The Dissertation Committee for Susan Kathleen Monkman Certifies that this is the approved version of the following dissertation:

Scheduling of Product Families on Multiple, Identical Parallel Production Lines to Minimize Setup Costs

Committee:

Douglas J. Morrice, Supervisor

Edward G. Anderson

Jonathan F. Bard

Stephen Gilbert

John Hasenbein

Leon Lasdon

Scheduling of Product Families on Multiple, Identical Parallel Production Lines to Minimize Setup Costs

by

Susan Kathleen Monkman, B.A.; M. Eng.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2006

Dedication

I dedicate this dissertation to God and to my husband, Jay.

I could not have done it without them.

And also to my parents, whose support is invaluable.

Acknowledgements

Several people have been very helpful to me during this process. I would like to especially acknowledge the time that Douglas Morrice and Jonathan Bard spent with me and the standards they set for me. I would also like to thank Jennifer Loveland, who was the best company contact and project partner that I could have wished for. And finally, I would like to thank some fellow doctoral students who went before me and encouraged me, including Dr. Guzin Bayraksan and Lt. Colonel Donna Korycinski, PhD.

Scheduling of Product Families
on Multiple, Identical Parallel Production Lines
to Minimize Setup Costs

Publication No._____

Susan Kathleen. Monkman, PhD.

The University of Texas at Austin, 2006

Supervisor: Douglas J. Morrice

This dissertation addresses and solves a real-world production scheduling problem that was discovered early in 2003 at Dell, Inc.'s Morton L. Topfer Manufacturing Center (TMC). With continually increasing product variety and production volumes, TMC had reached the designed limits of the factory. When looking forward to 2004, TMC expected a volume increase of 20 percent and a doubling of product variety. Initially, no product family scheduling method was in place. This problem is a product family scheduling problem on multiple, identical parallel assembly lines with sequence-dependant setup costs and the goal of minimizing total setup costs.

This dissertation solves the scheduling problem using three different approaches. The first approach was used to resolve the company's immediate scheduling difficulties and assumes that setup costs are similar enough to be treated as sequence independent. As a result of the implementation of this approach, Dell accommodated an effective production volume increase of 35 percent and a doubling in product variety. Furthermore,

Dell realized a conservative cost avoidance of over one million dollars annually, primarily from avoided overtime.

The second approach explicitly considers the unusual sequence-dependant setup costs. We developed a three-step heuristic that involves assignment, sequencing, and time scheduling steps, each with an optimization component. For the most complex step we developed a greedy randomized adaptive search procedure (GRASP). We compared the setup costs resulting from using this approach against the costs using the first approach. Empirical results show the new approach achieved a reduction in factory-wide setups costs of 14 to 21 percent with single line setup cost reductions of 0 to 49 percent.

The final approach tackles the difficult problem of combining the assignment and sequencing steps. The combined model theoretically dominates the second approach but it is both NP-hard and very large so cannot be solved to optimality. Instead, we develop a new GRASP for this combined model. We compare the setup costs using this final approach with the setup costs achieved by the second approach. Results show an average increase in setup costs of 46 percent, with a range of 27 to 60 percent increase.

Table of Contents

List of Tables x

List of Figures xi

Chapter 1 Introduction and Literature Review 1

Chapter 2 The Dell Scheduling Heuristic 7

2.1 Introduction 7

2.2 Background 8

2.2.1 Product Description 8

2.2.2 Factory Configuration 8

2.2.3 Setup Description 9

2.2.4 Production Planning 10

2.3 New Situation 11

2.4 Problem Description 12

2.5 Approach and Assumptions 14

2.6 Assignment Optimization Model (Step 1) 16

2.7 Lane Assignment and Sequencing (Step 2) 21

2.8 Product Family Scheduling (Step 3) 22

2.9 Full Heuristic Procedure 23

2.10 Implementation and Results 25

2.11 Challenges and Lessons Learned 27

Chapter 3 A Separated Models Scheduling Heuristic 30

3.1 Introduction 30

3.1.1 Problem Description 31

3.1.2 Approach 33

3.1.3 Related Literature 34

3.2. Model Formulations 36

3.2.1 Assignment Model 38

3.2.2 The Sequencing Model 42

3.2.3 Time Scheduling Model....45
3.3 Solution Approaches....49
3.3.1 Solving the Assignment Model....49
3.3.2 Solving the Sequencing Model....50
Initialization:....51
Phase 1 (solution construction):....52
Phase 2 (node elimination):....53
Phase 3 (node swap):....54
3.3.3 Solving the Time Scheduling Model....55
3.3.4 Solving the LMM Heuristic....55
3.4. Empirical Analysis....56
3.5 Conclusion and Future Research....60
Chapter 4 A Combined Model Scheduling Heuristic....62
4.1 Introduction....62
4.2 Combined Model....64
4.3 Solving the Combined Model....67
4.3.1 GRASP Discussion....67
4.3.2 GRASP Pseudo-code....68
Phase 0: Initialization....68
Phase 1: Develop feasible solutions....68
Phase 2a: Solution Improvement by Node Elimination....70
Phase 2b: Solution Improvement by Neighboring Node Swaps.70
4.3.3 GRASP Parameters and Other Notes....71
4.3.4 GRASP Constraint Enforcement....72
4.4 Empirical results....72
4.5 Conclusions....74
Chapter 5 Conclusions and Future Research....76
References....80
Vita....83

List of Tables

Table 1: Percentage Reduction of Total Setup Costs for Each Scenario of the New Heuristic over the LMM Heuristic.....57

Table 2: Percentage Reduction in Setup Costs of the New Heuristic over the LMM Heuristic by Production Line. 58

Table 3: Percentage Reduction in Setup Costs of the New Heuristic over the LMM Heuristic by Number of Families.....59

Table 4: Percentage Reduction of Total Setup Costs for Each Scenario of the Combined Model Heuristic over the Separated Models Heuristic ...73

List of Figures

Figure 1: Layout of a single production line with N_a chassis lanes, ...9

Figure 2: Layout of a single production line with L chassis lanes,.....32

Figure 3. Graph representation of sequencing problem.....43

Chapter 1 Introduction and Literature Review

This dissertation addresses and solves a real-world production scheduling problem that was discovered early in 2003 at Dell, Inc.'s Morton L. Topfer Manufacturing Center (TMC). TMC was designed to produce high volumes of a wide variety of make-to-order products. From the opening of this factory in 2000 to the beginning of this research project in early 2003, TMC offered a steadily increasing level of product variety. With this continually increasing product variety, as well as continually increasing production volumes, by early 2003 TMC had reached the designed limits of the factory in terms of the product variety it could accommodate. When looking forward to 2004, TMC expected a additional volume increase of 20 percent and a doubling of product variety in a very short period of time.

At the beginning of this research project, no scheduling method was in place for determining when the different product families should be produced on each production line. This was because, when the product variety was within the design limits of the factory, no scheduling method was needed – the production lines could be configured such that an order for a product from any family could be built on at least one of the lines any time during the production shift. As the product variety increased, this was no longer the case, and a method had to be developed to determine when each production line would be set up to produce products from each production family. This production scheduling problem can be summarized as a product family scheduling problem on multiple, identical parallel assembly lines with sequence-dependant setup costs with the goal of minimizing total setup costs.

This dissertation solves the scheduling problem using three different approaches. The first approach was the approach used to resolve the company's immediate scheduling

difficulties and assumes that setup costs are similar enough to be treated as sequence independent. As a result of the implementation of this approach, TMC was able to accommodate an effective production volume increase of 35 percent and a doubling in product variety. Furthermore, Dell realized a conservative cost avoidance of over one million dollars annually, primarily from avoided overtime.

The second approach we used explicitly considers the unusual sequence-dependant setup costs. We developed a three-step heuristic that involves assignment, sequencing, and time scheduling steps to produce a production schedule. Each of these steps uses a separate optimization program. For the most complex step, the sequencing step, we developed a greedy randomized adaptive search procedure (GRASP). We compared the setup costs resulting from using this approach against the costs using the first approach. The empirical results show that the new approach achieved a reduction in factory-wide setups costs of 14 to 21 percent with single line setup cost reductions of 0 to 49 percent.

The final approach tackled the difficult problem of combining the optimization models used in the assignment and sequencing steps. Theoretically, the combined model provides solutions that dominate the separated models approach but the combined model is both NP-hard and very large so it cannot be solved to optimality. Instead, we develop a new GRASP for this combined model. The interesting question is whether or not this combined model, when solved heuristically, would still outperform the separated models. We compare the setup costs using this combined model GRASP with the setup costs achieved by the second approach. Results show an average reduction in setup costs of 56 percent, with a range of 44 to 61 percent.

According to Allahverdi et al.'s (1999) rendition of Lawler et al.'s (1993) standard three field notation, the problem studied in this dissertation can be written most

closely as P/$SC_{sd,b}$/TSC indicating that it is a problem with identical parallel machines (P) with sequence dependant batch setup costs ($SC_{sd,b}$) where the objective is to minimize the total setup costs (TSC). Most of the research on scheduling machines with sequence dependant setups is concerned with optimizing some due date related measure that is negatively impacted by setup times. The survey paper by Cheng et al. (2000) mentions only papers that are concerned with setup times, and the survey paper by Allahverdi et al. (1999) shows that only 18 of the 156 papers reviewed are concerned with minimizing total setup costs rather than a due-date related measure. Since the problem this dissertation addresses does not have formal due dates for the jobs, and thus leads to very different modeling formulations than research considering due dates, only the literature concerned with setup costs will be discussed here.

The three field notation for the current research as stated above does, however, leave out a very important difference between the problem studied here and the problems in existing research. The fact that a single line can be set up to manufacture products from three different families without incurring setup costs between products within these three families leads to a kind of problem that seems to have not been addressed previously. Since the three product families that are on the line at one time can consist of any subset of three of the families produced by the factory the problem is not a traditional batch scheduling problem, even though we are, in essence, batching orders for production by product family. This means that all the previous research on job or batch scheduling has limited relevance to the current problem. The most closely related research is discussed below.

The production scheduling problem in existing research that most closely resembles the current problem was presented in Balakrishnan and Vanderbeck (1999). In their problem products from a set product families must be produced on several parallel

identical machines in order to minimize setup costs and balance the workload across the lines. Each family required a different setup, and the setup consisted of swapping out types of parts, as in the current problem. One primary difference is that each line can only assemble products from one family without requiring a setup. Another major difference is that Balakrishnan and Vanderbeck (1999) adopt a setup policy that makes the setups sequence-independent.

One other problem with similar structure presented in the literature arises in flexible manufacturing factories where individual machines are fitted with tool magazines that hold a variety of tools to work on each job. When two jobs are processed consecutively and all the tools required for both jobs are in the tool magazine, then no setup is required between these jobs. When all tools are available for the first job, but a tool is required for a second job that is not in the magazine, then a setup is required. The problem of minimizing these tool setups has been addressed by Al-Fawzan and Al-Sultan (2002) and by Crama and van de Klumdert (1999), among others. The primary differences between the scheduling problem dealing with tool magazines and our scheduling problem is that in our scheduling problem the jobs can be divided into product families and production volumes are very high. These differences alter the modeling approaches that can be used to solve the problem.

In the first two approaches presented in this paper the assignment step is modeled separately from the sequencing step. One paper that provides a review of literature related to both the assignment and sequencing problems found in chemical process scheduling is provided by Pinto & Grossmann (1988). The assignment step in the current problem is a variation of the generalized assignment problem where instead of assigning tasks in order to not exceed a resource's capacity, here we want to assign demand to each line in order to exceed the available line capacity and thereby keep each line busy for the whole shift.

A survey of older work on the generalized assignment problem is given by Cattrysse & Van Wassenhove (1992). More recently, a paper has appeared with a very similar variation of the assignment problem, found in the dairy industry (Foulds & Wilson, 1997). The primary difference between the Foulds & Wilson model and the model we propose is that their model does the equivalent of requiring that all the demand for a single product family be produced by only one line.

Methodologically, the literature discussing modifications of the traveling salesman problem (TSP) are important. The sequence-dependant sequencing problems presented in this dissertation can be modeled as traveling salesman subtour problems (TSSP), sometimes called traveling salesman subset-tour problems (ref) which is also abbreviated TSSP in the literature (Mittenthal and Noon, 1992). Renaud and Boctor (1998) provide a nice overview of particular cases of the TSSP, including the traveling purchaser problem (TPP) and the covering salesman problem. The TPP was originally introduced by Ramesh (1981), and consists of a purchaser that must purchase a set of commodities from a set of locations where each commodity is sold in at least one location, and each locations sells at least one commodity. The goal is to minimize travel costs and purchasing costs, as different locations may sell each commodity at a different price. Our problem is more complex than most, though, due to a required constraint modification that requires more than one visit to some subsets of nodes and to the fact that the subsets of nodes heavily overlap. The requirement to visit more that one node in a subset does arise in the capacitated TPP though, which is the most similar TSSP to our problem. The capacitated TPP is presented in Laporte et al. (2003), where they developed a branch-and-cut algorithm using polyhedral theory to derive several classes of valid inequalities. Although the capacitated TPP is similar to our problem, it is unclear how

well the Laporte et al. algorithm would perform on our problem and how well it could be implemented in the real world situation.

To solve the large, NP-hard TSSP models we use a heuristic method called the greedy randomized adaptive search procedure (GRASP). Therefore, that literature is also relevant. GRASP was initially developed in Feo and Bard (1989), and Feo et al. (1991). Feo and Resende (1995) provide a detailed description of the common structure and development of the procedure, as well as a review of applications. Since then, GRASP has been implemented in many different works including vehicle routing, machine scheduling, and lot-sizing, to name a few. Our research demonstrates the usefulness of GRASP for sequencing with unusual sequence dependant setup costs.

The combined assignment and sequencing model used in the third solution method in this dissertation is unusual and not readily found in existing literature. The most closely related problem is the multiple vehicle routing problem (MVRP). In the MVRP, multiple vehicles must be routed from a depot through numerous locations to either pickup or deliver a product (e.g., Boudia et al., 2006). Our problem differs significantly from a MVRP though, because it does not require a visit to every node. Rather, our combined assignment and sequencing problem retains the subset-tour characteristic of the single-line sequencing problem. Therefore the combined problem could be referred to as a multiple vehicle routing subset-tour problem (MVRSP). As of this time, we have been unable to discover any research that addresses this particular problem. Some additional related literature is also mentioned within each chapter.

