, since it is has the largest number of decision variables for a given problem size (number of products to be manufactured and delivered) of any of the models developed.

The master problem for the manufacturing plant problem can be expressed as:

\[
\min \sum_{i \in I, j \in J} Y_{ijl} \tag{3.17}
\]

\[
A_{ij} \leq \text{maxLabor}_j \quad \forall i \in I, j \in J \tag{3.18}
\]

\[
A_{ij} \geq \text{minLabor}_j \quad \forall i \in I, j \in J \tag{3.19}
\]

\[
A_{ij} = \sum_{l \in L} l \cdot Y_{ijl} \quad \forall i \in I, j \in J \tag{3.20}
\]

\[
\sum_{l \in L} Y_{ijl} = 1.0 \quad \forall i \in I, j \in J \tag{3.21}
\]
The master problem formulation includes all binary- and integer-valued decision variables in the manufacturing plant problem. As with the problem formulation in chapter 2, \( i \in I \) is a day in the total set of days in the problem time horizon \( I \), \( j \in J \) is a workcenter in the set of all workcenters \( J \), \( k, m \in K \) are products in the set of all products \( K \), and \( l \in L \) is labor assignment (number of people assigned) in the set of all possible labor assignments \( L \). \( A_{ij} \in \mathbb{Z} \) is an integer decision variable describing the amount of labor (number of people) assigned to workcenter \( j \) on day \( i \). \( F_{ijk} \in \{0,1\} \) is a binary decision variable that takes the value of one if product \( k \) is assigned to work center \( j \) on day \( i \) and zero otherwise. \( K_{ijkl} \in \{0,1\} \), \( Y_{ijl} \in \{0,\text{hoursPerShift}\} \), \( D1_{ikm} \in \{0,1\} \), \( D2_{ikm} \in \{0,1\} \), \( T1_k \in \{0,1\} \), and \( T2_k \in \{0,1\} \) are decision variables used for linearizing constraints. The array \( \text{maxLabor}_j \) represents the maximum labor allowed at each workcenter, \( \text{minLabor}_j \) is the minimum labor allowed at each workcenter, \( \text{workcenters}_{kj} \) is a binary-valued input representing whether product \( k \) requires workcenter \( j \) for production, and \( M_2 \) is a large positive integer. Equation 3.17 is the part of the problem objective minimizing the total labor required across the time horizon. Equations 3.18 and 3.19 impose minimum and maximum labor constraints for each workcenter for each day. Equation 3.20
is a linearizing constraint that sets $A_{ij}$ to the correct integer value for given the labor assignment defined by $Y_{ijl}$. Equation 3.21 ensures that there is only one value assigned for a labor assignment to a workcenter for all days in the time horizon. Equation 3.22 makes sure each product is only scheduled at each workcenter once in the total time horizon. Equations 3.23-3.25 are linearizing equations that set $K_{ijkl} = Y_{ijl} \cdot F_{ijk}$. Equation 3.27 ensures that workcenters that are not required for a product are assigned to the same day as the last required workcenter for that product. Equations 3.26 and 3.28 are other supporting linearizing constraints, as discussed in chapter 2. Equation 3.29 is the set of Benders cuts, or optimality cuts, and equation 3.30 is the set of feasibility cuts. The full expressions for these cuts are written out in appendix A. These are derived from the optimal solutions to the duals of the feasible subproblems or the extreme rays of the unbounded subproblems, respectively. A new Benders cut or feasibility cut is added with each iteration of the algorithm. The subproblem for the manufacturing plant problem can be written:

$$
\min \sum_{k \in K} Q_k
$$

(3.31)

$$
S_{ijk} - S_{i,j+1,k} \leq - \sum_{l \in L} \text{cycleTimes}_{jl} \cdot \text{workcenters}_{kj} \cdot K_{ijkl}
\forall i \in I, j \in J, k \in K
$$

(3.32)

$$
S_{i,j+1,k} - S_{ijk} \leq \sum_{l \in L} \text{cycleTimes}_{jl} \cdot \text{workcenters}_{kj} \cdot K_{ijkl} + M_1 \cdot \text{workcenters}_{k,j+1}
\forall i \in I, j \in J, k \in K
$$

(3.33)

$$
\text{workcenters}_{kj} \cdot \bar{F}_{ijk} \cdot S_{ijk} \leq \text{hoursPerShift} - \sum_{l \in L} \text{cycleTimes}_{jl} \cdot \text{workcenters}_{kj} \cdot \bar{Y}_{ijl}
\forall i \in I, j \in J, k \in K
$$

(3.34)
\[ S_{ijm} - S_{ijk} - L_{ijkm} \leq 0 \quad \forall i \in I, j \in J, k, m \in K \]  
(3.35)

\[ S_{ijk} - S_{ijm} + L_{ijkm} \leq 2 \cdot \text{hoursPerShift} \cdot D_{ijkm} \quad \forall i \in I, j \in J, k, m \in K \]  
(3.36)

\[ S_{ijk} - S_{ijm} - L_{ijkm} \leq 0 \quad \forall i \in I, j \in J, k, m \in K \]  
(3.37)

\[ S_{ijm} - S_{ijk} - L_{ijkm} \leq 0 \quad \forall i \in I, j \in J, k, m \in K \]  
(3.38)

\[-L_{ijkm} \leq 2 \cdot M_2 - M_2 \cdot \text{workcenters}_{kj} \cdot \bar{F}_{ijk} - M_2 \cdot \text{workcenters}_{mj} \cdot \bar{F}_{ijm} \]
\[ - \sum_{l \in L} \text{cycleTimes}_{jl} \cdot \text{workcenters}_{kj} \cdot \text{workcenters}_{mj} \cdot \bar{Y}_{ijl} \]
\[ \forall i \in I, j \in J, k, m \in K \]  
(3.39)

\[ \sum_{i \in I} \bar{F}_{ijk} \cdot S_{ijk} - P_{jk} \leq - \sum_{i \in I} \text{hoursPerShift} \cdot i \cdot \bar{F}_{ijk} \quad \forall j \in J, k \in K \]  
(3.40)

\[ P_{jk} - \sum_{i \in I} \bar{F}_{ijk} \cdot S_{ijk} \leq \sum_{i \in I} \text{hoursPerShift} \cdot i \cdot \bar{F}_{ijk} \quad \forall j \in J, k \in K \]  
(3.41)

\[ W_k - P_{\text{last},k} \leq \sum_{i \in L, l \in L} \text{cycleTimes}_{\text{last},l} \cdot \text{workcenters}_{k,\text{last}} \cdot \bar{F}_{i,\text{last},k} \cdot \bar{Y}_{i,\text{last},l} \]
\[ \forall k \in K \]  
(3.42)

\[ P_{\text{last},k} - W_k \leq - \sum_{i \in L, l \in L} \text{cycleTimes}_{\text{last},l} \cdot \text{workcenters}_{k,\text{last}} \cdot \bar{F}_{i,\text{last},k} \cdot \bar{Y}_{i,\text{last},l} \]
\[ \forall k \in K \]  
(3.43)

\[ Q_k + W_k \leq \text{deadlines}_k + 2 \cdot \text{hoursPerShift} \cdot \text{timeHorizon} \cdot T_{1_k} \quad \forall k \in K \]  
(3.44)

\[-Q_k - W_k \leq - \text{deadlines}_k \quad \forall k \in K \]  
(3.45)

\[ Q_k - W_k \leq 2 \cdot \text{hoursPerShift} \cdot \text{timeHorizon} \cdot T_{2_k} - \text{deadlines}_k \quad \forall k \in K \]  
(3.46)

\[ W_k - Q_k \leq \text{deadlines}_k \quad \forall k \in K \]  
(3.47)
The dual of the subproblem is written out in appendix A. For this work, the primal subproblem is formulated and solved, the dual values are extracted in the case of a feasible subproblem, and Farkas values are extracted for an infeasible subproblem, corresponding to an unbounded dual subproblem. These values are used to build the optimality and feasibility cuts.