Chapter 2 The Dell Scheduling Heuristic

2.1 INTRODUCTION

This chapter describes a production scheduling problem that has arisen in a high volume assemble-to-order electronics manufacturer and presents a scheduling algorithm to solve this problem. The company studied is Dell, Inc. and the facility where the problem arose is the Morton L. Topfer Manufacturing Center (TMC) in Austin, Texas. Dell's direct, assemble-to-order business model allows the company to do business directly with the customer and assemble products only after an order is placed by a customer (Dell and Magretta, 1998). TMC, opened in 2001, was designed to support the Dell model so it is both flexible and capable of producing a high product variety which makes it atypical given the typical production volume – product variety tradeoff (Stevenson, 2001). This combination of flexibility and high volume gives rise to the interesting nature of the scheduling problem presented here.

The rest of the chapter is organized as follows. We present the original business environment, the changes to the environment, and the details of the resulting problem in Sections 2.2, 2.3, and 2.4, respectively. In Section 2.5 we describe how we approached the problem and what assumptions we needed to make in order to solve the problem. Following this, we use Sections 2.6-2.9 to present the formulation details for the 3 separate steps of the heuristic, as well as the overall heuristic procedure. Then we discuss the implementation and results in Section 2.10 followed by the challenges and lessons learned in Section 2.11.

2.2 BACKGROUND

2.2.1 Product Description

Each computer produced at TMC begins as an order placed with the Dell sales organization. Then the factory is notified, the production of the order scheduled, and the parts acquired by the factory. Within the factory, all the parts required for the computer are grouped together into a kit, and then the computer is assembled, packaged, and shipped to the customer. Each computer requires a chassis (case) and a selection of component parts that are assembled into this chassis.

The computers produced at TMC are grouped into product families to aid in the production planning process. A product family is defined by the chassis required by a product. Hence, the terms "chassis" and "product family" are essentially interchangeable. Each chassis has a variety of components that can be assembled with it. We will refer to a product family's chassis and set of components collectively as the product family's parts. Each family has some components that are not required by any other family and also has components that are used by various other families. Similarly, some components are limited to one family and other components to several families.

2.2.2 Factory Configuration

The factory has multiple identical kitting lines located side-by-side. Figure 1 depicts a rough schematic of one individual kitting line. As shown, each kitting line has spaces designed to hold pallets of chassis and spaces designed to hold boxes of components. Each production line can hold N_a (•1) different types of chassis (one type of chassis per pallet) and N_b (•1) different types of components (one type of component per box). Each chassis space on a line is called a *lane* and each component space is referred to as a *bin*.

When the line is running, a worker selects one chassis from the N_a lanes for the next product to be built and sends the chassis to the next worker down the line. The workers down the line select the correct components for that order from the bins and add them to the chassis. The production computer system keeps track of which chassis are in which lanes and which components are in which bins, and signals the workers to select the correct chassis and components for each custom order.

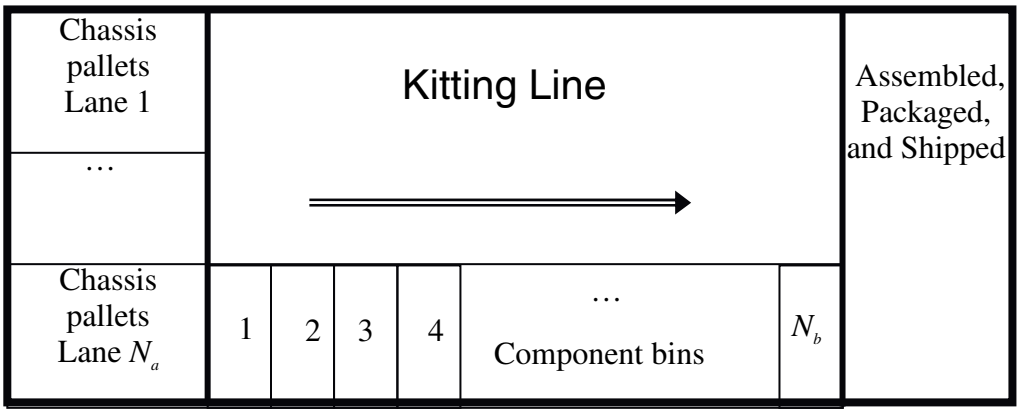

Figure 1: Layout of a single production line with N_a chassis lanes, and N_b component bins.

2.2.3 Setup Description

As is sometimes the case with products that can be grouped into product families (Webster and Baker, 1995), in Dell's situation setups are not required when a product from one family is kitted immediately following another product from the same family. What is unusual in this problem is that no setup is required when kitting a product from one family immediately following a product from a different family if the chassis and parts required by both families are on the kitting line at the same time. Therefore, a setup is only required when a product must be kitted that is from a different family than all the N_a families whose parts are currently on the line.

A setup in this situation entails removing the pallet of chassis for one family from a lane and placing the pallet of chassis required for the next order in that lane. Components must also be changed as well. Those that are not needed for the families of the chassis remaining in the other lanes, or for the family being added to the line, must be removed from the bins. Then, the components that are required for the new family that are not already on the line must be placed in the bins. And, finally, the production computer must be updated with the new information on which parts are in which spaces.

2.2.4 Production Planning

Initially Dell offered customers a choice of about 10 to 12 different computer chassis that were to be assembled in TMC. In addition, Dell offered customers several component choices for each type of chassis. With several parallel kitting lines, each type of chassis could be placed at the front of at least one kitting line at all times, and any components offered with that type of chassis could also be placed on that line. As a result, any computer requiring any chassis and combination of components could be built at any time without having to change which parts were placed on any kitting line. The only exceptions were when the product offerings changed or demand levels switched drastically from one type of chassis to another and the demand change was large enough to cause demand for one type to exceed the capacity of the line or lines on which it was placed. In addition, the enforcement of JIT parts procurement and the kitting of orders on a nearly first come, first serve (FCFS) basis allowed Dell to deliver orders to customers within a very short time window. This ability gave Dell a distinct competitive advantage (Breen, 2004) and eliminated the need to assign formal due dates to each order.

Prior to using our algorithm, Dell assembled customer orders primarily on a FCFS basis according to the date on which an order was placed, but some limited prioritization was used based on customer priorities and order sizes. Product families were assigned to

parallel kitting lines on an ad hoc basis using simple heuristics designed primarily to balance demand across lines. This practice worked reasonably well with 10 to 12 product families, manageable volumes, and excess capacity. Under such circumstances very few setups were required and the schedulers could determine good assignment solutions without the assistance of more sophisticated optimization techniques.

2.3 NEW SITUATION

From 2001 to 2003 the number of product families produced at TMC grew slowly. By early 2003, the number of product families had increased to the point where it had reached the design limits of TMC. Furthermore, in 2004, TMC was facing an overall volume increase of almost 20 percent and an anticipated doubling of product families.

Various factors drove the increase in volume and the number of product families. First, in the face of increasing demand, senior management made a conscious decision to neither expand existing facilities nor significantly increase personnel count in Austin. Instead, demand increases were handled through increased utilization of existing resources. Additionally, Dell experienced product proliferation due to an increase in the number of new product family introductions, increased options for customers within each product family, and longer end-of-life times for old product families. Finally, a change in supply chain strategy dictated that all product families needed to be built at all factories in order to mitigate the risk of a supply shortage, reduce logistics costs, and shorten lead times to customers.

This rapidly increasing product variety threatened Dell's ability to continue producing custom computers at a very high production rate and also threatened to lengthen the short delivery lead-times that were so valuable to them. Once the number of families exceeded the number of lanes on a kitting line, setups became necessary. Since these setups were so rare in the past, there was no method in place to plan or schedule

them. Also, if the factory were to stick with the FCFS policy, a very large number of setups could be required, since a computer requiring any possible chassis could be required at any time. Each setup required significant time and labor so it was not feasible to perform enough setups to continue producing in FCFS order.

The factory began batching orders by product family on-the-fly and delaying orders that required chassis that couldn't fit on the line. When many orders for a family piled up, a setup was then performed to place that chassis onto the line. One consequence was that orders were no longer being produced in FCFS order and some orders could easily get delayed beyond the desired delivery window. Another serious consequence was that unplanned setups often resulted in significantly lower production rates, incurring significant costs. Even if a setup did not cause downtime or slowtime, it still required significant labor on the part of material handlers and forklift drivers to change out the parts. (The difference between slowtime and downtime is that during slowtime the production line is still running but at reduced production rates whereas downtime refers to time when the production line is stopped completely. The lower production rates during slowtime can be caused by such issues as short waits for parts or slower processing times due to working around buildups of parts or sorting through backlogged jobs in the computer system.) A third consequence was that excess parts would pile up because the software that scheduled the orders to be produced also procured the parts for that order. When the kitting of orders was delayed, the parts for those delayed orders began clogging the workspace.

2.4 PROBLEM DESCRIPTION

In light of the impending increase in number of product families, the factory needed a procedure to plan setups in advance to prevent downtime and slowtime as well as the buildup of parts in the work areas. The planning procedure would need to schedule

when the parts for each product family should be on each kitting line during the production shift.

The procedure for scheduling these product families had to provide a schedule in advance of each production shift. It also needed to work with Dell's Factory Planner software (from i2 Technologies) so that an order would be scheduled to be produced only when the correct chassis and parts for that order were on the kitting line. This advanced planning would allow the employees to setup one family's parts while the line was kitting orders requiring one of the other families on the line, effectively eliminating downtime or slowtime caused by setups. It would also make the part-ordering by Factory Planner more accurate since this advanced planning would allow Factory Planner to order only the parts required for the orders for families actually on the line. In order to prevent unacceptable delays in order delivery to the customer, management also determined that the scheduling method must allow for every family to be placed on at least one kitting line for some time each day so orders for that family would not be delayed more than one day.

The majority of academic research involving production scheduling does not explicitly consider the costs of performing setups (Allahverdi et al., 1999 and Allahverdi et al., 2006). In addition, the nature of the setups that must be performed at TMC are somewhat different than setups usually considered in research, since setups are required between jobs from different families only if the parts for the second job's family are not already on the line. One of the problems with characteristics most similar to the problem presented in this chapter arises in flexible manufacturing factories where individual machines are fitted with tool magazines that hold a variety of tools to work on each job. When two jobs are processed consecutively and all the tools required for both jobs are in the tool magazine, then no setup is required between these jobs. When all tools are

available for the first job, but a tool is required for a second job that is not in the magazine, then a setup is required. The problem of minimizing these tool setups has been addressed by Al-Fawzan and Al-Sultan (2002) and by Crama and van de Klumdert (1999), among others. The primary differences between the scheduling problem dealing with tool magazines, and the scheduling problem within TMC, is that the jobs at TMC can be divided into product families, and that TMC runs very high production volumes. These differences alter the modeling approaches that can be used to solve the problem.

2.5 APPROACH AND ASSUMPTIONS

To address this scheduling problem within the constraints set by senior management, we developed an algorithm using a combination of optimization programs and heuristics. As mentioned in Pinto and Grossman (1988), every production scheduling algorithm contains three essential elements: assign tasks to machines, sequence tasks, and schedule when each task must be performed. Following this logic, our scheduling method contains three steps. In step 1 we use a linear mixed-integer optimization program to determine the chassis to line assignments. Step 2 takes the chassis line assignments and determines the sequence in which each family should be placed on each lane within each kitting line. The chassis–to-lane assignments are determined using a linear integer program which balances demand across lanes within each production line. Then the sequence is partially determined by insuring that chassis left in a lane at the end of a previous production period begin the next period in that lane. A heuristic based on demand is used to determine the rest of the sequencing. In step 3 we use a heuristic to determine when each family's parts should be placed on, or removed from, each line. In this step, we ensure that the parts for each family will be on the line for at least enough time to fulfill the expected demand.

We chose to schedule product families instead of actual jobs for several reasons. First, a setup is not required every time a line switches from producing a job from one family to a job from another family. Rather, a setup is only required when a line switches from producing jobs from the set of families currently on the line to producing a job from a family not on the line. Second, there are no formal due dates for each job so modeling at that level of detail is not required. Third, the volume of demand is so high that an optimization model that included every individual job would take an unacceptably long time to solve. And finally, some customer order information is not known at the time that the schedule must be input into Factory Planner, so every job could not possibly be scheduled before the beginning of the shift.

Our overall approach contains several simplifying assumptions that can be justified by the application context. Although demand is not known completely when a schedule must be produced, the average forecasted demand is known with a high degree of certainty within the 1-shift scheduling horizon. In addition, the flexibility of the production lines, in terms of being able to kit products from several families without a setup, results in the absorption of moderate shifts in expected demand from one family to another. We also assume constant homogeneous kitting times. This approximation is acceptable because production volumes are very high and the kitting process is highly repetitive, so kitting times are fairly uniform. Our scheduling horizon is assumed to be a single shift since this satisfied Dell's planning needs. It is important to note that even though the models are single period, they do provide some linkage between periods since they account for which chassis types were on each line and in each lane at the end of the previous period. Finally, we assume that the cost for each setup is roughly the same, regardless of which family is being removed from, and which is being added to, the line. While this is not exactly true in reality, it was considered to be a good enough

approximation for the initial assignment of product families to lines and it avoids additional complexity associated with sequencing. Ongoing research is being conducted to study the impact of random demand and planning over multiple time periods (e.g., Tanrisever and Morrice, 2005). Chapters 3 and 4 address this problem with sequence-dependant setup costs.

2.6 ASSIGNMENT OPTIMIZATION MODEL (STEP 1)

Very early in the project, an initial mixed integer program (MIP) was formulated and used to assign product families to lines. The primary objective of the original MIP was to formalize the simple existing heuristic by assigning families to lines in order to balance the amount of work assigned to each kitting line. Since the number of families had just begun to exceed the design limits of the factory, the program also kept the number of lines with more families than lanes to a minimum while insuring that families with very high demands were placed on more than one line so that the demand for them could be met. Other objectives included not moving a chassis from one line to another unless necessary and making sure that families were assigned to kitting lines in such a way that they could be routed correctly through the rest of the factory.

In order to change the focus of the product family assignments from line balancing to minimizing the number of setups we formulated a mixed integer, linear programming model to minimize the total number of setups, thereby minimizing the setup costs. Based on previously stated assumptions, this model uses the average forecasted demand for each family. The model accounts for the fact that there is limited amount of space on the kitting lines for the component parts, so it also tries to place chassis that require some of the same components together on the same kitting line. This grouping of chassis by similar components is done by minimizing the largest number of components assigned to a line that is in excess of the number of available component

spaces. The two objectives are weighted such that the benefit of reducing the excess number of parts on a line will never outweigh the penalty of adding another setup, and thus, reducing parts on a line will never cause an additional setup.

The indices used in the model are:

i index for chassis; 1,…,N_f ; without loss of generality the chassis are ordered from those with the highest demand to those with the lowest demand

j index for lines; 1,…,N_l

k index for components; 1,…,N_p

The model variables are defined as follows:

x_{ij} 1 if chassis i is assigned to line j; 0 otherwise,

y_{ij} percentage of demand for chassis i assigned to line j,

s_{ij} 1 if a setup will be required to assemble i on line j; 0 otherwise,

r_{jk} 1 if part k is assigned to line j (is required on line j); 0 otherwise,

z_j the number of parts required by line j,

z the largest number of parts required on any line in excess of the number of available component spaces.

The model parameters are:

a_{ij} 1 if chassis i was on line j at the end of the previous shift; 0 otherwise,

p_{ik} 1 if chassis i requires part k; 0 otherwise,

d_i demand for chassis i (any demand backlogged from last period plus new demand); without loss of generality, chassis are indexed from that with the highest demand to that with the lowest demand

N_c capacity of each line in units of demand,

N_f number of product families,

N_l number of lines,

N_a number of lanes on a line,

N_p number of components,

N_h number of chassis that are high runners (families with very high demand)

L a weight indicating the relatively large effort required for an entire setup vs. the relatively small effort required for the additional part swaps needed when excessive parts are assigned to at least one line

N_b number of component spaces available on a line to hold different types of component parts.

The assignment model is given below. A solution assigns demand and parts for all product families to lines, ensures that enough demand is assigned to every line in order to keep it fully utilized, minimizes the number of setups, and minimizes the maximum number of components on any line.

$$\textit{Minimize} \quad L\sum_{i=1}^{N_f}\sum_{j=1}^{N_l} s_{ij} + z \tag{1a}$$

$$\textit{Subject to} \quad \sum_{j=1}^{N_l} y_{i,j} = 1 \qquad i = 1,...,N_f \tag{1b}$$

$$x_{ij} \geq y_{ij} \qquad i = 1,...,N_f,\ j = 1,...,N_l \tag{1c}$$

$$s_{ij} \geq x_{ij} - a_{ij} \qquad i = 1,...,N_f,\ j = 1,...,N_l \tag{1d}$$

$$\sum_{i=1}^{N_f} y_{ij} d_i \geq N_c \qquad j = 1,...,N_l \tag{1e}$$

$$\sum_{i=1}^{N_f} x_{ij} \geq N_a \qquad j = 1,...,N_l \tag{1f}$$

$$\sum_{i=1}^{N_h} x_{i,j} \geq 2 \qquad j = 1,...,N_l \tag{1g}$$

$$\sum_{j=1}^{N_l} x_{ij} \geq 2 \qquad i = 1,...,N_h \tag{1h}$$

$$r_{jk} \geq \frac{1}{N_f} \sum_{i=1}^{N_f} p_{ik} x_{ij} \quad j = 1,...,N_l,\ k = 1,...,N_p \qquad (1i)$$

$$z_j = \sum_{k=1}^{N_p} r_{jk} \qquad j = 1,...,N_l \qquad (1j)$$

$$z \geq z_j - N_b \qquad j = 1,...,N_l \qquad (1k)$$

$$x_{ij} \in \{0,1\} \qquad i = 1,...,N_f,\ j = 1,...,N_l \qquad (1l)$$

$$s_{ij} \in \{0,1\} \qquad i = 1,...,N_f,\ j = 1,...,N_l \qquad (1m)$$

$$r_{j,k} \in \{0,1\} \qquad \mathrm{j} = 1,...,N_l,\ \mathrm{k} = 1,...,N_p \qquad (1n)$$

$$y_{ij} \in [0,1] \qquad i = 1,...,N_f,\ j = 1,...,N_l \qquad (1o)$$

$$z \geq 0 \qquad (1p)$$

The objective function (1a) minimizes the number of setups and the maximum number of parts in excess of bin spaces on any one line. Constraint (1b) ensures that all 100% of the demand for each family is assigned to a production line. Constraint (1c) makes sure that if any demand for a family is assigned to a line, then that chassis is assigned to that line. Constraint (1d) determines if a setup will be required to produce products needing chassis *i* on line *j*, taking into consideration whether or not that chassis is still on that line from the previous production period. This constraint provides some linkage between the current planning period and the last time period. Constraint (1e) ensures that there will be enough demand assigned to each line to keep the line busy for the whole production period. While it is more common to see a less than or equal to (or "bin-packing") constraint than this greater than or equal to (or "bin-covering") constraint, we included the latter because Dell places a high value on production time and does not want any of it to be lost. Constraint (1f) ensures that we use all lanes on every line. This constraint will never cause additional setups for the current optimization run but does ensure that all lanes are full for the next optimization run, possibly reducing the number

of setups required next shift. Constraint (1g) ensures that each line has at least 2 high runners assigned to it. Constraint (1h) ensures that every high runner chassis is assigned to, at minimum, 2 lines. Constraints (1g) and (1h) may seem unusual. They are included to allow for some balancing of demand across kitting lines without affecting the setups. This ability to somewhat balance demand was key to meet the needs and obtain support from all the team members of the project. These two constraints ensure that each kitting line has multiple chassis with very high demands (high runners) assigned to it, and every high runner is assigned to more than one line, thereby allowing demand for any high runner to be shifted from one of the lines it is assigned to the other without an additional setup. Constraint (1i) ensures that part k is assigned to line j if it is required by any chassis assigned to line j. The right hand side of constraint (1i) will always be less than or equal to one. Constraint (1j) calculates the total number of different types of component required by chassis assigned to the line, for each line. And constraint (1k) sets z to be the largest number of parts that any line is required to have in excess of how many parts that line can hold. Variable definitions are given in constraints (1*l*) to (1p).

The model actually implemented was slightly more complex than the one described above. Other elements were added in to ease the flow of jobs through rest of factory in various ways. None of these additional objectives were allowed to cause an extra setup though.

Theoretically, this optimization model could be hard to solve based on worst case analysis (the simplest model above is NP Hard). In practice, we found that it solves quite quickly using the CPLEX solver from ILOG, called by GAMS, for realistic problem instances.

2.7 LANE ASSIGNMENT AND SEQUENCING (STEP 2)

Once product families have been assigned to lines, the chassis assigned to each line are then assigned to the lanes that hold the chassis pallets according to the following assignment problem. The model solution assigns product families to lanes so as to balance demand across the lanes. It also forces chassis that were sitting in a lane at the end of the previous production period to remain in that lane so they can begin the next period there, preventing the inadvertent addition of setups.

The model variables are:

x_{ik} 1 if family i is assigned to lane k; 0 otherwise

m amount of demand assigned to the lane with the least demand

The parameters are:

d_i demand for family i that is assigned to the line

a_{ik} 1 if chassis i was left in lane k at the end of the previous shift; 0 otherwise

N_f number of product families

N_a number of lanes on each line

The model:

$$\textit{Maximize} \quad m \tag{2a}$$

$$\textit{Subject to} \quad x_{ik} \geq a_{ik} \qquad i = 1,...,N_f, k = 1,...,N_a \tag{2b}$$

$$\sum_{k=1}^{N_a} x_{ik} = 1 \qquad i = 1,...,N_f \tag{2c}$$

$$\sum_{i=1}^{N_f} x_{ik} d_i \geq m \qquad k = 1,...,N_a \tag{2d}$$

$$\sum_{i=1}^{N_f} x_{ik} \geq 1 \qquad k = 1,...,N_a \tag{2e}$$

$$x_{ik} \in \{0,1\} \qquad i = 1,...,N_f, k = 1,...,N_a \tag{2f}$$

The objective function (2a) combined with constraint (2d) maximizes the minimum amount of demand assigned to any lane. Constraint (2b) ensures family i is assigned to lane k if it was left in lane k at the end of last shift. This constraint provides continuity between the current period and the last period. Constraint (2c) guarantees that every chassis gets assigned to exactly one lane. We want to make sure that each chassis is assigned to at least one lane so the parts for that family can be scheduled to be on the line in the next step of the heuristic. On the other hand, we want to assign each chassis to no more than 1 lane because assigning a chassis to more than one would incur an extra setup without adding any benefit. Constraint (2d) makes sure that m is not larger than least amount of demand assigned to any lane. And constraint (2e) ensures that each lane will have at least one chassis assigned to it. The variables are defined in constraint (2f).

Once families have been assigned to lines and lanes, the sequencing decisions are made. Constraint (2b) fixes the sequence for several families who must start the shift on the line, in a specific lane, because they were left there at the end of the previous shift. Once the starting group is determined, a simple heuristic is used which basically prioritizes higher demand families within each lane. For each lane, following the chassis left on the line at the end of the previous shift, the remaining chassis assigned to that lane are sequenced from the chassis with the highest expected demand to the chassis with the lowest expected demand. This simple heuristic is used because of the assumption that setup costs are sequence-independent. The case of sequence-dependant setup costs is investigated in Chapters 3 and 4.

2.8 PRODUCT FAMILY SCHEDULING (STEP 3)

The final step of our scheduling method takes the chassis-to-lane assignments and the sequences and develops a schedule indicating when each family should be added to,

or removed from, the lane it is assigned to. In this step, each lane can be dealt with individually.

First, each lane's available production time for the production period is allotted to each family in proportion to the percentage of the total demand assigned to that lane that that chassis represents. To do this, we first determine the *available production minutes* for the lane by taking take the number of minutes available in a single production shift and subtracting off any planned downtime minutes (due to breaks or scheduled maintenance) and the time it will take to switch between the families in that lane. The available production minutes are then allotted to each family in the lane proportionally according to the proportion of the expected demand for that family compared to the expected demand for all families within the lane.

Once the time allotments are determined for each family in each lane, the allocated time is then laid out on a timeline in each lane, from the start to the end of the shift, inserting the scheduled break times and leaving the required setup time between allotted chassis times. Minor adjustments are then made to ensure setup resources are not overloaded and at least one lane has a chassis in production when a setup is occurring in another lane. Since each kitting line can have chassis for several product families on the line (one in each lane) at any given time, there is flexibility within the schedule to accommodate significant demand variability without requiring a schedule change.

2.9 FULL HEURISTIC PROCEDURE

At the time of the implementation of this solution, it was Dell's practice to run the assignment model in step 1 once every two weeks. Dell found that within this time window, demand generally did not change enough to require a new assignment of families to lines. Before each production shift the schedulers would then feed the family assignment information into what they referred to as the "cycling tool" which performed

steps 2 and 3 of the scheduling algorithm. (The tool was named this because it allows them to cycle through all product families assigned to a line during a single shift.) Once the cycling tool determines the schedule of when each family's parts are on each line, this information is fed into Factory Planner so that Factory Planner can dynamically schedule jobs to be built on each line as new demand information becomes available, while staying within the restriction of only scheduling a job on a line when the family's parts for that job are on the line. In addition, the schedule allows Factory Planner to order parts for customer orders only when their families are scheduled to be on a line, preventing excess part build-up. One other advantage of this schedule is that Factory Planner can schedule the jobs on a FCFS basis within the families that are currently on the line.

The final schedule information is also provided to the materials teams who do the setups and to the scheduling teams who monitor the schedule over the course of the day. Since some new demand information is being revealed over the course of a shift, the schedulers are also provided with the ability to override the schedule and change chassis times for any line, prioritize certain families, or alter which orders are scheduled to be produced, if necessary.

Although the same families stay on each line until the bi-weekly run of the assignment model, both the amount of time a family remains on a line and the sequence will change from shift to shift. The sequence will change because the chassis that begin a production period on a line will not be the same types as the ones on that line at the end of the period. Therefore, the next production period will start with a different set of chassis, and the sequence will be altered. The amount of time that a family spends on a line will change because the amount of demand for that family will change, and because the sequence will change. The cycling heuristic is run every shift using a Visual Basic front-end running on a Microsoft Access database. The results are automatically ported to

Factory Planner which has capabilities for handling these schedules. Once Factory Planner has the product families scheduled, it can then dynamically schedule orders as they arrive, making sure that they are added into the production schedule only when the requires parts are scheduled to be on a kitting line.

We chose to keep step 1 separate from steps 2 and 3, instead of incorporating the line and lane assignments together, for two reasons. First, due to the nature of the lane-assignment model, altering the lane assignments in step 2, or the scheduled times in step 3, will not add any additional setups and the alteration can be incorporated into any shift's production plan relatively easily. On the other hand, a change in the assignment of chassis to lines in step 1 requires additional setups which creates a significant amount of extra work. Second, moderate fluctuations in demand across days can be handled by scheduling or sequencing changes in steps 2 or 3. The chassis-to-line assignments in step 1 need to be changed only when there is a large fluctuation in demand. For these reasons, the line assignment optimization is only run bi-weekly or when a significant shift in demand is seen, whereas the scheduling optimization and heuristics in steps 2 and 3 are used on a once-per-shift basis to schedule the families.

2.10 IMPLEMENTATION AND RESULTS

Our approach was implemented in two phases over a seven month time period. Phase 1 included the implementation of step 1; the assignment model. Phase 2 incorporated steps 2 and 3 (i.e., the "cycling heuristic"). This phased approach was used for a couple reasons. First, the line assignment model was relatively easy to implement because it runs off-line as a planning model and the results need only be updated in Factory Planner bi-weekly. Second, at the beginning of the project, it was projected that the number of product families would increase by only about 25-50 percent and that the line assignment optimization would be sufficient to handle this increase in complexity.

The change in supply chain strategy that dictated all product families be built at all factories was announced just before the implementation of the line assignment model. With the strategy change, it became evident that product family increase would be roughly two-fold and that something beyond the line assignment model would have to be incorporated in order to handle this level of complexity.