As with the problem formulation in chapter 2, \( S_{ijk} \in [0, \text{hoursPerShift}] \) is the time product \( k \) is scheduled to begin work at workcenter \( j \) on day \( i \), \( P_{jk} \in \mathbb{R} \) the absolute time in the total time horizon that product \( k \) is scheduled to begin production at workcenter \( j \), \( W_k \in \mathbb{R} \) is the absolute time in the total time horizon that product \( k \) is scheduled to complete work at its last workcenter, and \( Q_k \in \mathbb{R} \) is the total lateness, or the absolute value of the difference between the deadline time and the absolute end time of product \( k \). \( L_{ijkm} \in [0, \text{hoursPerShift}] \) is a decision variable used for linearizing a constraint. The matrix \( \text{cycleTimes}_{jl} \) represents the cycle time that corresponds to a labor allocation of \( l \) people to workcenter \( j \), \( \text{deadlines}_k \) represents the deadlines for each of the products \( k \), \( \text{hoursPerShift} \) is the total number of hours in a shift, \( \text{timeHorizon} \) is the total number of days in the time horizon, and \( M_1 \) is a large positive integer. Equation 3.31 is the part of the objective minimizing lateness. Equations 3.32 and 3.33 ensure that a product is not scheduled at the next workcenter until it is completed at the last workcenter if both workcenters are scheduled to process that product on the same day. Equation 3.34 ensures that each workcenter should completely finish a unit of work before the end of the day. Equations 3.35-3.38 are linearizing equations setting \( L_{ijkm} = |S_{ijk} - S_{ijm}| \). Equation 3.39 enforces that each workcenter only works on one product at a time. Equations 3.40-3.41 set the absolute start times of each product at each workcenter. Finally, equations 3.42-3.47 set the lateness for each product to the absolute value of the difference between the deadline and the absolute end time for the product. Note that this decomposition of the problem allows us to eliminate a number of the decision variables required for linearization, including all \( G_{ijk} \) variables. This eliminates \( i \cdot j \cdot k \) continuous decision variables from the problem formulation. Results are detailed in 3.3.
3.3 Results and Discussion

In order to test the performance of the Benders decomposition formulation for the manufacturing plant problem, the same set of randomly generated problem parameters used for the original manufacturing plant MILP in chapter 2 was applied to the Benders formulation. As with the MILP formulations in chapter 2, the master and subproblem formulations of the Benders problem were solved using the Java implementation of Gurobi [16]. The problems were tested with a timeout of one hour, and the models were solved using Gurobi version 7.5.1 with Java version 1.8.0 on a 2.9 GHz Intel core i5 processor. Of the 20 problems with the smallest problem size of 10 days in the time horizon and five products, none finished in under one hour with the Benders formulation. Since none of the smallest problems converged before the set time out of one hour, none of the larger problems were tested with the Benders formulation. Instead, to demonstrate the difference in solve time between the original MILP with Gurobi and Benders decomposition, 20 randomly generated problems for a problem size of two days in the time horizon and three products were tested with both the Benders formulation and with Gurobi. Problem parameters were generated in the same way as in chapter 2. Of the 20 problems tested, 19 converged in under one hour with the Benders implementation, all 20 converged in under one hour with the original MILP formulation, and the averaged results are shown in table 3.3. With this small problem size, Gurobi outperforms Benders by nearly three orders of magnitude on average.

These solve times for the manufacturing plant model indicate that Benders decomposition is not sufficient on its own to decrease the solution time as will be necessary for solving real-world scale problems, and Benders is in fact detrimental to solution times for this problem. There are a number of reasons that this is likely the case. First, the master problem in this case is far larger of a problem than the subproblem, and there are many more binary- and integer-valued decision variables than continuous variables in this problem. Because of this, the number of possible feasibility cuts to be added to the master problem is large. There are \( i \cdot j \) total integer-valued
variables and $i \cdot j \cdot l + i \cdot j \cdot k + 2 \cdot i \cdot j \cdot k^2 + 2 \cdot k + i \cdot j \cdot k \cdot l$ binary-valued decision variables. With a maximum of five possible integer values for the integer variables, as with the problem formulation detailed in chapter 2, this leads to $5^i \cdot j \cdot l + i \cdot j \cdot k + 2 \cdot i \cdot j \cdot k^2 + 2 \cdot k + i \cdot j \cdot k \cdot l$ possible combinations of master problem decision variables and the same number of possible feasibility cuts. If optimality cuts are not found and added in early iterations of the algorithm’s execution, slow convergence can result. Indeed, as the manufacturing plant problem evolves, many more feasibility cuts are added than optimality cuts, and the problem converges slowly. As illustrated by the results in table 3.3, the average number of feasibility cuts is much higher than the average number of optimality cuts for the small problems that converged in under one hour.

<table>
<thead>
<tr>
<th>Number of Products</th>
<th>Number of Workcenters</th>
<th>Time Horizon</th>
<th>Ave. Feasibility Cuts</th>
<th>Ave. Optimality Cuts</th>
<th>Ave. Benders Solve Time [s]</th>
<th>Ave. MILP Solve Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>104.84</td>
<td>11.00</td>
<td>11.78</td>
<td>0.0158</td>
</tr>
</tbody>
</table>

Table 3.1: Feasibility Cuts, Optimality Cuts, and Solve Times for Two Days, Three Products

Another potential reason for slow convergence is that in classical Benders decomposition, every time the subproblem is executed and a feasibility or optimality cut is added, the master problem needs to be rebuilt with the new constraints added. For problems that require many iterations for convergence as is the case here, each iteration adds an extra constraint to the master problem, and the master problem takes longer to build and solve with each added iteration.

While Benders decomposition has been shown to improve solve times for smaller problems with fewer discrete decision variables, for the supply chain problem being considered in this work, it cannot provide the necessary improvements in solve times. In order to improve solve times beyond what can be accomplished with exact techniques, approximate techniques are explored in chapter 4.
3.4 Future Work

In order to address the problem of the need to rebuild the master problem with every iteration, Benders decomposition can be implemented using callback capabilities of modern day optimizers [1]. Using this strategy, the master problem is built once, and every time an integer-feasible solution is found at a branch-and-bound node, the subproblem is solved, and an optimality or feasibility cut is added as a lazy constraint. A lazy constraint is a constraint that remains inactive until a feasible solution is found. Once a feasible solution is found, the solution is checked against all lazy constraints, and if the solution violates any lazy constraint, it is discarded and one or more of the violated lazy constraints are added to model as regular constraints [2]. In addition to saving time by requiring the master problem to be formulated only once, implementing Benders with callbacks would also help combat the problem of slow convergence due to a large number of feasibility cuts versus optimality cuts since a cut is added with each integer-feasible solution found as the branch-and-bound algorithm executes rather than with each new optimal integer solution found each time a new master problem iteration is run.

Also left as part of future work is the implementation of an integrated supply chain model with Benders decomposition and further exploration of the Dantzig-Wolfe decomposition technique. Since the manufacturing plant problem has a large number of complicating constraints in addition to a large number of complicating variables, we might not expect Dantzig-Wolfe decomposition to show much improvement over the Benders implementation. However, similar variants of Dantzig-Wolfe decomposition to the one discussed for Benders decomposition above might also show improved solve times from the original MILP formulation.
Chapter 4

Approximate Techniques

While many large-scale problems, including the supply chain problem, lend themselves well to the MILP framework for modeling of their objectives, decision variables, and constraints, exact techniques used to solve MILP problems are often insufficient to solve these large models in time scales that are required by their given domains. In order to address this, approximate techniques can be applied to solve large problems more quickly than exact techniques can. The following sections detail approximate techniques that have been applied in previous work, the hierarchical decomposition technique that has been applied in this work, and the solve time improvements achieved by this technique. The hierarchical technique developed in this work is applied to the manufacturing plant problem modeled in chapter 2. It decomposes the problem into three levels: an approximate work allocation stage at the top level, an approximate labor allocation stage at the middle level, and the scheduling problem formulated as a MILP at the lowest level.

4.1 Related Work

Approximate techniques have been applied to supply chain problems and other problems of similar structure, integrating both the MILP framework and others. For example, within the supply chain context, Santoso et al. [24] and Tsao et al. [29] implement multi-stage approximate techniques for a supply chain network problem...
and an integrated facility location-inventory allocation problem, respectively. In [24], a two-stage stochastic model is created for deciding on which manufacturing facilities to open in the first stage, constituting the strategic network design decisions, and then optimizing processing and transportation of products in the second stage, constituting the tactical decisions. The first stage uses the Sample Average Approximation (SAA) scheme to approximate the tactical contribution to the objective function in order to inform facility opening decisions at the strategic level. The second stage uses Benders decomposition to solve for the true optimal tactical decisions based on the strategic decisions from the first stage. Since the strategic and tactical decisions are highly coupled, the problem can be also formulated as a monolithic optimization problem. Santoso et al. [24] compare results for two models applying the monolithic formulation, applying classical Benders decomposition to the monolithic problem, and applying the two-stage problem with an accelerated Benders decomposition strategy. In both cases, the two-stage problem significantly outperforms the other two strategies in terms of solve times and sizes of problems solved. While the high level nature of the problem being solved in [24] is different from the lower-level problem considered in this thesis, the two-stage decomposition strategy provides an interesting framework from which to think about solving large-scale optimization problems.

In [29], a two-phase technique involving a continuous approximation of discrete decisions is applied to an integrated facility location and inventory allocation problem. In the problem modeled in their work, decisions need to be made about where to locate distribution facilities and about inventories at the facilities over time. In the first phase of the problem, the larger facility location problem is broken down into regions that are small enough that it is possible to use a continuous approximation for all decisions made in the second phase. Then the second phase continuous optimization problem is solved. Results are compared for the disintegrated facility location and inventory problem and the integrated problem applying the two-phase approximation approach, and the integrated strategy shows objective function improvements over the disintegrated strategy. In this thesis, the manufacturing plant problem cannot be broken down in a way that allows for continuous approximations of all decisions (for
example, labor allocation is a purely discrete decision). However, decomposing the problem such that discrete decisions and continuous decisions are made in different stages is an interesting and potentially beneficial strategy.