The line assignment model was implemented in the fall of 2003. It enabled Dell to seamlessly accommodate its first major increase (about 25 percent) in product variety during the first quarter of 2004. The complete 3-step algorithm was put in use by the factory in the summer of 2004 just before the number of families peaked at more than two-fold the original level. The delay between the first and second phase was a result of the timing of the implementation of the change in supply chain strategy and some complexities encountered integrating the second stage heuristic into the factory processes and the Factory Planner software.

Post implementation, head count did increase a little due to the increase in the number of setups necessitated by the significant increase in product variety. However, the increase was slightly less than what was predicted even after the new algorithm was developed, due to careful planning and the diligence of the managers involved. If Dell had maintained its former production scheduling practices, it projected a decrease of more than 20 percent in hourly production rates due to significant increases in lost production time from increased product family changeovers. This lost production time would have resulted in hundreds of additional hours in overtime per year being required in order to produce the same total volume.

A number of additional indirect benefits occurred as a result of this project. Once the scheduling issue was resolved, it became clear that there were opportunities to improve the processes used to bring raw materials into the factory. Additional metrics

have begun to be tracked and improved procedures have been implemented in this area. Cross-functional teams have also been organized to continually improve how the cycling tool is implemented. These teams are also now able to focus on complimentary processes in other areas within TMC.

Based on our work, Dell was not only able to achieve the two-fold increase in product variety, but it also accomplished an effective production volume increase of over 35 percent. Furthermore, Dell realized a conservative cost avoidance of more than one million dollars annually mostly associated with the additional overtime that would have been required in the absence of our solution to handle the increase in production volume with more product variety.

2.11 CHALLENGES AND LESSONS LEARNED

One challenge we faced was changing the mind-set of employees in regard to how to assign product families to kitting lines. The focus had to change from balancing work across the lines to minimizing the number of setups, which required a significant cultural change. The concern was that, under the new circumstances, balancing workload across lines could result in costly changeovers and slowtimes. This cultural change was accomplished through a project using Dell's Business Process Improvement methodology. A cross functional group from materials, operations, scheduling, engineering, and simulation worked together to agree upon goals of the project and identify causes of reduced production rates and increased material handling. Through this team environment, all key stakeholders were part of choosing and implementing the solution. Requirements to formulate the linear program were taken from this team and initial results were shared with the team before implementation. Measurements of key metrics were taken before and after implementation of the balance from the linear program. These numbers along with the team's experience helped show that planning the

setups could be better for all groups. The process was repeated to refine the balance to more effectively meet the needs of all team members.

Another challenge was making sure that the improvements we made to production planning in the kitting area also worked well with other parts of the factory following kitting such as assembly, packaging and shipping. This challenge proved to be more complicated than it seemed at first. One issue that had to be addressed was the way that orders were routed through the factory. Originally, products were routed through the rest of the factory according to the kitting line from which they came. With the existing routing logic, constraints in other parts of the factory caused significant limitations on the kitting lines to which each family could be assigned. In order to allow enough feasible solutions in our new optimization program to ensure significant benefits, we needed to change this routing logic. Through process analysis we discovered that the computer system that determined the routings could be reprogrammed to route orders through the factory based on which chassis they required instead of from which kitting line they came. This change in the routing program allowed more choices in the assignment of families to kitting lines without hurting the workload balance and setups in the rest of the factory.

One other primary challenge involved coordinating the cycling heuristic with the Factory Planner software. In order to take advantage of Factory Planner's cycling capabilities, an automated solution had to be developed in which order data is extracted from Factor Planner, formatted to create the cyclic schedule, and then exported back to Factor Planner in an acceptable format. While the intermediate automated step took an additional man-month to develop, it reduced the complexity for line schedulers and was much less prone to human error. Additionally, since Factory Planner runs every two

hours, manual creation of the cyclic schedule in this time frame is almost physically impossible.

The benefits of this project did not stop with TMC. Based on the success of this project, TMC's sister facility, also in Austin, TX, has requested that a similar approach be developed and implemented on their kitting lines. This facility produces somewhat lower volume, higher complexity products such as servers and data storage devices. Additionally, the Dell project partner has used the knowledge gained from this work at TMC to help design the production scheduling system in Dell's new facility in Winston Salem, NC that opened in October 2005.

Chapter 3 A Separated Models Scheduling Heuristic

3.1 INTRODUCTION

This chapter presents a new production scheduling heuristic to address a complex production scheduling problem that has arisen in a high-volume, assemble-to-order electronics manufacturer. This manufacturer's factory must balance the challenges of remaining highly flexible and responsive to constantly changing, customized demand while maintaining a high rate of production and quick order turn-around in order to continue to be a front runner in its highly competitive industry. To achieve these goals, the factory was designed according to the lean manufacturing and just-in-time philosophies. Within tight space constraints, the factory must be able to manufacture a continually increasing variety of customized products. This situation has given rise to the need for a new production scheduling method to reduce setup costs while maintaining low order turn-around times.

Although this problem is found in the electronics industry, the contributing conditions are common to many industries. During the 1980s a major trend began in American manufacturing towards lean production facilities, limiting space for inventories, and flexible production lines. In recent years, there has also been a great proliferation of products targeting specific customer desires and an increase in the instances where customers can design the product they order, leading to mass customization. The combination of these two trends results in increasing strain on manufacturing facilities across industries to be even more flexible while producing high volumes of products within tight space constraints.

The scheduling problem addressed in this chapter is the same problem that was presented in Chapter 2, except that in this chapter we explicitly consider the sequence

dependant nature of the setup costs. We present a new three stage procedure for the scheduling of the product families on parallel, identical production lines in order to minimize sequence-dependent setup costs. The performance of this procedure is ultimately compared against the production scheduling heuristic presented in chapter 2.

3.1.1 Problem Description

To fully describe the scheduling problem and the non-traditional nature of the sequence-dependant setup costs we must discuss the factory's products, layout, and materials system. The factory produces a wide variety of products that are grouped into product families to aid in the production planning process. A product family is defined by the chassis type (i.e., case) required by a product. Hence, the terms "chassis" and "product family" will be used interchangeably. Each chassis has a variety of components that can be assembled with it. We will refer to a product family's chassis and set of components collectively as the product family's parts. Each family has some components that are not required by any other family and also has components that are used by various other families. Similarly, some components are limited to one family and other components to several families.

The factory has multiple identical production lines located side-by-side. Figure 2 depicts a rough schematic of one individual production line. As shown, each production line has spaces designed to hold pallets of chassis and spaces designed to hold boxes of components. Each production line can hold L ($\bullet 1$) different types of chassis (one type of chassis per pallet) and C ($\bullet 1$) different types of components (one type of component per box). Each chassis space on a line is called a *lane* and each component space is referred to as a bin. When the line is running, a worker selects one chassis from the L lanes for the next product to be built and sends the chassis to the next worker down the line. The workers down the line select the correct components for that order from the bins and add

them to the chassis. The production computer system keeps track of which chassis types are in which lanes and which components are in which bins, and signals the workers to select the correct chassis and components for each custom order. Therefore, each production line can produce products from any subset of L families, in any sequence, without requiring any changes to the types of chassis that are in the lanes.

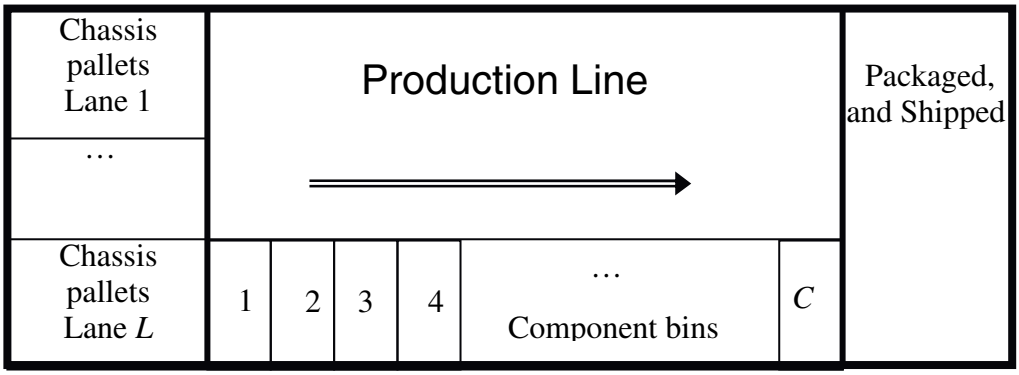

Figure 2: Layout of a single production line with L chassis lanes, and C component bins.

A setup must take place when a line needs to produce products from a family whose chassis type is not in any of the lanes on the line. The setup entails removing the pallet for one chassis type from a lane and placing the pallet of the chassis type required for the next order in that lane. Components must also be changed as well. Those that are not needed for the other chassis types remaining in the other lanes, or for the new chassis type being placed in a lane, must be removed from the bins. Then, the components that are required for the new chassis type that are not already on the line must be placed in the bins. And, finally, the production computer must be updated with the new information on which parts are in which spaces.

The costs for this type of setup are sequence-dependent, but what is not immediately apparent is that these costs depend on the current *subset* of L families on the

line and the new *subset* of *L* families that will be on the line after the setup. More specifically, the total number of parts that will need to be changed during a setup equals the number of parts currently on the line that will no longer be needed by the new subset of *L* families plus the number of parts that are not currently on the line that will be needed by the new subset of *L* families. We use this number of parts that need to be changed during a setup as the setup cost. This is a nontraditional use of the term because sequence-dependant setup costs commonly depend only on the relationship between the single product or family being worked on before the setup and the single product or family being worked on after the setup. In our problem, the component requirements of the product families remaining in the other lanes during the setup also affect the setup cost.

3.1.2 Approach

In the spirit of Pinto and Grossmann (1998), we approach this problem by addressing each of the three major scheduling steps separately: assign tasks to equipment, sequence the tasks, and schedule the exact time that each task will occur within the sequence. In our problem, the "tasks" to be assigned are product families and the "equipment" is the set of production lines. The timing of a task for us is actually the time that the parts for a given product family are placed on the assigned production line.

Due to the non-traditional nature of our setup costs, the second and third steps of the heuristic are the more novel portion of our approach. In the case of traditional sequence-dependant setup costs, the sequencing problem can usually be modeled as a traveling salesman problem (TSP) where each task to be sequenced is represented by a node in the graph, and each arc between two nodes represents the setup cost when switching from performing one task to performing the other task. To obtain a solution where all tasks are completed, each node must be visited. Since our setup costs depend on

the subsets of L families on the line before and after the setup, we need to create a graph for our problem where each node represents one possible subset of L families, with the arcs representing the setups costs of changing between these subsets. With this formulation, it is no longer necessary to visit every node, but rather to just make sure enough nodes are included in the solution to guarantee that each family will be on the line for a sufficient amount of time during the shift. This turns the problem into a traveling salesman subtour problem (TSSP), which is sometimes also referred to as a traveling salesman subset-tour problem (Mittenthal and Noon, 1992) Since the TSSP is an NP-hard problem (it is a special case of a TSP), it is difficult to solve practical problems to optimality. Thus, we develop a greedy randomized adaptive search procedure (GRASP) to find a good, but not necessarily optimal, solution to the sequencing TSSP within a small amount of computational time.

3.1.3 Related Literature

Several streams of literature relate to our problem and solution approach. There is a wealth of literature on production scheduling, as well as quite a bit of work on variations of the TSP and on the GRASP. Here we discuss only the most closely related work.

The majority of academic research involving production scheduling does not explicitly consider the costs of performing setups (Allahverdi et al., 1999). In fact, in Allahverdi et al. (2006) about 300 papers published from 1999 to 2006 that explicitly consider setup times or setup costs were surveyed, but only 5 of these include setup costs, rather than setup time, in the performance criterion (Miller et al. (1999), Baptiste and Le Pape (2005), Agnetis et al. (2004),Vignier et al. (1999), Liu and Chang (2000)). Using the $\alpha/\beta/\gamma$ notation, our problem can be classified as a $P/SC_{sd,b}/TSC$ problem although, as mentioned previously, the nature of our sequence dependant setup costs is non-

traditional. Within the categories of both single-machine and parallel-machine problems with traditional sequence dependant setup times or costs, none of the papers in Allahverdi et al. (2006) use setup costs in their performance criterion.

A related problem does arise in flexible manufacturing factories where individual machines are fitted with tool magazines that hold a variety of tools to work on each job. When two jobs are processed consecutively and all the tools required for both jobs are in the tool magazine, then no setup is required between these jobs. When all tools are available for the first job, but a tool is required for a second job that is not in the magazine, then a setup is required. The problem of minimizing these tool setups has been addressed by Al-Fawzan and Al-Sultan (2002) and by Crama and van de Klumdert (1999), among others. The primary differences between the scheduling problem dealing with tool magazines and our scheduling problem is that in our scheduling problem the jobs can be divided into product families and production volumes are very high. These differences alter the modeling approaches that can be used to solve the problem. Another related research problem appears in Balakrishnan and Vanderbeck (1999). In this paper product families are assigned to parallel assembly lines to minimize setup costs, in a similar manner to our problem. One primary difference is that, in Balakrishnan and Vanderbeck (1999), each line can only assemble products from one family without requiring a setup. Another significant difference is that those authors adopt a setup policy that makes the setups sequence-independent.

Renaud and Boctor (1998) provide a nice overview of particular cases of the TSSP, including the traveling purchaser problem (TPP) and the covering salesman problem. Our problem is more complex than most, though, due to a required constraint modification that requires more than one visit to some subsets of nodes and to the fact that the subsets heavily overlap: each node belongs to L different subsets. The closest

related TSSP to our sequence-dependant sequencing problem is the capacitated TPP for which Laporte et al. (2003) developed a branch-and-cut algorithm using polyhedral theory to derive several classes of valid inequalities. Although the capacitated TPP and our problem turn out to be similar, it is unclear how well the Laporte et al. algorithm would perform on our problem and how well it could be implemented in the real world situation. For more details, please see section 3.3.2.