Additional approximate dynamic programming approaches for supply chain problems are applied in the literature in order to reduce the size of the problems that need to be solved by approximating the objective function contributions for pieces of the problem abstracted away. For example, Toriello [28] considers an inventory routing model with multiple suppliers and consumers. A MILP formulation of the problem is developed and an approximate dynamic programming (ADP) approach is used to approximate long term cost (part of the objective function) based on short term decisions. This allows for optimization over shorter time periods with consideration of long-term impacts. Their results suggest that the ADP gives high-quality solutions much faster than optimizing over a longer time horizon. In the motivating example for this thesis, schedules are developed one to two weeks in advance of execution, and the bottleneck lies not in the length of the time horizon, but in the large number of products that need to be scheduled. Nonetheless, dynamic programming approaches could be an interesting approach for abstracting other parts of the problem or for considering longer time horizons in the future.

Beyond the supply chain context, heuristic scheduling approaches have been applied to planning and scheduling problems in disciplines such as robotics, hospital patient treatment planning, and mine operations planning. In [15], Gombolay, Wilcox, and Shah decompose a highly-coupled task assignment and scheduling problem into the task allocation and sequencing parts, and heuristics are used at the task allocation level to guide towards better outputs by accounting for spatial and resource concerns. Solve time improvements are achieved over a benchmark MILP formulation. In [8], Castro and Petrovic consider a multi-objective scheduling problem for patient procedures within a hospital. In their approach, a heuristic is used to generate an initial feasible schedule, and then the seed schedule is improved upon in three levels that each solve for one of the three objective components using MILP formulations. Although solution time is reduced with this technique as compared to solving with
a single, unified MILP, global optimality is lost. In [19], Lipovetzky et al. solve a mine operations planning problem using a hybrid approximation technique. In this technique, resource constraints are modeled using the MILP framework, and the sequencing problem is formulated as a planning problem. A heuristic objective function term is used in the MILP to guide towards outputs that lead to better sequencing solutions in the planning problem. Improvements in solution quality are achieved using the heuristic technique over formulating the resource problem as a feasibility problem. Though these heuristic approaches tend to find near-optimal solutions for small problems, they do not tend to scale well to larger problems. Since we consider a large-scale supply chain problem in this work, alternate techniques are explored.

One interesting approach, the multi-abstraction search approach (MASA), implements a hierarchical abstraction technique that is shown to yield a 15% and 25% reduction in makespan compared with a MILP-based approximate approach and a conventional hill climbing algorithm, respectively (Zhang and Shah [31]). In that approach a workcell task assignment and scheduling problem is decomposed into three levels: abstract agent allocation to workcells at the top level, a less abstract hill climbing approach that considers task assignment at the middle level, and fine tuning of the agent and task assignments incorporating scheduling details at the bottom level. The MASA approach is applied to a workcell task assignment and scheduling problem, which differs from the manufacturing plant problem considered in this work in a few respects. First, in the MASA problem, there are a fixed number of agents to be allocated to the workcells. In the manufacturing plant problem in this work, labor is theoretically unlimited, but it is practically capped by the maximum labor value for each workcenter each day. Second, in the manufacturing plant problem, each workcenter performs only one type of task and specific agents are not assigned to specific tasks as with the model in the MASA approach. Finally, in the model used with the MASA approach, there are no specified deadlines as there are in the manufacturing plant problem, and instead makespan is minimized as the overall objective. Despite these differences, the parallels in structure between that problem and the manufacturing plant problem and the decomposability of both problems make MASA an
interesting approach to explore within the larger supply chain context. The work in this thesis implements and analyzes an approach inspired by the hierarchical elements of the MASA technique along with elements of heuristic approaches.

4.2 Hierarchical Approximate Technique

As in chapter 3, the approximate algorithm explored in this chapter is implemented for the manufacturing plant problem, and expansion to the full integrated supply chain problem is left as future work. The approximate algorithm implemented in this work is a three-level hierarchical technique. At the top level, products are approximately assigned to workcenters on given days in the time horizon. At the middle level, labor is approximately assigned to workcenters at the manufacturing plant for each day in the time horizon given the work allocation decided upon at the top level. Finally, at the lowest level, a scheduling problem formulated as a MILP is solved for each day in the time horizon given the work allocation from the top level and the labor allocation from the middle level. Figure 4-1 shows the algorithm hierarchy.

4.2.1 Overview

The high level pseudocode for the hierarchical algorithm is in algorithm 2. The algorithm takes as input the same parameters as those inputted into the manufacturing plant model in chapter 2. These include the number of products to be produced, the required workcenters for each product, workcenters_{kj}, the deadlines for each product, deadlines_{k}, the minimum and maximum labor for each workcenter, minLabor_{j} and maxLabor_{j}, the cycle times corresponding to each labor amount for each workcenter, cycleTimes_{jl}, the number of hours in a shift, hoursPerShift, and the length of the time horizon, timeHorizon (line 1). The outputs of the algorithm include the labor allocation to the workcenters, \( \bar{A}_{ij} \), and the scheduled start times of each product at each workcenter, \( \bar{P}_{jk} \) as detailed in chapter 2 (line 2).

First, the current best solution is initialized to infinity, a boolean indicating whether or not a solution has been found in initialized to false, the number of it-
erations is initialized to zero, and the maximum number of iterations is initialized to the chosen value (lines 3-6). The main loop of the algorithm begins in line 7 and runs until either the first solution has been found or the maximum number of iterations has been exceeded, whichever happens later. In the first iteration, the algorithm begins by initializing work at all workcenters for each product to the product’s deadline day in order to seed the algorithm near a solution with minimal lateness (lines 8-9). In all subsequent iterations of the algorithm, work is allocated according to the process detailed in section 4.2.2 (lines 10-11). In line 12, a boolean describing whether there are feasible labor allocations and schedules for all days in the time horizon given the top level work assignment is initialized to true. Once the top level assignment of work for products to days in the time horizon is achieved at the top level, the work assignment is sent to the middle level for labor allocation for each of the days in the time horizon (line 13). The bottom two stages constitute a labor allocation and scheduling problem for each of the individual days in the time horizon. At the middle level, labor is assigned to workcenters independently for each day in the time horizon according to the process described in section 4.2.3 (line 14). With each labor allocation attempted for each day, the lowest level scheduling problem is run as detailed in section 4.2.4 (line 15). If there are no feasible labor allocations given the top level work assignment for any one of the days in the time horizon, the solution boolean is set to false and the for loop is exited (lines 16-18). In this case in which the top level assignment is infeasible, the algorithm returns to the top level to try a new work allocation. If the scheduling level produces a feasible schedule for each day in the time horizon given the labor allocation from the middle level, indicating work allocation feasibility as well, the algorithm calculates the current objective value (including labor and lateness as in chapters 2 and 3), stores it as the current objective, and sets the noSolution boolean to false (lines 19-21). The current objective is compared to the current best objective, and the current objective is stored as the current best solution along with the decision variable assignments if it is better than all previous solutions (lines 22-23). Finally, the number of iterations is incremented (line 24). The algorithm then returns to the top level to begin a new iteration.
Since the algorithm performs work allocation approximately and independently of information about labor allocation or scheduling, the true optimal work allocation to days may not be achieved as it is when solving the original MILP formulation. Similarly, at the middle level, labor allocation takes place without incorporating detailed scheduling information. By decoupling these components of the problem, the ability to guarantee that the true optimal labor allocation is reached is lost. Additionally, the approximate technique is incomplete and will indefinitely continue to search for a solution if the problem is infeasible. In spite of these limitations, solving the problem using this approximate decomposition technique may lead to near-optimal solutions in less time than it takes to optimize the full manufacturing plant MILP.

\begin{algorithm}
\caption{Approximate Algorithm Pseudocode}
\begin{algorithmic}[1]
\STATE \textbf{Inputs:} workcenters\textsubscript{kj}, deadlines\textsubscript{k}, minLabor\textsubscript{j}, maxLabor\textsubscript{j}, cycleTimes\textsubscript{jl}, hoursPerShift, timeHorizon
\STATE \textbf{Outputs:} $\bar{A}_{ij}$, $\bar{P}_{jk}$
\STATE Initialize currentBestObjective = $\infty$
\STATE noSolution $\leftarrow$ true
\STATE iteration $\leftarrow$ 0
\STATE Set maxIteration
\WHILE{noSolution = true and iteration < maxIterations}
\IF{iteration = 0}
\STATE Initialize all workcenters for all products to the product’s deadline day
\ELSE
\STATE Run top level: work allocation to days
\STATE solution $\leftarrow$ true
\ENDIF
\FOR{All days $i$ in the time horizon}
\STATE Run middle level: labor allocation for day $i$
\IF{No feasible labor allocation for day $i$}
\STATE solution $\leftarrow$ false
\STATE break
\ENDIF
\IF{solution = true}
\STATE noSolution $\leftarrow$ false
\STATE Set currentObjective given current solution
\IF{currentObjective < currentBestObjective}
\STATE currentBestObjective $\leftarrow$ currentObjective
\ENDIF
\ENDIF
\ENDFOR
\STATE iteration $\leftarrow$ iteration + 1
\ENDWHILE
\end{algorithmic}
\end{algorithm}
4.2.2 Top Level: Work Allocation to Days

At the top of level of the algorithm, the work required at each workcenter for each product is approximately assigned to a day in the time horizon. All workcenters for a given product are initialized to the product’s deadline day, and as work is then shifted between days in the time horizon, precedence constraints between workcenters are maintained. In order to decide how to move work between days, two heuristics are calculated with each iteration of the algorithm: a normalized workcenter workload heuristic and a normalized product workload heuristic. The workcenter workload heuristic is expressed in equation 4.1 and the product workload heuristic is expressed in equation 4.2.

$$WCL_{ij} = \frac{cycle\, Times_{j,0}}{hours\, Per\, Shift} \cdot \sum_{k \in K} \tilde{F}_{ijk} \quad \forall \, i \in I, j \in J$$ (4.1)
\[ PWL_{ik} = \frac{1}{\text{hoursPerShift}} \cdot \sum_{j \in J} \text{workcenters}_{jk} \cdot \text{cycleTimes}_{j,0} \cdot \tilde{F}_{ijk} \]

\[ \forall i \in I, k \in K \quad (4.2) \]

The workcenter workload heuristic, \( WCWL_{ij} \) is calculated for every day in the time horizon \( i \) and every workcenter \( j \). For each workcenter and each day, it is the number of products assigned to that workcenter times the maximum cycle time for that day, divided by the total hours in a shift in order to normalize. The product workload heuristic, \( PWL_{ik} \) is calculated for every day in the time horizon and every product. It is the sum of the maximum cycle times for each required workcenter for each product and each day divided by the total hours in a shift to normalize.

Pseudocode for the top level piece of the approximate algorithm is shown in algorithm 3. The inputs of the work allocation level include the required workcenters for each product, \( \text{workcenters}_{kj} \), the deadlines for each product, \( \text{deadlines}_k \), the cycle times corresponding to each labor amount for each workcenter, \( \text{cycleTimes}_{jd} \), the number of hours in a shift, \( \text{hoursPerShift} \), the length of the time horizon, \( \text{timeHorizon} \), and the incumbent assignment of work to days for each product, \( \tilde{F}_{ijk} \) (line 1). The output of the work allocation level is the final allocation of workcenters for each product to days, \( \bar{F}_{ijk} \) (line 2). With each high level iteration of the approximate algorithm, both sets of heuristics are calculated (lines 3 and 4) and the \( \text{noValidReassignment} \) boolean which describes whether a new valid work allocation has been found is initialized to true (line 5). Then the main loop of the top level work allocation algorithm begins and runs until a valid reallocation of work has been found (line 6). In lines 7 and 8, the largest of each of the heuristics is determined. Between those two maximum values, if the workcenter workload heuristic is larger, the following procedure takes place (lines 11-12): if the associated day for the workcenter heuristic is the first day in the time horizon, work at the associated workcenter for a randomly selected product and all of the subsequent workcenters for that product.
that are also assigned to the first day are moved back a day to the second day in the time horizon. If the associated day for the workcenter heuristic is not the first or last day in the time horizon and one of the products processed at the workcenter on that day also has a workcenter assigned to the first day in the time horizon, then the associated workcenter for the heuristic and all subsequent workcenters required for that product on that day are moved back a day in the time horizon. Otherwise, work at the associated workcenter for a randomly selected product and all of the preceding assigned workcenters for that product on that day are moved up a day in the time horizon.

If the product workload heuristic is larger, the following procedure takes place (lines 9-10): if the associated day for the product heuristic is the first day in the time horizon, the last required workcenter for that product on that day is moved back a day to the second day in the time horizon. Otherwise, the first required workcenter for that product on that day is moved up a day in the time horizon.

After one of the above procedures has taken place, noValidReassignment is set to false (line 13). Then the new work allocation assignment is checked against all old assignments to make sure the new assignment has not been attempted before (line 14). If it has been attempted before, noValidReassignment is set to true, PWLmax or WCWlmax is reset to zero depending on which one was higher, the incumbent work allocation is reset to the previous allocation, and the reassignment procedure is run again for the next highest of the calculated heuristic values (lines 15-20).

### 4.2.3 Middle Level: Labor Allocation to Workcenters

At the second level of the approximate algorithm, labor is approximately assigned to workcenters for each day in the time horizon while taking into consideration the work allocation from the top level. The second level assigns labor for each day in the time horizon independently. For a given day, labor is initialized to the minimum possible labor for each workcenter at the plant. With each iteration of the middle level, a workcenter workload heuristic is calculated for each workcenter at the plant, and this heuristic is used to determine how to increment labor for each day. The heuristic is
Algorithm 3 Work Allocation Pseudocode

1: Inputs: workcenters\(_{kj}\), deadlines\(_k\), cycleTimes\(_{jl}\), hoursPerShift, timeHorizon, \(\bar{F}_{ijk}\)
2: Output: \(\bar{F}_{ijk}\)

3: Calculate PWL\(_{ik}\) ← \(\frac{1}{\text{hoursPerShift}} \cdot \sum_{j \in J} \text{workcenters}_{jk} \cdot \text{cycleTimes}_{j,0} \cdot \bar{F}_{ijk}\)

4: Calculate WCWL\(_{ij}\) ← \(\frac{\text{cycleTimes}_{j,0}}{\text{hoursPerShift}} \cdot \sum_{k \in K} \bar{F}_{ijk}\)

5: Initialize noValidReassignment ← true
6: while noValidReassignment = true do
7: \(PWLmax \leftarrow \max_{i,k} PWL_{ik}\)
8: \(WCWLmax \leftarrow \max_{i,j} WCWL_{ij}\)
9: if \(PWLmax > WCWLmax\) then
10:   Move workcenter for the product to different day
11: else
12:   Move product from workcenter
13: noValidReassignment ← false
14: if New top level assignment is a repeat assignment then
15:   noValidReassignment ← true
16: if \(PWLmax > WCWLmax\) then
17:   \(PWLmax \leftarrow 0\)
18: else
19:   \(WCWLmax \leftarrow 0\)
20: Reset top level assignment to previous top level assignment
calculated as:

\[ H_j = \sum_{k \in K} \bar{F}_{\text{day},j,k} \cdot \text{cycleTimes}_{j,\text{currentAssignment}} \quad \forall j \in J \quad (4.3) \]

For each workcenter, the heuristic is the cycle time for the workcenter with the current labor assignment multiplied by the number of products assigned to it for that day. The pseudocode for the middle level labor allocation problem is shown in algorithm 4. The inputs of the middle level algorithm include the minimum and maximum labor for each workcenter, \( \text{minLabor}_j \) and \( \text{maxLabor}_j \), the cycle times corresponding to each labor amount for each workcenter, \( \text{cycleTimes}_{jl} \), and the work allocation from the top level, \( \bar{F}_{ijk} \) (line 1). The output of the middle level is the labor allocation for each day and each workcenter, \( \bar{A}_{ij} \) (line 2). Line 3 initializes all labor to the minimum possible value corresponding to each workcenter. The heuristic values are calculated for each workcenter in line 4, and the index for the workcenter with this highest heuristic value is set in line 5. If the workcenter with the highest heuristic value is already at its maximum labor (line 6), the heuristic value for that workcenter is set to zero (line 7), the index of the workcenter with the new maximum heuristic value is set (line 8), and labor is incremented at the workcenter with the next highest heuristic value (lines 12-13). Otherwise labor is incremented by one at the workcenter associated with the highest heuristic value (lines 14-15). If all workcenters are assigned their maximum values for labor, the scheduling level is run to check for feasibility (lines 9-10). If there is no feasible schedule for any one of the days in the time horizon, the current work allocation to days at the top level is not feasible, so the algorithm returns to the top level for a new work allocation (line 11).