GRASP was initially developed in Feo and Bard (1989), and Feo et al. (1991). Feo and Resende (1995) provide a detailed description of the common structure and development of the procedure, as well as a review of applications. Since then, GRASP has been implemented in many different works including vehicle routing, machine scheduling, and lot-sizing, to name a few.… Our research demonstrates the usefulness of GRASP for sequencing with unusual sequence dependant setup costs.

The remainder of the chapter is organized in the following manner. Section 2 provides a detailed description of our scheduling procedure including model formulations for each stage. This section also specifies how our scheduling procedure differs from the LMM procedure. In Section 3, we describe the solution methods including details on GRASP for the TSSP sequencing problem. An empirical analysis comparing our procedure with the LMM heuristic is the subject of Section 4. Section 5 contains our conclusions and future research directions.

3.2. MODEL FORMULATIONS

Our heuristic is a "divide and conquer" approach to what would otherwise be an intractable problem. It is structured so that progressively more detailed information is added at each step, while the results of each step become constraints in the next. This is done not only to make the procedure computationally tractable but also to align with how planning is done at the electronics manufacturer.

In the first step, families are assigned to production lines without regard to sequencing. The objectives are to assign families to lines to minimize the number of setups and minimize the maximum number of components assigned to any line since component space is limited. Once the line assignments have been made, the second step focuses on sequencing the product families assigned to each line to minimize setup costs as measured by component changeovers. The third step determines the specific times when product families will be on each line during a production period. In addition to sequencing information, this step relies on information about the planned downtimes due to setups, breaks, and scheduled maintenance. Our heuristic uses a mixed integer program in each of these steps to achieve a low cost schedule.

For several reasons, we have chosen to schedule the times that product families were on each production line, rather than schedule each job. First, this factory produces thousands of products each day using a first-come, first-serve (FCFS) policy instead of order due dates with the goal of shipping orders as soon as possible. Second, setup costs depend only on which family is on the line, not which job is on the line. And third, due to the way order information arrives to the factory, the set of jobs to be completed is not known with certainty when the setups need to be planned. The proposed scheduling heuristic will produce a schedule that will allow production to be in nearly FCFS order, reduce production down-time for setups, and accommodate reasonable differences between actual orders to be produced and the order information known when the setups were planned.

Our approach contains several simplifying assumptions that can be justified by the application context. We use a single-period scheduling horizon, which is one production shift, since this satisfies the planning needs of the electronics manufacturer. Additionally, the models can be solved quite quickly to provide meaningful results within

a practical time frame. It is important to note that even though the models are designed for a single-period, some linkage between periods is accounted for in the assignment model which keeps track of which chassis types were on each line at the end of the previous period (see constraint (3) in the assignment model of Section 2.1). Although demand is not known fully when a schedule must be produced, the average forecasted demand is known with a high degree of certainty within a one-period scheduling horizon. In addition, the flexibility of the production lines and the distribution of high runners across multiple lines absorb moderate shifts in expected demand from one chassis type to another. We also assume constant homogeneous production times. This is an acceptable approximation because production volumes are very high and the process is highly repetitive, so production times are fairly uniform. Ongoing research is being conducted to study the impact of random demand and planning over multiple time periods. Despite all of these assumptions, the heuristic we present here can be used to resolve the company's actual scheduling problem.

3.2.1 Assignment Model

For the assignment step, we use the same mixed integer program that was presented in Chapter 2. To aid understanding of how this model fits into this chapter's three step heuristic, the model is reiterated here. In addition, some notation has been changed slightly to facilitate this interaction of this model with the newly developed models in this chapter and the next. A solution to this model assigns demand and components for all product families to lines, ensures that enough demand is assigned to every line in order to keep it fully utilized, minimizes the number of setups, and minimizes the maximum number of components on any line. Two additional constraints are included in this assignment model for managerial purposes because together they allow for some shifting of demand across lines without requiring a different chassis-line

assignment. These constraints ensure that each production line has chassis with very high demands (high runners) assigned to it, and every high runner is assigned to more than one line. One further constraint enumerated here limits the number of families assigned to a single line to a practical limit established by the company.

The model variables for this mixed-integer program are defined as follows:

x_{ij} 1 if chassis i is assigned to line j; 0 otherwise

y_{ij} percentage of demand for chassis i assigned to line j

s_{ij} 1 if a setup will be required to assemble chassis i on line j; 0 otherwise

r_{jk} 1 if part k is assigned to line j (is required on line j); 0 otherwise

z_j number of parts required by line j

z largest number of parts required on any line

The model parameters are:

a_{ij} 1 if chassis i was on line j at the end of the previous shift; 0 otherwise

p_{ik} 1 if chassis i requires part k; 0 otherwise

d_i demand for chassis i (any demand backlogged from last period plus new demand); without loss of generality, chassis are indexed from that with the highest demand to that with the lowest demand

N_c capacity of each line in units of demand

F number of product families

N_l number of lines

L number of lanes on a line

N_p number of parts

H number of chassis that are high runners (in the spirit of the Pareto principle, the high runners are the relatively small percent of chassis (23% for data in

Section 4) that make up a relatively high percentage of the overall demand (67% for data in Section 4))

S weight indicating the relatively large effort required for an entire family setup vs. the relatively small effort required for the additional individual part swaps needed when more parts are assigned to a single line than will fit on the line at one time; the weight is set to be larger than the maximum number of parts required by a single family

C number of spaces available on a line to hold different types of components.

The model formulation is as follows:

$$\textit{Minimize} \quad S\sum_{i=1}^{F}\sum_{j=1}^{N_l} s_{ij} + z \tag{3a}$$

$$\textit{Subject to} \quad \sum_{j=1}^{N_l} y_{i,j} = 1 \qquad i = 1,...,F \tag{3b}$$

$$x_{ij} \geq y_{ij} \qquad i = 1,...,F,\ j = 1,...,N_l \tag{3c}$$

$$s_{ij} \geq x_{ij} - a_{ij} \qquad i = 1,...,F,\ j = 1,...,N_l \tag{3d}$$

$$\sum_{i=1}^{F} y_{ij} d_i \geq N_c \qquad j = 1,...,N_l \tag{3e}$$

$$\sum_{i=1}^{F} x_{ij} \geq L \qquad j = 1,...,N_l \tag{3f}$$

$$\sum_{i=1}^{H} x_{i,j} \geq 2 \qquad j = 1,...,N_l \tag{3g}$$

$$\sum_{j=1}^{N_l} x_{ij} \geq 2 \qquad i = 1,...,H \tag{3h}$$

$$\sum_{i=1}^{F} x_{ij} \leq L + 5 \qquad j = 1,...,N_l \tag{3i}$$

$$r_{jk} \geq \frac{1}{F}\sum_{i=1}^{F} p_{ik} x_{ij} \qquad j = 1,...,N_l,\ k = 1,...,N_p \tag{3j}$$

$$z_j = \sum_{k=1}^{N_p} r_{jk} \qquad j = 1,...,N_l \qquad (3k)$$

$$z \geq z_j - C \qquad j = 1,...,N_l \qquad (3l)$$

$$x_{ij} \in \{0,1\} \qquad i = 1,...,F,\ j = 1,...,N_l \qquad (3m)$$

$$s_{ij} \in \{0,1\} \qquad i = 1,...,F,\ j = 1,...,N_l \qquad (3n)$$

$$r_{j,k} \in \{0,1\} \qquad \mathrm{j} = 1,...,N_l,\ \mathrm{k} = 1,...,N_p \qquad (3o)$$

$$y_{ij} \in [0,1] \qquad i = 1,...,F,\ j = 1,...,N_l \qquad (3p)$$

$$z \geq 0 \qquad (3q)$$

The objective function (3a) minimizes the number of setups and the maximum number of parts on any one line. The two objectives are balanced by the relative cost parameter *S*. Constraint (3b) ensures that all 100% of the demand for each chassis type is assigned to a production line. Constraint (3c) makes sure that if any demand for a chassis type is assigned to a line, then that chassis is assigned to that line. Constraint (3d) determines if a setup will be required to produce products needing chassis *i* on line *j,* taking into consideration whether or not that chassis is still on that line from the previous production period. This is the main constraint that provides linkage between the current planning period and the previous period. Constraint (3e) ensures that there will be enough demand assigned to each line to keep it busy for the entire production period. While it is more common to see a less than or equal to (or "bin-packing") constraint than this greater than or equal to (or "bin-covering") constraint, we included the latter because the electronics manufacturer places a high value on production time and does not want any of it to be lost. Therefore the manufacturer always maintains a small backlog of demand. Constraint (3f) ensures that we use all lanes on every line. This will never cause additional setups for the current optimization run but does ensure that all lanes are full for the next optimization run, possibly reducing the number of setups required next shift.

Constraint (3g) ensures that each line has at least 2 high runners assigned to it. Constraint (3h) ensures that every high runner chassis is assigned to, at minimum, 2 lines. Constraint (3i) limits the total families assigned to a line to L+5 in order to limit the total number of setups that must be performed on a single line in a single shift. (L+5 was given by the company as the largest number of setups that they desired to do on a single line during a single shift.) Constraint (3j) ensures that component k is assigned to line j if it is required by a chassis assigned to line j. Constraint (3k) calculates the total number of different types of components required by chassis assigned to the line, for each line. Constraint (3l) sets z to be the largest number of components that any line is required to have in excess of the number of components that line can hold. Variable definitions are given in constraints (3m) to (3q).

3.2.2 The Sequencing Model

We formulate the problem of how to sequence the chassis types on each line as a directed graph where each node is the subset of L possible chassis in the lanes at any given time and the arcs represent feasible setups (where one chassis switched out for one other in the setup). For example, if there are five product families (A,B,C,D,E) assigned to an assembly line that has only three lanes (L=3), there would be eleven nodes including a start/end node (see Figure 2.). In general, if there are F product families assigned to an assembly line with L lanes then there will be ${}_{F}C_{L}$ + 1 (i.e., (F choose L) + 1) nodes.

The arcs in the graph between the nodes represent the setups. Between any two nodes that share all but one product family in common there are directed arcs in either direction. Each arc has a cost equal to the number of components that must be changed during the setup. The arcs between pairs of nodes that have more than one different product family are not allowed. While changing more than one product family at a time is

physically possible, it is undesirable. Changing more than one family at a time creates a situation in which two or more product families are unavailable for production on the same line at the same time. This results in a higher risk of line shut-down. The company management was unwilling to assume this level of risk because shutting down a line is very costly. In addition, there are no arcs beginning and ending at the same node, since this would represent no change. There are arcs from each node that represents a subset of product families to the start/end node, each of which have a cost of zero and represent the end of the sequence. The costs on the arcs going from the start/end node to any other node represent the number of parts that will need to be changed when setting up the line to run the families in the first node in the sequence, considering which subset of L families remained on the line at the end of the previous shift.

The goal is to find the least costly path from the start node to the end node that passes through a number of nodes in each subset in proportion to the demand for the corresponding product family. This will ensure that orders from each product family get built during each production shift which was a management requirement in the application setting. An example of a possible solution path is shown in Figure 3.

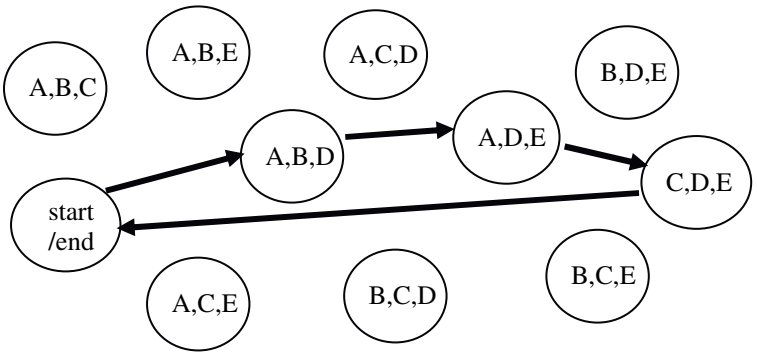

Figure 3. Graph representation of sequencing problem with a possible solution path

The model variables for the sequencing model are defined as follows:

x_{ij} 1 if the product family subset associated with node i immediately precedes the product family subset associated with node j in the solution; 0 otherwise

The models parameters, indices, and sets are:

V set of all nodes; node 0 represents the start and end of the sequence, and all other nodes represent feasible combinations of product families that can be produced on the line simultaneously

N_v total number of nodes (nodes are numbered 0,1,…,N_v-1)

i, j, k indices for nodes

T_n set of all subsets of set $V \setminus \{0\}$ of size n where $N = \left\lfloor \frac{|V \setminus \{0\}|}{2} \right\rfloor$; $n = 2,\ldots,N$

A' subset of all nodes that include product family A

Ω set of all subsets: { A', B', C',… }

w index for subsets A', B', C',…

$d_{A'}$ is the amount of demand that is assigned to the line being sequenced for the family associated with subset A'

L number of lanes on a line

c_{ij} weight for arc (i,j) representing the setup cost for the line setup from families in node i to the families in node j

The IP formulation of the TSSP sequencing model is as follows:

$$\textit{Minimize} \quad \sum_{i=0}^{N_v-1} \sum_{j=0}^{N_v-1} c_{ij} x_{ij} \tag{4a}$$

$$\textit{Subject to} \quad \sum_{i=0}^{N_v-1} x_{ij} = \sum_{k=0}^{N_v-1} x_{jk} \qquad j = 0,\ldots,(N_v\text{-}1) \tag{4b}$$

$$\sum_{k=0}^{N_v-1} x_{0k} = 1 \tag{4c}$$

$$\sum_{i \in T_n} \sum_{j \in T_n, j \neq i} x_{ij} \leq n-1 \qquad n = 2,...,N \qquad \text{(4d)}$$

$$\sum_{i \in w} \sum_{j=0}^{N_v-1} x_{ij} \geq \frac{d_w}{\sum_{f \in \Omega} d_f} \left(\sum_{i=0}^{N_v-1} \sum_{j=0}^{N_v-1} x_{ij} - 1 \right) \qquad \forall w \in \Omega \qquad \text{(4e)}$$

$$x_{ij} \in \{0,1\} \qquad i = 0,...,(N_v\text{-}1),\ j = 0,...,(N_v\text{-}1) \qquad \text{(4f)}$$

The objective function (4a) minimizes the setup costs by minimizing the cost of a tour that begins and ends at node 0 and visits each family at least once. Constraint (4b) ensures that the number of arcs that enter a node equals the number of arcs that leave the node. Constraint (4c) ensures that the path leaves the start/end node only once. Constraint (4d) prevents subtours that do not begin at node 0. Constraint (4e) ensures that the optimal tour visits each subset enough times that the family can be on the production line for enough time to meet the demand. More specifically, the left-hand side counts the number of times the path visits a node in the family's subset which must be at least as large as the right-hand side which multiplies the proportion of the line's demand made up by the family's demand and the number of nodes in the optimal path. If, for example, a line's demand is primarily for orders belonging to one family, all of the nodes in the optimal path must also be in that family's subset, allowing the time scheduling model, described in the next section, to place the parts for that family on the line for the entire shift. And constraint (4f) ensures that the variables are binary.