4.2.4 Bottom Level: Scheduling

Once the algorithm has reached the lowest level, work has been assigned to days in the time horizon and labor has been assigned to workcenters for each day in the time horizon. The work allocation to days and labor allocation to workcenters are the inputs to the lowest level scheduling problem. The output of the lowest level is
Algorithm 4 Labor Allocation Pseudocode

1: Inputs: minLabor\_j, maxLabor\_j, cycleTimes\_jt, \( F_{ijk} \)
2: Output: \( A_{ij} \)
3: Initialize \( A_{day,j} \leftarrow \text{minLabor}_j \quad \forall \ j \in J \)
4: Calculate \( H_j = \sum_{k \in K} F_{day,j,k} \cdot \text{cycleTimes}_j_{currentAssignment} \quad \forall \ j \in J \)
5: \( j \leftarrow \text{index}(\max_j H_j) \)
6: if \( A_{day,j} = \text{maxLabor}_j \) then
7: \( H_j \leftarrow 0 \)
8: \( j \leftarrow \text{index}(\max_j H_j) \)
9: if \( H_j = 0 \) then
10: Run scheduling level and save result
11: Return to top level
12: else
13: Increment \( A_{day,j} \) by 1
14: else
15: Increment \( A_{day,j} \) by 1

The schedule of all products through all workcenters for a given day. The scheduling problem for each day in the time horizon is solved independently, taking into consideration the labor assignment from the middle level and the work assignment from the top level, and is formulated as a MILP, which is a reduced version of the original MILP described in chapter 2. The MILP for each day can be written as follows:

\[
\min \sum_{k \in K} Q_k \tag{4.4}
\]

\[
S_{jk} + \sum_{l \in L} \text{cycleTimes}_{jl} \cdot \text{workcenters}_{kj} \cdot F_{day,j,k} \cdot Y_{day,j,l} \leq S_{j+1,k} \quad \forall j \in J, k \in K \tag{4.5}
\]

\[
S_{jk} + \sum_{l \in L} \text{cycleTimes}_{jl} \cdot \text{workcenters}_{kj} \cdot F_{day,j,k} \cdot Y_{day,j,l} \geq S_{j+1,k} - M_1 \cdot \text{workcenters}_{k,j+1} \quad \forall j \in J, k \in K \tag{4.6}
\]
\[ S_{jk} + L_{jkm} \geq S_{jm} \quad \forall j \in J, k, m \in K \quad (4.7) \]

\[ S_{jk} + L_{jkm} \leq S_{jm} + 2 \cdot \text{hoursPerShift} \cdot D_{2jkm} \quad \forall j \in J, k, m \in K \quad (4.8) \]

\[ S_{jm} + L_{jkm} \geq S_{jk} \quad \forall j \in J, k, m \in K \quad (4.9) \]

\[ S_{jm} + L_{jkm} \leq S_{jk} + 2 \cdot \text{hoursPerShift} \cdot D_{1jkm} \quad \forall j \in J, k, m \in K \quad (4.10) \]

\[ D_{1jkm} + D_{2jkm} = 1.0 \quad \forall j \in J, k, m \in K \quad (4.11) \]

\[ M_2 \cdot \text{workcenters}_{kj} \cdot F_{\text{day},j,k} + M_2 \cdot \text{workcenters}_{mj} \cdot F_{\text{day},j,m} + \]
\[ \sum_{l \in L} \text{cycleTimes}_{jl} \cdot \text{workcenters}_{kj} \cdot \text{workcenters}_{mj} \cdot \bar{Y}_{\text{day},j,l} \]
\[ \leq L_{jkm} + 2 \cdot M_2 \quad \forall j \in J, k, m \in K \quad (4.12) \]

\[ \sum_{l \in L} \text{cycleTimes}_{jl} \cdot \text{workcenters}_{kj} \cdot \bar{Y}_{\text{day},j,l} \cdot \bar{F}_{\text{day},j,k} + \]
\[ \text{workcenters}_{kj} \cdot F_{\text{day},j,k} \cdot S_{jk} \leq \text{hoursPerShift} \quad \forall j \in J, k \in K \quad (4.13) \]

\[ \text{hoursPerShift} \cdot \text{day} \cdot \bar{F}_{\text{day},\text{last},k} + \bar{F}_{\text{day},\text{last},k} \cdot S_{\text{last},k} = P_{\text{last},k} \quad \forall j \in J, k \in K \quad (4.14) \]

\[ W_k = \sum_{l \in L} \text{cycleTimes}_{\text{last},l} \cdot \text{workcenters}_{k,\text{last}} \cdot \bar{F}_{\text{day},\text{last},k} \cdot \bar{Y}_{\text{day},\text{last},l} + P_{\text{last},k} \]
\[ \forall k \in K \quad (4.15) \]

\[ Q_k + W_k \leq \text{deadlines}_k + 2 \cdot \text{hoursPerShift} \cdot \text{timeHorizon} \cdot T_{1k} \quad \forall k \in K \quad (4.16) \]

\[ Q_k + W_k \geq \text{deadlines}_k \quad \forall k \in K \quad (4.17) \]

\[ Q_k + \text{deadlines}_k \leq W_k + 2 \cdot \text{hoursPerShift} \cdot \text{timeHorizon} \cdot T_{2k} \quad \forall k \in K \quad (4.18) \]

\[ Q_k + \text{deadlines}_k \geq W_k \quad (4.19) \]

\[ T_{1k} + T_{2k} = 1.0 \quad \forall k \in K \quad (4.20) \]
This MILP above is a subset of the overall manufacturing plant MILP described in chapter 2 and describes only the scheduling piece of the problem for an individual day in the time horizon. $\bar{F}_{ijk}$ and $\bar{Y}_{ijl}$ (which can be derived from $\bar{A}_{ij}$) have been set in the top and middle levels, respectively, and are used here to enforce constraints on scheduling for each day appropriately. As with the manufacturing plant model in chapter 2, $j \in J$ is a workcenter in the set of all workcenters $J$, $k, m \in K$ are products in the set of all products $K$, and $l \in L$ is labor assignment (number of people assigned) in the set of all possible labor assignments $L$. $\bar{F}_{day,j,k} \in \{0,1\}$ is a binary decision variable that takes the value of one if product $k$ is assigned to workcenter $j$ on day $i$ and zero otherwise. It is set for each day at the top level of the algorithm. $\bar{Y}_{day,j,l} \in \{0, \text{hoursPerShift}\}$ is a decision variable used to represent labor at the workcenters and can be derived from the $\bar{A}_{day,j}$ values set at the middle level of the algorithm and constraint 2.4. $S_{jk} \in [0, \text{hoursPerShift}]$ is the time product $k$ is scheduled to begin work at workcenter $j$ on the day being considered. $P_{jk} \in \mathbb{R}$ the absolute time in the total time horizon that product $k$ is scheduled to begin production at work center $j$, $W_k \in \mathbb{R}$ is the absolute time in the total time horizon that product $k$ is scheduled to complete work at its last workcenter, and $Q_k \in \mathbb{R}$ is the total lateness, or the absolute value of the difference between the deadline time and the absolute end time of product $k$. $D_{1jkm} \in \{0,1\}$, $D_{2jkm} \in \{0,1\}$, $L_{jkm} \in [0, \text{hoursPerShift}]$, $T_{1k} \in \{0,1\}$, and $T_{2k} \in \{0,1\}$ are decision variables used for linearizing constraints. The array $cyctime_{s,t}$ represents the cycle time that corresponds to a labor allocation of $l$ people to workcenter $j$, $d_ealines_k$ represents the deadlines for each of the products $k$, and $workcenters_{k,j}$ is a binary-valued input representing whether product $k$ requires workcenter $j$ for production. $M_1$ and $M_2$ are large positive integers with $M_1 < M_2$. We set $cyctime_{s,t,0} = M_1$ for all $j$, and cycle times corresponding to disallowed labor allocation values for a given workcenter are set to $M_2$. Finally, $\text{hoursPerShift}$ is the total number of hours in a shift and $\text{timeHorizon}$ is the total number of days in the time horizon.

Equation 4.4 is the problem objective, minimizing the total lateness. Equations 4.5 and 4.6 ensure that a product is not scheduled at the next workcenter until it is
completed at the last workcenter. Equations 4.7-4.11 are linearizing equations setting $L_{jkm} = |S_{jk} - S_{jm}|$. Equation 4.12 enforces that each workcenter only works on one product at a time. Equation 4.13 ensures that each workcenter completely finishes a unit of work before the end of the day. Finally, equations 4.15-4.20 set the lateness for each product to the absolute value of the difference between the deadline and the absolute end time for the product.

The MILP is solved using the Gurobi optimizer, and results are used to assess the overall problem objective function value with each iteration of the algorithm. If the MILP is infeasible for any day in the time horizon, the labor allocation from the middle level must be infeasible, and the algorithm returns to the middle level for a new labor allocation.

### 4.2.5 Incompleteness of Approximate Technique

The approximate algorithm as detailed in the previous sections will find a solution if one exists, but it will not be able to determine that a feasible solution does not exist if the problem is infeasible given the problem inputs. The main loop of the algorithm, as shown in algorithm 2, runs either until the maximum number of iterations has been exceeded or until a solution is found, whichever comes later. Therefore, if the problem is infeasible and no solution is possible, the algorithm will continue to search for solutions indefinitely without terminating, and it is incomplete.

### 4.3 Results and Discussion

In order to test the approximate algorithm and compare its performance to the original MILP solver, the same set of randomly generated manufacturing plant problems that was tested with the original MILP as detailed in chapter 2, was tested with the approximate algorithm. Problems run included the same 20 sets of randomly generated parameters as were evaluated with the original manufacturing plant MILP. As with the manufacturing plant MILP, the approximate algorithm was tested holding the number of workcenters at the plant, the maximum amount of labor possible at
any one workcenter, and the problem time horizon constant at three workcenters, five
people per workcenter, and 10 days, respectively. While the number of workcenters,
maximum labor, and time horizon parameters were held constant, the number of
products was varied between five and 100 since 100 products was the largest prob-
lem size for which a majority of problems tested finished within a one hour timeout.
Problem sizes of five, 10, 15, 20, 25, 30, 40, 50, 60, 90, and 100 products were tested.
Note that this includes two additional cases (90 and 100 products) on top of what
was evaluated with the original MILP since larger problems could be solved with the
approximate technique. For the 90 and 100 product size cases, 20 additional sets of
problem parameters were generated. For all cases, the 20 sets of problem parameters
were randomly generated as follows: an integer value representing the minimum labor
for a workcenter was sampled from a uniform distribution between zero and five. An
integer value representing the maximum labor for a workcenter was sampled from
a uniform distribution between that workcenter’s minimum labor and five to ensure
that is was larger than the minimum value. Cycle times (in hours) corresponding to
each possible labor amount for each workcenter were sampled from a uniform distri-
bution between zero and one in order to represent real-world cycle times that are on
the order of minutes. Binary values representing which workcenters were required for
each product (zero if the workcenter was not required for the product and one if it
was) were randomly, uniformly selected. Finally, deadline days for each product were
sampled from a uniform distribution between zero and the total number of days in
the time horizon (10 days), and deadline times for each product on its deadline day
were sampled from a uniform distribution between zero and the number of hours per
work day (eight hours).

The approximate algorithm was run with a maximum of 10 top level iterations or
until the first solution was found, whichever required more iterations. The algorithm
was implemented with Java version 1.8.0, and the lowest level scheduling problem was
implemented with Gurobi version 7.5.1 for Java, as with the other MILP formulations
in this work [16]. All tests were run on a computer with a 2.9 GHz Intel core i5
processor.
Table 4.1 shows the number of problems for each problem size that finished within the time out limit of one hour. For the original MILP, all problems through a size of 40 products were solved within the limit, half of the problems with a size of 50 products were solved in under an hour, and no larger problems were solved within the time limit. With the approximate technique, all problems through a size of 60 products were solved within the time limit, and both the 90 and 100 product cases had four timeouts apiece. None of the problems tested with the approximate technique took more than 10 top level iterations to reach a solution. For the approximate technique, the low level MILP was the bottleneck as problem size increased. All timeouts for the approximate technique were cases in which over 14 products were randomly generated to have deadlines on the same day in the time horizon. Since the approximate algorithm initializes all workcenters for all products to the product’s deadline day in its first iteration, if a large number of products have deadlines on the same day, the low level MILP becomes prohibitively large. Since 14 products seemed to be a limit, the largest problem we might expect the current approximate technique to be able to solve in under one hour is 140 products. Addressing this limit is left as future work.

<table>
<thead>
<tr>
<th>Number of Products</th>
<th>MILP Number Solved</th>
<th>MILP Number Not Solved</th>
<th>Approx. Number Solved</th>
<th>Approx. Number Not Solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>20</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>20</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>20</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>10</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>60</td>
<td>-</td>
<td>-</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>90</td>
<td>-</td>
<td>-</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>100</td>
<td>-</td>
<td>-</td>
<td>16</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 4.1: Number of Problems Solved in under One Hour for Varying Problem Sizes
Figure 4-2 shows the median solve times for each problem size for the original MILP as were reported in chapter 2, compared with both the median times to reach the first solutions found for each problem size using Gurobi for Java to solve the original MILP model and the median times to run 10 iterations of the approximate algorithm for each problem size. The median data points take into account the problems only that were solved within the one hour time limit, and 25%-75% quartile ranges for each data point are shown. Note that all times to reach an initial solution to the MILP for the five and 10 product cases were small enough that the 25% and 75% quartile values were approximately zero with the accuracy with which solve times were recorded. Thus, these data points do not appear on the log plot. As the data suggest, the median time taken for the approximate technique to complete 10 iterations is smaller than the median time it takes for the original MILP to reach its first solution and is smaller than the median time it takes for the original MILP to reach its optimal solution for all problem sizes. Additionally, the approximate technique solves larger problems than the original MILP can solve with the one hour time out.

While the solve times for the approximate technique as compared to the original MILP formulation are promising, it is important also to consider the solution quality as it compares to the true optimal solution. Table 4.2 shows the average percent difference between the true optimal solution and the approximate technique’s solution for each problem size. Only problems that did not time out for either the original MILP or the approximate technique are considered in this calculation, since the true optimal solution is only available for comparison in those cases. As problem size increases, the approximate technique’s performance in terms of solution quality predictably declines. The worst case performance of all problems completed with both the approximate technique and the MILP formulation was a 20.35% difference in solution for a problem size instance of 40 products. Of note is that the solution quality seems to improve between the 40 and 50 product problem sizes. A possible reason for this is that only half of the 50 product problems were solved with the original MILP formulation, while all were solved for the 40 product case. Since the
Figure 4-2: Log Median Solve Times for Original MILP and Approximate Algorithm

problems that were not solved with the original MILP formulation were likely harder problems than those that were solved, it is possible that the solutions that the approximate algorithm reaches for the unsolved problems within 10 iterations would also be of lower quality than the ones that were solved by both techniques within the time limit. Further tests would be necessary to confirm this.

It is also of interest to compare the approximate algorithm’s solution to the first solution found by the original MILP solver. Table 4.2 also shows the average percent differences between the original MILP’s first solution and the approximate algorithm’s solution after 10 iterations. On average, the approximate technique performs better in 10 iterations and in less time than the original MILP’s first solutions. In 19.28% of all problem instances for which there were both an approximate solution and at least one MILP solution reached within one hour, the original MILP’s first solution was better than the approximate algorithm’s solution. In the worst case the initial MILP solution was 12.72% better than the approximate solution for a problem size instance of 50 products. Table 4.3 shows the total percentage of initial MILP solutions that
were better than the approximate solutions found for the same problems.

<table>
<thead>
<tr>
<th>Number of Products</th>
<th>Average % Difference between Optimal Solution and Approximate Solution</th>
<th>Average % Difference between First MILP Solution and Approximate Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.1727%</td>
<td>-24.7811%</td>
</tr>
<tr>
<td>10</td>
<td>0.1207%</td>
<td>-34.1768%</td>
</tr>
<tr>
<td>15</td>
<td>0.9537%</td>
<td>-27.9169%</td>
</tr>
<tr>
<td>20</td>
<td>1.4452%</td>
<td>-15.3078%</td>
</tr>
<tr>
<td>25</td>
<td>2.3389%</td>
<td>-25.3231%</td>
</tr>
<tr>
<td>30</td>
<td>3.4827%</td>
<td>-23.6805%</td>
</tr>
<tr>
<td>40</td>
<td>5.8274%</td>
<td>-20.7128%</td>
</tr>
<tr>
<td>50</td>
<td>3.9984%</td>
<td>-29.0158%</td>
</tr>
</tbody>
</table>

Table 4.2: Average Percent Differences between Approximate Solutions and MILP Solutions

<table>
<thead>
<tr>
<th>Number of Products</th>
<th>Number of Initial MILP Solutions Found</th>
<th>Percentage of Superior Initial MILP Solutions</th>
<th>Percentage Difference for Worst Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>20</td>
<td>10%</td>
<td>1.44%</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>10%</td>
<td>0.11%</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>10%</td>
<td>2.08%</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>15%</td>
<td>2.47%</td>
</tr>
<tr>
<td>25</td>
<td>20</td>
<td>30%</td>
<td>9.74%</td>
</tr>
<tr>
<td>30</td>
<td>20</td>
<td>20%</td>
<td>3.78%</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>40%</td>
<td>8.80%</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>15%</td>
<td>12.72%</td>
</tr>
</tbody>
</table>

Table 4.3: Percentage of Problems in which Initial MILP Solution is Better than Approximate Solution

### 4.4 Future Work

While the approximate algorithm implemented in this work has demonstrated improvement over exact techniques for the supply chain problem, further improvements could be made to solve even larger problems and to improve solution quality. A current limitation of the algorithm is that while it often finds a near-optimal solution in an early iteration, if a product deadline is early in a day such that its last workcenter cannot be completed before the deadline on that day and the more optimal solution
is to complete it on the previous day, the algorithm does not find the optimal work assignment in an early iteration in general. This was observed in some of the smaller problems tested in this work, and it would likely contribute to a larger deviation from the optimal solution for larger problems. In order to improve this, it would be of value to explore alternate heuristic techniques for modifying the work allocation to days at the highest level.