3.2.3 Time Scheduling Model

Once the line assignments have been made and the sequences of families decided, the exact times that each family's parts are placed on, and removed from, the line needs to be determined. This time scheduling of families does not automatically fall out of the sequencing step, as it often does for more traditional scheduling problems, because the time that each family's parts spend on a line overlaps with the time that several other

families' parts are on the line. Although this step does not alter the setup costs, it does have an important role in how well the factory can respond to customer demand. Orders (jobs) within a family can only be produced when the parts for that family are on the line. In addition, although the line can hold the parts for L families at a time, only one job can be processed by the line at any one time. As a result, the families on the line at one time are actually sharing the available production time of the line, and the actual jobs within those specific families need to be scheduled.

As mentioned previously, the factory does not have due dates for each job but does desire to keep lead-times low for all jobs. Here lead-time is the time from when the customer order is placed to the sales organization to when the product is shipped to the customer. During the production shift, demand information is continually filtered from the sales organization to the factory, and often the demand information does not come to the factory in FCFS order according to the date and time the order was placed. However, the factory dynamically schedules the orders to be built in FCFS orders as much as possible. Therefore, as more demand information becomes visible to the factory during the production shift, jobs are scheduled dynamically on each line in FCFS order. The fact that the demand does not become visible in FCFS order can cause older orders to become visible to the factory after a long delay, which then jump to the front of the queue of jobs to be processed. This situation also adds to the variability in customer demand as it becomes visible to the factory. For a more detailed explanation, see Loveland et al. (2005). However, this dynamic job scheduling application can only schedule a job in a given family to be produced when the parts for that family are scheduled to be on the line. Therefore, a family representing a higher percentage of demand on the line needs to have its parts placed on the line for a higher portion of the production shift's time, so this family's jobs will be able to get enough of the line's available production time to prevent

orders from waiting in the production queue for too long. Because the sequence of subsets of families to be placed on the line has already been determined, this allotment of time to families is adjusted by changing when each node (each subset of three families) is scheduled to be on the line, with a setup duration between each node's end time and the next node's start time. An integer programming model is given for this problem; however, one additional concept is necessary for complete understanding.

In addition to the amount of time that a production line can actually be producing jobs during a shift (the line's available production time), there is also the concept of available lane minutes. This is the amount of time that a family's chassis can spend in one of the lanes on the line. The maximum amount of available lane minutes equals the number of lanes on a line times the line's available production minutes. But, during a setup, one lane becomes unavailable for use by a family, because the family in that lane is being removed and a new family is being added. So, we subtract off the setup duration times the number of setups required on the line during the shift to arrive at the total number of available lane minutes on the line during the production shift.

The variables for the time scheduling linear program are as follows:

T_i start time for node i on the line (T_{N+1} is an extra variable to help make sure the end time for node i equals the end of the production shift, T)

z auxiliary variable to facilitate a max-min objective

The model indices, sets, and parameter are as follows:

i index for nodes in the optimal sequence produced by TSSP model; $i = 1,\ldots,N_n$

j index for families assigned to the line; $j = 1,\ldots,F$

A_j set of all nodes that include family j

B_i set of all families included in node i; $|B_i| = L$ for all i

s index for setups, $s = 1,...,N_n-1$

E_s set of families on the line during setup s; $E_s = B_s \cap B_{s+1}$ and $|E_s| = L - 1$ for all s

e_j number of sets E_j that include family j

T length of the production shift in minutes (excluding any scheduled breaks)

d_j amount of demand for family j assigned to the line

p_j percent of the line's total demand required by family j; $p_j = \dfrac{d_j}{\sum_{k=1}^{F} d_k}$

g amount of time required for a setup

M the total number of lane-minutes in a production shift; i.e., the total number of minutes that family chassis can spend in a lane during the production shift which is calculated as follows: L lanes per line times the total number of minutes in a shift minus the amount of minutes that a lane will be unavailable so $M=LT-(N-1)g$ (i.e., the number of setups needed during the production run times the number of minutes each setup takes)

The model is as follows:

$$\textit{Maximize} \quad z \tag{5a}$$

$$\textit{Subject to} \quad T_{i+1} \geq T_i + g \qquad i = 1,\ldots,N_n \tag{5b}$$

$$\frac{1}{T}\left[\sum_{i \in A_j} (T_{i+1} - T_i - g) + g e_j\right] - p_j \geq z \qquad j = 1,\ldots,F \tag{5c}$$

$$T_1 = 0 \tag{5d}$$

$$T_{N+1} = T + g \tag{5e}$$

$$z \geq 0 \tag{5f}$$

The objective (5a) is to maximize the minimum amount by which any family's allotted percentage of lane-minutes exceeds the percent of the line's total demand required by that family. Constraint (5b) ensures that the proper sequence of nodes is

maintained. Constraint (5c) calculates the amount by which any family's allotted percentage of lane-minutes exceeds the percent of the line's total demand required by that family and ensures that z is less than that amount. Constraint (5d) ensures that the first node starts at the beginning of the shift; i.e., at time 0. Constraint (5e) works with (5b) to ensures that the last node finishes production at the end of the shift; i.e., at time T. Constraint (5f) defines z to be a nonnegative continuous variable and works with (5c) to ensure that each family gets enough time on the line, e.g., if family 1 constitutes 50% of the demand on the line, then that family needs to spend at least 50% of the available lane minutes on the line.

While the concept of scheduling time on the line in proportion to demand is used in both Chapter 2 and in this heuristic, the concept is used in different ways and consequently yields quite different results. In Chapter 2, the sequencing heuristic is applied to each lane; in this procedure, the heuristic is applied to each L-family node. In Chapter 2, the third step strongly determines the sequence of L-family subsets on each line which significantly changes the setup costs for the line; in this procedure, the L-family sequencing on each line is completely determined by the end of the second step (i.e., the sequencing step) in the procedure, so the time-scheduling step does not change the line's setup costs.

3.3 SOLUTION APPROACHES

3.3.1 Solving the Assignment Model

The line assignment problem described in Section 2.1 was programmed into GAMS and solved by calling Cplex 9.0. Practical problems, such as those presented in Section 4, are quite large and difficult to solve to optimality within a time limit that management finds acceptable. Therefore, the computations were halted when the

optimality gap reached 0.5%. (When a stopping criteria of 0.01% was used, the system occasionally ran out of resources before the stopping criteria could be reached.) With this 0.5% stopping criterion, the model always solved within 3 minutes, which is very reasonable for the company involved. Solutions times and company requirements are discussed further in Section 4.

3.3.2 Solving the Sequencing Model

The TSSP single-line sequencing problem described in Section 2.2 is a large-scale optimization problem. The number of subtour elimination constraints grows exponentially with the number of families assigned to the line, quickly overwhelming the capabilities of any standard optimization package. In fact, our largest sequencing problem (eight families on a single line) has 3,249 variables and approximately $4x10^{16}$ constraints. When GAMS (calling Cplex 9.0) was used for a much smaller sequencing problem with only 256 variables and ~ 16,000 constraints, it took nearly one hour to solve.

In order to solve this problem, we developed a GRASP. GRASP was chosen because it is fairly easy to understand, easy to implement, and found to provide good solutions to practical problems with a small amount of computation effort, as shown in Section 4. All three of these factors made this heuristic attractive to the practitioners in this application setting. Alternative approaches were considered. For example, our problem could be reformulated as a capacitated traveling purchaser problem (TPP) and solved using a branch and cut approach described in Laporte et al. (2003). However, their approach was found to be much more complicated and therefore less practical in our application setting. More specifically, Laporte et al. (2003) use polyhedral theory to develop a branch and cut algorithm to solve the capacitated TPP. Our sequencing problem could be reformulated such that, as long as certain parameters are set appropriately, our formulation would match the capacitated TPP formulation. Therefore,

branch-and-cut solution procedure should work for our sequencing problem. Nevertheless, the GRASP was selected because it is easily understood by someone unfamiliar with polyhedral theory and advanced integer programming algorithms, and it is straight forward to incorporate into the overall production scheduling heuristic. Also, the GRASP can be updated quickly to accommodate changes in the business environment and it is not known how easily the Laporte et al. method could be similarly adapted. And finally, it is unclear how well the Laporte et al. algorithm would perform on our problem because it was tested with problem instances that used parameter values quite different from ours.

The GRASP used for this sequencing problem contains the standard solution construction in phase 1 (Feo and Resende, 1995) and two separate solution improvement stages denoted by phase 2 and phase 3. Phase 2 consists of a node elimination routine, and phase 3 consists of a standard neighboring node swapping routine. The node elimination routine in particular is less common because in a standard TSP every node must be visited so none can be eliminated. In a TSSP, though, we only need to visit enough nodes to satisfy the other constraints, so it may be possible to eliminate nodes placed in the path during Phase 1. The details of the GRASP that we developed for this problem are given below. Detailed solution times are given in Section 4, but in all instances this GRASP solved in less than one minute.

Initialization:

1. Create list of all possible nodes for line by building all possible subsets of *L* families from list of all families assigned to line
2. Calculate number of nodes that must be visited for each family based on the percentage of each family's demand relative to the total demand

assigned to the line (calculation shown in constraint (4) in problem formulation in Section 2.2)

3. Set Procedure Parameters
    - K is number of possible nodes with lowest-cost arcs from the current node to them from which to randomly choose when developing a solution
    - M is number of times to iterate through Phase 1 (total number of feasible solutions to generate)
    - N is number of lowest-cost feasible solutions to keep from Phase 1 for possible improvement in Phases 2 and 3
    - Set $P1 = P2 = P3 = 0$, where Pi is a counter for Phase i
    - Set $K = 2$, $M = 5000$ and $N = 2500$

Phase 1 (solution construction):

Step 0: Set "current node" to start/end node

Step 1: For current node, create restricted candidate list by finding K lowest-cost arcs existing the current node and going to a node not already in the solution path

Step 2: Generate a random number between 1 and K to randomly select next node in solution path from the restricted candidate list

Step 3: Update current node to be the selected node from Step 2 and update the number of times each family has been visited for families in selected node. If each family has been visited the required number of times in the solution path, go to Step 4, otherwise go to Step 1.

Step 4: Put start/end node at the end of the solution path to terminate path. Calculate total cost of solution by adding arc cost for arcs used in solution path.

Step 5: If total solution cost is less that the N^{th} lowest solution cost so far then add solution path to the top-solutions-list in the correct position based on solution cost, shifting other solutions as appropriate and discarding the now- N^{th} + 1 lowest cost solution; otherwise discard the new solution.

Step 6: Increment P1 counter.

If *P1* counter = *M*, go to Step 7, otherwise go to step 0.

Phase 2 (node elimination):

Step 7: For solution *P2* of the top *N* solutions saved from Phase 1: check to see if the number of nodes in the solution path equals the theoretical minimum number of nodes.

If yes: go to Step 10. Otherwise proceed to Step 8.

Step 8: For each node in path,

a. check to see if node can be removed and feasibility maintained

b. If yes: calculate cost reduction that would be achieved if node was removed, and keep track node with best cost-reduction value

If this best cost-reduction value is positive, go to Step 9; otherwise, go to Step 10.

Step 9: Remove node with best cost-reduction value. Go to Step 7

Step 10: Increment *P2*

If *P2* = *N*, go to Step 11; otherwise, go to Step 7.

Phase 3 (node swap):

Step 11: For solution *P3* of the top *N* solutions saved from Phase 1: Starting with first node in path after start node, for each node in solution, calculate the cost-reduction value of swapping the node with the node following it in the solution path, keeping track of the node with the best cost-reduction value

If this best cost-reduction value is positive, go to Step 12; otherwise, go to Step 13.

Step 12: Swap the node with best cost-reduction value with the node following it in the solution path. Go to Step 11.

Step 13: Increment *P3*

If *P3* = *N*, go to Step 14; otherwise, go to Step 11.

Step 14: End of GRASP. Output solution path with the lowest total cost. (Often there are multiple lowest-cost paths.)

An interesting feature of this GRASP that seems uncommon is the selection of *M* and *N*. These numbers were chosen based on preliminary analysis that showed that much smaller numbers sometimes returned a higher cost, but much larger numbers did not find a lower cost. Those familiar with GRASP may be surprised at the relatively high value of *N* (the number of Phase 1 solutions kept for Phase 2 and Phase 3 analysis). We discovered that when *N* was a small percentage of *M*, as is often the case when using GRASP, the best solution was often not found. It seems that when *M* is very high, and *N* is very low, the best *N* solutions from Phase 1 already have the theoretical minimum number of nodes, and therefore cannot be improved by node elimination in Phase 2. In some cases, though, the overall best cost solution comes from an initial Phase 1 solution

that had an extra node that was then eliminated in Phase 2. When N is a sufficiently high percentage of M, generally about 50 percent, enough of these "extra node" solutions are retained from Phase 1 for improvement in Phase 2 that the procedure consistently produces the same good solution.

3.3.3 Solving the Time Scheduling Model

The time scheduling model presented in Section 2.3 can be solved using GAMS calling the Cplex solver. However, because the time scheduling model for the new production scheduling heuristic presented in this chapter maintains the node sequence found by the GRASP, the total setup costs will not be changed in this step of our heuristic. The size of this model is also very small so the computation times for time-scheduling each line will not add significantly to the overall heuristic solution time. Therefore, there is no need to discuss the solution of this model.