Additional improvements could be obtained by modifying the bottom two levels of the algorithm as well. At the middle level of the algorithm, the current strategy is to initialize labor at the lowest possible amount and increment labor one person at a time until a feasible solution is found. While this is effective with a small number of possible labor allocations, it might be a prohibitively slow strategy for a larger range of possibilities for labor allocations. Alternate heuristic search strategies could be employed at the middle level to more quickly reach a feasible labor allocation. Finally, the lowest level MILP scheduling problem remains the bottleneck for the approximate technique. In order to solve larger problems, exploring ways to further decompose the scheduling level problem will be necessary.
Chapter 5

Conclusion

The main objectives of this thesis included modeling individual supply chain components, developing an integrated supply chain model for better end-to-end optimization of processes as products flow through the supply chain, and exploration and development of decomposition techniques to improve solve times of these large-scale problems. To this end, MILP models of three components of the supply chain including the manufacturing plant, the distribution center, and consolidation transport were developed and their solve times were evaluated using a state-of-the-art optimizer, Gurobi. Additionally, an integrated model incorporating aspects of all three individual supply chain component models was developed and solve times were evaluated. Benders decomposition was applied to the manufacturing plant model as a baseline for studying the efficacy of exact decomposition techniques on large-scale supply chain problems. Finally, an approximate hierarchical algorithm incorporating heuristic techniques was developed and evaluated for the manufacturing plant problem in order to further study how non-exact decomposition techniques impact runtimes for these large-scale problems.

5.1 Summary of Results

The main contribution of chapter 2 was the development and evaluation of each of the three supply chain component models along with the integrated model. The primary
findings from the evaluation include the following:

- The manufacturing plant MILP model solves problems with sizes of up to 50 products, three workcenters, a maximum labor of five people per workcenter, and a time horizon of 10 days within a one hour timeframe.

- The consolidation transport MILP model solves problems with sizes of up to 800 products for a 10 day time horizon within a one hour timeframe.

- The distribution center MILP model solves problems with sizes of up to 11 products within a one hour timeframe.

- The integrated MILP model solves problems with sizes of up to 15 products, three workcenters, a maximum labor of five people per workcenter, and a time horizon of 10 days within a one hour timeframe.

- The size of problem that can be solved for each of the models developed in chapter 2 is limited by the increase in number of decision variables and constraints as problem size increases. In particular, in the manufacturing plant, distribution center, and integrated models, the number of decision variables varies quadratically with an increase in number of products.

- The state-of-the-art optimizer, Gurobi, solves problems two orders of magnitude smaller than necessary for real-world supply chain problems in the cases of the manufacturing plant model, the distribution center model, and the integrated model. For the consolidation transport model, problems one order of magnitude smaller than necessary are solved.

Initial results suggest that a standard MILP solver, such as Gurobi, would not on its own be sufficient to solve real-world supply chain-sized problems in a timeframe of under one hour. The timeframes necessary to solve problems of relevant size would be too large to keep up with the fast-paced supply chain environment as orders come in and change and as there are changing capabilities related to facilities and
transportation in the process. As problems grow in size, it becomes intractable to solve large enough problems at all.

The main contribution of chapter 3 was the exploration of exact decomposition techniques as a method of improving runtimes and increasing the size of problem that can be solved in a relevant timeframe. The primary findings include the following:

- The application of Benders decomposition to the manufacturing plant problem resulted in worse performance than optimization of the original MILP formulation with Gurobi. On average, for small problems that were solvable with both the MILP formulation and Benders decomposition, Gurobi outperformed Benders decomposition by three orders of magnitude.

- The inferior solve times with the application of Benders decomposition are likely due to the large number of discrete decision variables as compared with the number of continuous decision variables in the manufacturing plant problem. This can lead to a large number of feasibility cuts compared to optimality cuts and overall slow convergence. In the problems that were solved using Benders decomposition, there were approximately 10 times as many feasibility cuts as optimality cuts on average.

Results from the implementation of Benders decomposition for the manufacturing plant problem suggested that an exact technique like Benders decomposition will not improve runtimes for a problem when the model has a large number of integer- and binary-valued decision variables relative to the number of continuous decision variables as the manufacturing plant model does. In cases in which the discrete part of a MILP model can be separated into independent parts, it might be possible to attain runtime improvements by further decomposing the problem and solving individual pieces in parallel. However, when binary and integer decision variables are highly coupled, as they are in the manufacturing plant problem, other techniques will need to be applied in order to improve runtimes.

The main contribution of chapter 4 was the development of an approximate decomposition technique and its application to the manufacturing plant problem. The
primary findings include the following:

- The approximate technique solved manufacturing plant problems faster than the Gurobi optimizer with the MILP formulation in all cases in which both techniques solved the problem before a one hour timeout. Improvements were greater for larger problems.

- The approximate technique arrived at its final solution before the Gurobi optimizer found its first solution for the MILP formulation when considering the median solve times for each. Again, the improvement was greater for larger problems.

- The approximate technique could solve problems twice as large as those solved with the Gurobi MILP formulation within a one hour time horizon.

- A bottleneck caused by the lowest level MILP in the approximate technique will limit it to solving problems with a size of about 140 products.

- Solution quality for the approximate technique declined as problem size increased, although all approximate solutions were within 20.35% of the solution outputted by Gurobi in cases in which both techniques solved the problem.

The above results show promise for solving larger-scale problems using similar techniques. However, the approximate technique is currently able to solve problems on the order of 100 products, and in the future, problems on the order of 1000 products will need to be solved. Nonetheless, the approximate algorithm applied here is a first step towards solving problems of larger scale.

5.2 Future Work

Three supply chain components and an integrated model were developed and evaluated as part of this work, and decomposition techniques were applied to study improvements in solution times. While initial steps towards solving larger-scale inte-
grated problems were achieved, there are a number of important future directions to research further.

One major remaining problem is the expansion of the integrated model to include a fuller version of the supply chain. The integrated model in this work includes one manufacturing plant, one set of consolidation transport vehicles, and one distribution center. In the real-world supply chain, multiple manufacturing plants develop products for multiple distribution centers, and there are also multiple distribution center steps in the end-to-end supply chain. Expansion of the integrated model to reflect these aspects of the supply chain will be important in optimizing supply chain processes with fuller information. It will also be important to develop and incorporate additional supply chain components, such as outbound transportation, moving forward. As expanded models are developed, it will be increasingly possible to schedule each step with knowledge of the other downstream and upstream steps. However, expanded integrated models add complexity to the optimization problem, and problems of relevant size will become increasingly large and difficult to solve with current techniques. One of the primary challenges in developing expanded integrated models will be understanding if decomposition techniques like the one developed in this thesis will suffice or if new techniques will need to be developed.

Another area to explore in the future is how to best model stochasticity in execution times in the supply chain processes. In this thesis, all decisions were assumed to be deterministic. In real-world supply chain problems, variability in outputs and deviations from scheduled work can occur, and plans might not be followed exactly. A challenge in considering stochasticity will be understanding where deviations are more and less likely to occur and how to best model supply chain components with these factors in mind. It will also be important to model stochasticity while keeping tractability in mind, a problem which may pose a further challenge.

In this thesis, all models were formulated as MILPs and optimized using a state-of-the-art optimizer. While the MILP framework lends itself well to the expression of problem decision variables, objectives, and constraints, other frameworks such as CPs, STNs, RTNs, and POMDPs among others, might optimize certain aspects of the
supply chain better than the MILP framework. A full study of different frameworks applied to the supply chain components and the integrated model would be of benefit in the understanding of the relative strengths and weaknesses of each framework as it applies to this type of problem. Further, a deeper knowledge of the relative strengths and weaknesses of each framework could shed light on better hybrid approaches for solving supply chain problems.

Finally, the supply chain component models developed in this thesis were based on a single supply chain, and parameters and constraints of each model were chosen to be representative of this single example. All of these parameters and constraints were also assumed to be correct and sufficient to describe the processes in the supply chain. In reality, parameters and constraints might not be fully understood, and it might not be possible to exhaustively list the most important aspects of the problem. Because of this, models that are developed are inherently limited in their descriptive power. In the future, it could be of benefit to explore machine learning techniques to better incorporate important model features. This could both improve model quality and allow models that are developed to generalize more easily.

The work done in this thesis was a first step towards modeling and solving large-scale integrated supply chain problems in less time than it takes standard optimizers to solve them. Initial solve time improvements were achieved with the approximate decomposition technique developed in this work, and future work could further improve both models and solve times associated with optimizing these models.
Appendix A

Benders Decomposition Formulations for the Manufacturing Plant Problem

A.1 Benders Optimality Cut Formulation

\[- \sum_{l \in L} cycleTimes_{jl} \cdot workcenters_{kj} \cdot K_{ijkl} \cdot u0_{oijk} \]
\[+ (\sum_{l \in L} cycleTimes_{jl} \cdot workcenters_{kj} \cdot K_{ijkl} + M_1 \cdot workcenters_{k,j+1}) \cdot u1_{oijk} \]
\[+ (hoursPerShift - \sum_{l \in L} cycleTimes_{jl} \cdot workcenters_{kj} \cdot Y_{ijl}) \cdot u2_{oijk} \]
\[+ 0 \cdot u3_{oijkm} + 2 \cdot hoursPerShift \cdot D2_{ijkkm} \cdot u4_{oijkm} + 0 \cdot u5_{oijkm} \]
\[+ 2 \cdot hoursPerShift \cdot D1_{ijkkm} \cdot u6_{oijkm} \]
\[+ M_2 \cdot (2 - workcenters_{kj} \cdot F_{ijk} - workcenters_{mj} \cdot F_{ijm}) \cdot u7_{oijkm} \]
\[- \sum_{l \in L} cycleTimes_{jl} \cdot workcenters_{kj} \cdot workcenters_{mj} \cdot Y_{ijl} \cdot u7_{oijkm} \]
\[- \sum_{i \in I} hoursPerShift \cdot i \cdot F_{ijk} \cdot u8_{oijk} \]
\[+ \sum_{i \in I} hoursPerShift \cdot i \cdot F_{ijk} \cdot u9_{oijk} \]
\[+ \sum_{i \in I, j \in L} cycleTimes_{last,l} \cdot workcenters_{k, last} \cdot K_{i, last, k, l} \cdot u10_{ok} \]
\[- \sum_{i \in I, l \in L} \text{cycleTimes}_{l, i} \cdot \text{workcenters}_{k, l} \cdot K_{i, l} \cdot u_{11, ok} \]
\[+ (\text{deadlines}_k + 2 \cdot \text{hoursPerShift} \cdot \text{timeHorizon} \cdot T_{1, k}) \cdot u_{12, ok} \]
\[- \text{deadlines}_k \cdot u_{13, ok} \]
\[+ (2 \cdot \text{hoursPerShift} \cdot \text{timeHorizon} \cdot T_{2, k} - \text{deadlines}_k) \cdot u_{14, ok} \]
\[+ \text{deadlines}_k \cdot u_{15, ok} \leq \theta \quad \forall o \in O, i \in I, j \in J, k \in K \quad (A.1) \]

A.2 Feasibility Cut Formulation

\[- \sum_{l \in L} \text{cycleTimes}_{j, l} \cdot \text{workcenters}_{k, l} \cdot K_{i, j} \cdot u_{0, fijk} \]
\[+ (\sum_{l \in L} \text{cycleTimes}_{j, l} \cdot \text{workcenters}_{k, l} \cdot K_{i, j} + M_1 \cdot \text{workcenters}_{k, j+1}) \cdot u_{1, fijk} \]
\[+ (\text{hoursPerShift} - \sum_{l \in L} \text{cycleTimes}_{j, l} \cdot \text{workcenters}_{k, l} \cdot Y_{ijl}) \cdot u_{2, fijk} \]
\[+ 0 \cdot u_{3, fijkm} + 2 \cdot \text{hoursPerShift} \cdot D_{2, ijkm} \cdot u_{4, fijkm} + 0 \cdot u_{5, fijkm} \]
\[+ 2 \cdot \text{hoursPerShift} \cdot D_{1, ijkm} \cdot u_{6, fijkm} \]
\[+ M_2 \cdot (2 - \text{workcenters}_{k, j} \cdot F_{ijk} - \text{workcenters}_{m, j} \cdot F_{ijm}) \cdot u_{7, fijkm} \]
\[- \sum_{l \in L} \text{cycleTimes}_{j, l} \cdot \text{workcenters}_{k, l} \cdot \text{workcenters}_{m, j} \cdot Y_{ijl} \cdot u_{7, fijkm} \]
\[- \sum_{i \in I} \text{hoursPerShift} \cdot i \cdot F_{ijk} \cdot u_{8, fjk} \]
\[+ \sum_{i \in I} \text{hoursPerShift} \cdot i \cdot F_{ijk} \cdot u_{9, fjk} \]
\[+ \sum_{i \in I} \text{cycleTimes}_{l, i} \cdot \text{workcenters}_{k, l} \cdot K_{i, l} \cdot u_{10, fk} \]
\[- \sum_{i \in I, l \in L} \text{cycleTimes}_{l, i} \cdot \text{workcenters}_{k, l} \cdot K_{i, l} \cdot u_{11, fk} \]
+ (deadlines$_k + 2 \cdot \text{hoursPerShift} \cdot \text{timeHorizon} \cdot T1_k) \cdot u_{12f_k}
- \text{deadlines}_k \cdot u_{13f_k}
+ (2 \cdot \text{hoursPerShift} \cdot \text{timeHorizon} \cdot T2_k - \text{deadlines}_k) \cdot u_{14f_k}
+ \text{deadlines}_k \cdot u_{15f_k} \leq 0, \quad \forall f \in F, i \in I, j \in J, k \in K \quad \text{(A.2)}

### A.3 Full Formulation of Benders Dual Subproblem

\[
\begin{align*}
\max & \quad \sum_{l \in L} \text{cycleTimes}_{jl} \cdot \text{workcenters}_{kj} \cdot K_{ijk} \cdot u_{0fijk} \\
& + \left( \sum_{l \in L} \text{cycleTimes}_{jl} \cdot \text{workcenters}_{kj} \cdot K_{ijk} + M_1 \cdot \text{workcenters}_{k,j+1} \right) \cdot u_{1fijk} \\
& + \left( \text{hoursPerShift} - \sum_{l \in L} \text{cycleTimes}_{jl} \cdot \text{workcenters}_{kj} \cdot Y_{ijl} \right) \cdot u_{2fijk} \\
& + 0 \cdot u_{3fijk} + 2 \cdot \text{hoursPerShift} \cdot D_{2fijkm} \cdot u_{4fijkm} + 0 \cdot u_{5fijkm} \\
& + 2 \cdot \text{hoursPerShift} \cdot D_{1fijkm} \cdot u_{6fijkm} \\
& + M_2 \cdot (2 - \text{workcenters}_{kj} \cdot F_{ijk} - \text{workcenters}_{mj} \cdot F_{ijm}) \cdot u_{7fijkm} \\
& - \sum_{l \in L} \text{cycleTimes}_{jl} \cdot \text{workcenters}_{kj} \cdot \text{workcenters}_{mj} \cdot Y_{ijl} \cdot u_{7fijkm} \\
& - \sum_{i \in I} \text{hoursPerShift} \cdot i \cdot F_{ijk} \cdot u_{8fjk} \\
& + \sum_{i \in I} \text{hoursPerShift} \cdot i \cdot F_{ijk} \cdot u_{9fjk} \\
& + \sum_{i \in I, l \in L} \text{cycleTimes}_{last,l} \cdot \text{workcenters}_{k,last} \cdot K_{i,last,k,l} \cdot u_{10f_k} \\
& - \sum_{i \in I, l \in L} \text{cycleTimes}_{last,l} \cdot \text{workcenters}_{k,last} \cdot K_{i,last,k,l} \cdot u_{11f_k}
\end{align*}
\]
\[
+ (\text{deadlines}_k + 2 \cdot \text{hoursPerShift} \cdot \text{timeHorizon} \cdot \bar{T}_1) \cdot u_{12f_k} \\
- \text{deadlines}_k \cdot u_{13f_k} \\
+ (2 \cdot \text{hoursPerShift} \cdot \text{timeHorizon} \cdot \bar{T}_2 - \text{deadlines}_k) \cdot u_{14f_k} \\
+ \text{deadlines}_k \cdot u_{15f_k}
\]  
(A.3)

\[
u_{0_{ijk}} - u_{1_{ijk}} + \text{workcenters}_{kj} \cdot \bar{F}_{ijk} \cdot u_{2_{ijk}} - u_{3_{ijkm}} + u_{4_{ijkm}} + u_{5_{ijkm}} - \\
u_{6_{ijkm}} + \sum_{i \in I} \bar{F}_{ijk} \cdot u_{8_{jk}} - \sum_{i \in I} \bar{F}_{ijk} \cdot u_{9_{jk}} \geq 0 \quad \forall i \in I, j \in J, k, m \in K
\]  
(A.4)

\[-u_{0_{i,j+1,k}} + u_{1_{i,j+1,k}} \geq 0 \quad \forall i \in I, j \in J, k \in K
\]  
(A.5)

\[u_{3_{ijkm}} - u_{4_{ijkm}} - u_{5_{ijkm}} + u_{6_{ijkm}} \geq 0 \quad \forall i \in I, j \in J, k, m \in K
\]  
(A.6)

\[-u_{3_{ijkm}} + u_{4_{ijkm}} - u_{5_{ijkm}} + u_{6_{ijkm}} - u_{7_{ijkm}} \geq 0 \quad \forall i \in I, j \in J, k, m \in K
\]  
(A.7)

\[-u_{8_{jk}} + u_{9_{jk}} \geq 0 \quad \forall j \in J, k \in K
\]  
(A.8)

\[-u_{8_{\text{last},k}} + u_{9_{\text{last},k}} - u_{10_k} + u_{11_k} \geq 0 \quad \forall k \in K
\]  
(A.9)

\[u_{10_k} - u_{11_k} + u_{12_k} - u_{13_k} - u_{14_k} + u_{15_k} \geq 0 \quad \forall k \in K
\]  
(A.10)

\[u_{12_k} - u_{13_k} + u_{14_k} - u_{15_k} \geq 1 \quad \forall k \in K
\]  
(A.11)
Bibliography