3.3.4 Solving the LMM Heuristic

The lane assignment model used in the LMM production scheduling heuristic was solved to optimality using GAMS calling the Cplex 9.0 solver. The time scheduling heuristic was then used to determine the sequence of families within each lane on the line. These calculations were done on a spreadsheet. Once the lanes were sequenced, the resulting overall line sequence could be "read" as a sequence of 3 family sets separated by setups. These 3-family sets then become nodes in a graph identical to the one presented in Figure 2, with identical part setup costs represented by the arcs. Then the total part setup cost was calculated using the arc costs associated with the solution path. In regard to the solution times for the LMM heuristic, the assignment model is the same as for the new heuristic, so the solve times are the same. The rest of the LMM heuristic

takes slightly longer than the solution times reported in Section 4 for the new heuristic, but the difference is not significant.

3.4. EMPIRICAL ANALYSIS

We benchmarked the new heuristic against the LMM heuristic since the LMM heuristic is the only other known scheduling method for this problem, and because this is the comparison that the company was interested in. We used real data generated over 20+ production periods from the electronics manufacturer. These data included demands for multiple product families and their part requirements on parallel, identical production lines. Using Pareto's rule, we were able to determine the number of high runners in the data for the assignment problem.

We constructed 5 scenarios using 5 different sets of demand information. Each scenario came from a randomly selected production period. To the demand data selected from a given production period, we added average demand over all the production periods times a company-specific multiplier to get the final demand data used in the models. The average demand times the multiplier represents an average level of backlog that the electronics manufacturer tries to maintain as a target in its operation.

We ran the assignment model on each scenario's demand data to get the assignments of families to the lines. Then we ran family and demand data produced by the assignment model through LMM lane assignment sequencing, and time scheduling steps. Similarly, we ran the same data through our new sequencing heuristic.

The percentage reductions in total setup costs ("% reduction") for each scenario of our new heuristic over the LMM heuristic are shown in Table 1. Examining the results by scenario shows an overall reduction in setup costs that averages 17.72%, with standard deviation 2.42% and ranges from 14.35% to 20.96%.

Table 1: Percentage Reduction of Total Setup Costs for Each Scenario of the New Heuristic over the LMM Heuristic

Scenario	% reduction	Scenario assign & sequence computation time (min)
1	18.30	3
2	20.96	3
3	14.35	2
4	16.74	2
5	18.24	1

Table 2 provides a more detailed view of the data by the line assignments made across all five scenarios combined (a total of 27 lines were scheduled across all five scenarios). The "# of families" column indicates the number of families assigned to a given line. Analyzing each line separately we find an average reduction per line of 15.63%, with a standard deviation of 12.98%. The best individual line reduction was 48.59% and, in the worst case, the new heuristic provided no reduction. We performed a t-test on the data to test the null hypothesis that the percentage reduction equals zero versus the alternative hypothesis that the percentage reduction is greater than zero. The null hypothesis was rejected at the one percent level strongly indicating a statistically significant reduction in total set-up costs.

Table 2: Percentage Reduction in Setup Costs of the New Heuristic over the LMM Heuristic by Production Line.

Line	# of families	% reduction	GRASP computation time (sec)	Line	# of families	% reduction	GRASP computation time (sec)
1	4	8.48	1	15	8	14.56	6
2	4	5.42	1	16	6	7.5	3
3	6	1.47	4	17	4	12.72	1
4	6	32.28	3	18	5	11.72	2
5	6	37.71	3	19	7	18.13	5
6	6	8.01	3	20	5	23.85	2
7	7	27.81	5	21	5	24.5	2
8	5	6.59	2	22	5	0	2
9	4	36.58	1	23	4	0	1
10	7	12.5	5	24	5	0	2
11	6	13.99	3	25	6	48.59	3
12	7	19.16	5	26	8	26.85	2
13	7	17.28	5	27	4	4.08	1
14	4	2.14	1				

A summary of the data in Table 2 is given in Table 3 and provides a different view. The “Count” column in Table 3 is simply the number of lines in Table 2 to which a certain number of families was assigned. When we look at the data by the number of families assigned to the line, we see that the average improvement for 4 and 5 family lines hovers around 10% whereas the average improvement seen for 6, 7, and 8 family lines is around 20%. This shows that as complexity increases, the benefit of using the new production scheduling heuristic increases significantly. However, best case results shown in the forth column in Table 3 indicate that significant improvement in the solution is possible for any number of families. Testing the null hypothesis that the percentage reduction equals zero versus the alternative hypothesis that the percentage reduction is greater than zero by family, the null was rejected at the five percent level for

4 and 5 family assignments and at the one percent level for 6 and 7 family assignments. This is additional evidence that the new production scheduling heuristic provides more benefit as complexity increases. With an extremely lower power test based on only two data points for the eight family case, we were unable to reject the null hypothesis.

Table 3: Percentage Reduction in Setup Costs of the New Heuristic over the LMM Heuristic by Number of Families.

# of families	Count	% reduction average	% reduction best case	p-value
4	7	9.92	36.57895	0.040
5	6	11.11	24.49664	0.028
6	7	21.36	48.59438	0.010
7	5	18.98	27.80612	0.001
8	2	20.71	26.85185	0.184

As the results show, the new heuristic consistently outperforms the LMM heuristic. This may seem to be an obvious result since the new heuristic takes into consideration more details of the real problem, but we could only truly guarantee that the new heuristic would perform as well or better if we could solve the sequence-dependant sequencing model to optimality. Since we are using a heuristic, better performance is not guaranteed. With this in mind, we can say that there are no specific systematic conditions under which we would expect the new heuristic to perform worse that the LMM heuristic. The LMM heuristic can only provide better results under the rare circumstance that it happens to come up with an optimal sequence-dependant solution that the GRASP is unable to find. We did not have this happen in any situation we examined.

The overall computation times presented in Table 1 show that this heuristic can be solved within minutes. The company required that the heuristic be able to produce a schedule within 30 minutes, so the heuristic is fast enough. Since the results presented

here used actual company data, no additional assumptions are required for the company to use this heuristic.

3.5 CONCLUSION AND FUTURE RESEARCH

In this chapter we have introduced a new production scheduling heuristic for a complex scheduling problem encountered in practice involving setup costs. To address the most difficult part of the solution procedure, we developed a GRASP. The performance of the new heuristic was shown to be superior to a previous heuristic developed and implemented by Loveland et al. (2005). The new procedure is practical, easy to understand and implement, and adaptable to changing business environments.

There are several possible directions of future research for this problem. One direction is to truly optimize the setup costs for the problem outlined in this chapter by combining the interdependent assignment and sequencing models. This model will resemble a multiple vehicle routing problem in that the lines resemble vehicles and the "routing" of these "vehicles" though the nodes in the graph resembles the sequencing of the nodes on the lines. But this problem will also retain the interesting features of the TSSP presented in this chapter: that only a portion of the nodes in the graph need to be visited, and the subsets of nodes representing families overlaps heavily. This optimization model will be quite complex but could also be approached using GRASP.

Another possible direction is a simulation analysis of different heuristics to solve this problem. The purpose of this analysis is multifold. First, the analysis can be designed to examine the performance of the two heuristics considered in this chapter plus, if developed, the additional scheduling heuristic using the combined (assignment and sequencing) optimization model mentioned above. The simulation analysis could permit the evaluation of the performance of the heuristics based not only on the setup costs, but also on the order delivery performance (order lead-times) using the production schedules

produced by each heuristic. The simulation could also allow for the testing of robustness of certain simplifications such as the use of demand averages and the assumption of identical, deterministic production times. Simulation could also be used as a meta-heuristic procedure where the simulation provides feedback to the mathematical programming models in an iterative manner and becomes part of the optimization procedure.

One more possible direction of future research is to further analyze the line assignment problem using stochastic programming. This is a beneficial research direction because, due to the timing of decisions in practice, the line assignment is done in the face of more uncertainty than any other step in the process. This problem could be investigated using a single period simple recourse model or a multiple-period model with recourse. The latter model would be the first multi-period model considered for the specific problem addressed in this chapter.

Chapter 4 A Combined Model Scheduling Heuristic

4.1 INTRODUCTION

In this chapter we revisit the sequence-dependant production scheduling problem presented in chapter 3. In the method used in chapter 3 we modeled the assignment and sequencing steps separately due to the complicated nature of the sequencing problem. We were then able to develop a combined optimization model that both assigns and sequences the product families on each line. Whereas the mixed-integer program used for the assignment problem and the integer program used for the sequencing problem were both linear, the combined model is a non-linear mixed-integer program.

This problem of combining the assignment and sequencing steps is unusual and not readily found in existing literature. The most closely related problem is the multiple vehicle routing problem (MVRP). In the MVRP, multiple vehicles must be routed from a depot through numerous locations to either pickup or deliver a product. The MVRP has been used for vehicle routing applications, and also for combined production planning and distribution applications (e.g., Boudia et al., 2006). The main difference between our problem and the MVRP is that we do not have to visit every node. Therefore, our problem could be referred to as a multiple vehicle routing subset-tour problem (MVRSP). A 2002 book on vehicle routing problems (VRPs), edited by Toth and Vigo, does not present any VRP variations that do not require all nodes to be visited, except for the case where penalties are assessed for any node that is not visited, which resembles a prize-collecting TSP rather than a subset-tour TSP. In older book, Jaillet and Odoni (1988) present one variation of the VRP, the probabilistic VRP, that does not require all nodes to be visited and does not penalize non-visited nodes. Rather, each node has a probability of requiring a visit. Once again, the nodes are not separated a-priori into subsets so this is

not a subset-tour type problem either. In addition, the authors do not re-solve the problem each time the node-visit requirements are determined. The combined model presented in this chapter is both NP-hard and very large for real problem instances. Therefore, the model cannot be solved to optimality and we must use a heuristic so solve it. We chose to develop a greedy randomized adaptive search procedure (GRASP) due the fact that the procedure was successful for the separated models approach. Aditionally, the straight-forward nature of the GRASP allows it to be easily modified to accommodate changes in the business environment and easily explained to company production schedulers. In Boudia et al. the authors also chose to develop a GRASP to solve the MVRP with split-pickups, but, of course, due to the differences between their problem and ours, their GRASP is not applicable to our problem.

The interesting question that arises with the introduction of this combined model is whether or not the resulting schedules will out-perform the schedules produced by the separated models method in terms of total factory-wide setup costs. The combined model will theoretically provide solutions that dominate the separated models approach when solved to optimality. But since we had to use a heuristic solution method that may or may not achieve optimality, we could not be sure of the outcome until we tested the model.

In Section 4.2 we present the combined assignment and sequencing model that, when followed by the time-scheduling procedure using the model presented in Section 3.2.3, comprises a new scheduling method. In Section 4.3 we discuss the GRASP we developed to solve this combined model. In Section 4.4 we compare the setup costs of using this new scheduling method against the setup costs of using the separated models method of scheduling presented in Chapter 3. And in Section 4.5 we discuss our conclusions.

4.2 COMBINED MODEL

Indices

i index for families; without loss of generality, families are ordered in decreasing order of demand levels

j index for lines

a, b and c indices for nodes

Variables

x_{ij} 1 if product family i is assigned to line j; 0 otherwise

y_{ij} percent of demand for product family i assigned to line j

z_{jab} 1 if arc from node a to node b is used on the sequencing path for line j; 0 otherwise

Parameters

d_i demand for product family i (backlogged demand plus new demand)

N_c capacity of each line j in units of demand

N_f number of product families

N_l number of lines

N_a number of lanes on a single line

N_v number of nodes

N_h number of chassis that are high runners

V set of all nodes

O_j start/end node for line j

S_i set of all nodes that include product family i

c_{jab} cost of using arc z_{jab}

T_n set of all subsets of set $V \setminus \{O_j\}$ of size n, for $n = 2, 3 \ldots, N$; where $N = \left\lfloor \frac{card(V \setminus \{O_j\})}{2} \right\rfloor$

Ω set of all family subsets $\{S_{i_1}, S_{i_2}, ...\}$

Model

$$\textit{Minimize} \quad \sum_{j=1}^{N_l}\sum_{a=1}^{N_v}\sum_{b=1}^{N_v} c_{jab} z_{jab} \tag{6a}$$

$$\textit{Subject to} \quad \sum_{j=1}^{N_l} y_{ij} = 1 \qquad i=1,\ldots,N_f \tag{6b}$$

$$x_{ij} \geq y_{ij} \qquad i=1,\ldots,N_f;\ j=1,\ldots,N_l \tag{6c}$$

$$\sum_{i=1}^{N_f} y_{ij} d_i \geq N_c \qquad j=1,\ldots,N_l \tag{6d}$$

$$\sum_{i=1}^{N_f} x_{ij} \geq N_a \qquad j=1,\ldots,N_l \tag{6e}$$

$$\sum_{i=1}^{N_h} x_{ij} \geq 2 \qquad j=1,\ldots,N_l \tag{6f}$$

$$\sum_{j=1}^{N_l} x_{ij} \geq 2 \qquad i=1,\ldots,N_h \tag{6g}$$

$$\sum_{i=1}^{N_f} x_{ij} \leq N_a + 5 \qquad j=1,\ldots,N_l \tag{6h}$$

$$\sum_{a=1}^{N_v} z_{jab} = \sum_{c=1}^{N_v} z_{jbc} \qquad j=1,\ldots,N_l;\ b=1,\ldots,N_v \tag{6i}$$

$$\sum_{b=1}^{N_v} z_{jO_j b} = 1 \qquad j=1,\ldots,N_l \tag{6j}$$

$$\sum_{a\in T_n}\sum_{b\in T_n} z_{jab} \leq |T_n| - 1 \qquad j=1,\ldots,N_l;\ n=2,\ldots,N \tag{6k}$$

$$\sum_{a\in S_i}\sum_{b=1}^{N_v} z_{jab} \geq \frac{d_{S_i} y_{ij}}{\sum_{k=1}^{N_f} d_{S_k} y_{kj}} \left(\sum_{a=1}^{N_v}\sum_{b=1}^{N_v} z_{jab} - 1\right)$$

$$i=1,\ldots,N_f;\ j=1,\ldots,N_l \tag{6l}$$

$$\sum_{a\in S_i}\sum_{b=1}^{N_v} z_{jab} \leq |V| x_{ij} \qquad i=1,\ldots,N_f;\ j=1,\ldots,N_l \tag{6m}$$

$$x_{ij} \in \{0,1\} \qquad i=1,\ldots,N_f;\ j=1,\ldots,N_l \tag{6n}$$

$$y_{ij} \in [0,1] \qquad i=1,\ldots,N_f;\ j=1,\ldots,N_l \tag{6o}$$

$$z_{jab} \in \{0,1\} \qquad j=1,\ldots,N_l;\ a=1,\ldots,N_v;\ b=1,\ldots,N_v \tag{6p}$$

The objective function (6a) minimizes the total setup costs by minimizing the sum of all the arc costs for each line's subtour. Constraint (6b) ensures that all 100% of the demand for each chassis type is assigned to a production line. Constraint (6c) makes sure that if any demand for a chassis type is assigned to a line, then that chassis is assigned to that line. Constraint (6d) ensures that there will be enough demand assigned to each line to keep it busy for the entire production period. Constraint (6e) ensures that we use all lanes on every line. Constraint (6f) ensures that each line has at least 2 high runners assigned to it. Constraint (6g) ensures that every high runner chassis is assigned to, at minimum, 2 lines. Constraint (6h) limits the total families assigned to a line to N_a+5 in order to limit the total number of setups that must be performed on a single line in a single shift. (L+5 was given by the company as the largest number of setups that they desired to do on a single line during a single shift.) Constraint (6i) ensures that the number of arcs that enter a node equals the number of arcs that leave the node. Constraint (6j) ensures that each line's path leaves the start node for that line only once. Constraint (6k) prevents subtours that do not begin with a line's start node. Constraint (6l) ensures that the optimal tour on each line visits each family's subset enough times that the family can be on the production line for enough time to meet the demand. More specifically, for each family-line combination, the left-hand side counts the number of times the line's path visits a node in the family's subset which must be at least as large as the right-hand side which multiplies the proportion of the line's demand made up by that family's demand and the number of nodes in the optimal path. And constraint (6m) ensures that if a family is not on a line, then no arcs from nodes including that family are used on that line's path. Variable definitions are given in constraints (6n) to (6p).

4.3 SOLVING THE COMBINED MODEL

The model presented above is NP-hard since it is a special case of a TSP. In addition, this model is very large for the real problem size. For the problem size we need to solve, there are over 6 million variables and an incredibly large number of constraints. (For the single-line sequencing problem there were only as many as 3,249 variables which gave us approximately $4x10^{16}$ constraints.) Therefore we needed to use a heuristic to solve the problem. Since the GRASP proved useful for the single-line sequencing problem, we decided to use this method again. For this combined model though, the GRASP proved much more difficult to develop. Below we present the GRASP heuristic that we developed, and then discuss some detailed aspects of it in the following subsection.

4.3.1 GRASP Discussion

The GRASP that we developed to solve the combined model is presented below. As is typical of GRASP, it consists of two phases. The first phase generated feasible solutions and the second phase improves solutions. This GRASP is quite a bit more complex than its counterpart presented in Chapter 3, however. In addition to having 4 times as many constraints to enforce, the combined model includes one constraint that cannot be maintained during the construction of the solution but, rather, must be checked after the solution is constructed: the non-linear family node-visits constraint (6*l*). In addition, the constraint (6d), which concerns minimum required demand levels on each line cannot be ensured during solution construction due to the random method of selecting additional nodes during solution construction.

As a result, Phase 1 of this GRASP consists of two main sections. The first section, including steps 1 through 6, constructs a solution using a greedy randomized method of building up the solution paths. The second section consisting of steps 7

through 11, checks the solution feasibility and repairs the solution if it is not feasible but likely fixable. Phase 2, the solution improvement phase, then uses 2 main procedures to improve the solutions. The first procedure, Phase 2a, lowers the solution cost by removing nodes (and therefore costly arcs) that are not required to maintain solution feasibility. Phase 2b then further reduces the solution costs by swapping neighboring nodes wherever feasible and cost effective.

4.3.2 GRASP Pseudo-code

Phase 0: Initialization

1. Input all data values from text files
2. Set all program parameter values
   - *K*=6; number of lowest cost arcs from which one arc will be randomly selected during each iteration of a single solution's construction during Phase 1
   - *FS*=2000; number feasible solutions to generate in Phase 1
   - *TS*=1000; number of lowest-cost feasible solutions to carry over from Phase 1 to Phase 2
   - *NNV*=10; number of family node-visits that are still needed to consider the solution potentially fixable and to attempt to add nodes to achieve feasibility
3. Initialize Phase 1 counter, *P*1, to zero

Phase 1: Develop feasible solutions

1. Initialize all solution variables
2. Randomly select one initial arc out of each start node and place the node at the end of each arc at the beginning of each line's respective path
3. Find the *K* eligible lowest cost arcs coming out of the current nodes on each path (*K* total arcs, not *K* arcs for each path)

   An arc is not eligible if:

   a. more than one family is different between the node at the beginning of the arc and the node at the end of the arc OR
   b. the node at the end of the arc is already on that line's path OR
   c. the line to whose path the arc would be added does not already have the new family that would need to be assigned to the line AND
      i. the line to whose path the arc would be added already has maximum number of families assigned to it OR

ii. the line to whose path the arc would be added already has the minimum number of families assigned to it AND at least one other line does not OR
iii. if new family is a high runner AND the line to whose path the arc would be added already has the minimum number of high runner families assigned to it AND at least one other line does not OR
iv. if new family is a low runner AND new family has already been assigned to another line
v. if new family is a high runner AND new family has already been assigned to 2 other lines

4. Randomly select one of the *K* eligible lowest cost arcs and add the node at the end of the arc to the end of the line's path that contains the node at the beginning of the arc
5. Check solution-ending conditions
   Solution is complete if:
   a. All low runner families have each been assigned to one line AND
   b. All high runner families have each been assigned to 2 lines

If solution is complete, go to step 6; else go to step 2 (continue building solution).

6. Assign demand to each line
   a. Assign the full amount of each low runner's demand to the line to which the low runner is assigned
   b. Starting at the line with the lowest total demand, assign enough demand from each of the line's high runner families to bring the line's demand up to desired levels (level of demand if all lines had equal amounts of demand)
   c. Follow each line's high runner from the previous step to its other line and assign the rest of its demand there
   d. Repeat step 6c until you reach a line where all high runner families' demand have been assigned
   e. If any line does not yet have high runner demand assigned to it, go to step 6b; else go to step 7
7. Determine solution feasibility regarding demand assigned to lines
   Solution is feasible if each line has more demand than capacity

If solution is feasible, go to step 9; else go to step 8

8. Attempt to fix demand spread
   a. Find the line with the lowest demand, shift demand for this line's high runners to it from the other lines with these high runners
   b. If another line has too little demand, repeat step 8a (limit placed on repetitions to avoid infinite loop)
   c. If demand assignment is now feasible, go to step 9; else discard solution and go to step 1
9. Determine solution feasibility regarding number of node-visits for each family

Solution is feasible if path visits enough nodes for each family on each line to which the family is assigned

If solution is feasible, go to step 12; else go to step 10.

10. Count number of family node-visits that would be required to achieve feasibility

If number of required node-visits is no more than *NNV* go to step 11; else discard solution and go to step 1

11. Attempt to add enough nodes to achieve feasibility
    a. Find a line without enough node visits
    b. Find the eligible arc from the end of this line's path with the least added cost per added needed node-visit
       An arc is not eligible if:
       i. more than one family is different between the node at the beginning of the arc and the node at the end of the arc OR
       ii. the node at the end of the arc is already on that line's path OR
       iii. the node at the end of the arc has any families that are not already assigned to the line
    c. If a lowest cost arc was found, add node at end of least-cost arc to the line's path, recalculate overall required node-visits base on one more node in path, and check for feasibility; else discard solution and go to step 1
    d. If line's family node-visits not feasible yet, return to step 11.b
    e. If line now feasible, but another line is not, return to 11a
    f. If solution is now feasible, continue; else discard solution and go to step 1
12. Compare solution cost to the most costly of the *TS* best solutions.
    If solution cost is less than the most costly of the *TS* best solutions go to step 13; else discard solution and go to step 1
13. Save solution and discard the most costly of the *TS* best solutions and increment *P*1
14. If *P*1<*FS* then go to step 1; else continue on to Phase 2a

Phase 2a: Solution Improvement by Node Elimination

For each top solution retained from Phase 1:
For each line: eliminate unnecessary nodes in order from best cost savings to least cost savings until no more nodes can be deleted (a node is unnecessary if it can be removed without violating any family's node-visit requirement)

Phase 2b: Solution Improvement by Neighboring Node Swaps

1. For each top solution retained from Phase 1:
   For each line: carry out feasible neighboring node swaps in order from best cost savings swap to least cost savings swap until no more cost reductions are possible (neighboring node swaps are feasible if no arc introduced by the

swap has more than one different family between the node at the beginning of the arc and the node at the end of the arc)

2. Find lowest cost solution of all top solutions, and return one top solution with the lowest cost

4.3.3 GRASP Parameters and Other Notes

We selected our parameter values based on developing a reasonable similarity between the current GRASP (prefaced with "c" for combined model) and the GRASP presented in section 3.3.2 (prefaced with "SL" for single-line), as well as achieving reasonable solution times. In the SLGRASP we randomly selected from the two lowest cost arcs when building a solution. In the cGRASP it seemed reasonable to the select from a number of arcs equal to two times the number of lines. This created unreasonably long solution times, so we set *K* equal to 6, the number of lines in this example. We selected *FS*=2000 because we seemed to be able to get 2000 feasible solutions within a good amount of time. We then set *TS* equal to one half of *FS* just as *N* equaled one half of *M* in the SLGRASP. *NNV* is a new parameter in cGRASP, and we set this such that the number of solutions that were rejected due to requiring more visits than *NNV* would approximately equal to the number of solutions rejected due to not being able to fix the solution to satisfy the violated node-visit constraint.

In the formulation of the cGRASP, we did effectively tighten one constraint and add another. We required that each low runner family be assigned to exactly one line, and each high runner family be assigned to exactly two lines. This essentially turns the inequality in constraint (6g) into an equal sign, which affects only the high runner families, and forces the y_{ij}'s to be binary for each low runner. This is the optimal number of family assignments in the assignment problem from section --- that was incorporated in the combined model. To prove that, within the context of the combined model, these constraint changes do not eliminate any possible solutions is a little complicated.

Logically, if any family is assigned to more lines than these restricted constraints allow, they will cause additional family setups. The only way this could be optimal is if, somehow, this additional setup on another line would actually allow an overall lower total setup cost for that line. We are working on the formal proof for this assertion.

4.3.4 GRASP Constraint Enforcement

The GRASP outlined above ensures that all the constraints in the model are maintained. Specifically, constraints (6b) and (6c) are enforced during steps 6. Constraint (6d) is enforced by steps 7 and 8. Constraint (6e) is guaranteed by the condition in 3.c.ii. Constraint (6f) is enforced step 3.c.iii. Constraint (6g) is enforced by 3.c.v and the fact that there are at least as many high runners as lines. Constraint (6h) feasibility in ensured by step 3.c.i. Constraints (6i) and (6j) are enforced by the fact that, in steps 2, 3, and 11.b, we only look for feasible arcs coming out of nodes that are at the end of the current path on a line. Constraint (6k) is ensured in step 3b. Constraint (6*l*) is enforced in step 11. And Constraint (6m) is kept feasible during steps 3 and 11.b.iii.

4.4 EMPIRICAL RESULTS

To test the performance of the schedules produced using this combined model and GRASP solution method we compared the total setup costs resulting from using this method to the setup costs resulting from using the separated models method presented in Chapter 3. It needs to be noted here, however, that one change had to be made to the SLGRASP in order to make the two methods completely comparable. More specifically, the two methods, as presented, use two different procedures for selecting the initial node in each line's sequence. The SLGRASP chooses the initial node for each line by randomly selecting from K=2 lowest-cost, eligible arcs from the start node, whereas the cGRASP chooses the initial node by randomly selecting from *all* eligible arcs out of the

start node. (Allowing the procedure to select from all eligible arcs simulates a random selection of the starting condition of each production line. In other words it, in effect, randomly selects which families were left on each line at the end of the previous shift.) To fix this discrepancy, we modified the SLGRASP to randomly select the initial node for each line from all eligible nodes.

For this comparison, we used realistic demand data for 6 separate production period scenarios and recorded the resulting factory-wide setup costs. The percent reduction results and heuristic run-times are given in Table 4.

Table 4: Percentage Reduction of Total Setup Costs for Each Scenario of the Combined Model Heuristic over the Separated Models Heuristic

Scenario	% Improvement	Solve times (min)
1	–46.32	14
2	–3.86	14
3	–56.89	8
4	–59.84	29
5	–27.38	64
6	–34.96	8

The results show an average reduction in setup costs of –46.32 percent, with a range of –59.84 to –27.38 percent, meaning that the new scheduling method performed worse than existing procedure. We performed a t-test on the data to test the null hypothesis that the percentage reduction equals zero versus the alternative hypothesis that the percentage reduction is less than zero. The null hypothesis was rejected at the one percent level, despite the low number of samples, strongly indicating a statistically significant degradation in total set-up costs. The solve times are all under the required 30 minutes except with scenario 5 which took 64 minutes to run. This appears to be because one low runner actually has very high demand in that scenario, making it difficult to find solutions with feasible demand spreads (to keep the lines running). This caused many

potential solutions to be rejected as infeasible due to at least one line having too little demand assigned to it, and forced the program to loop through Phase 1 many more times.

4.5 CONCLUSIONS

Overall, the results show that the combined model scheduling method does not outperform the separated models scheduling method. There are three factors that we know contribute to the poor performance of the new combined models scheduling method. The first is that the combined model is very large and difficult to solve. As compared to the largest single-line sequencing problem (solved with the SLGRASP), the combined model problem has over 45 times as many nodes (2606 nodes as compared to 57 nodes). And due to this, the time required to generate a feasible solution during Phase 1 of the cGRASP is much larger than for SLGRASP. In fact, in an attempt to achieve solve times of less than 30 minutes, we had to reduce the number of feasible solutions generated in Phase 1 of the cGRASP to 2000 from the 5000 generated in Phase 1 of the SLGRASP.

The second factor contributing to the poor performance of the cGRASP also relates to the number of nodes. In the combined model problem there are 2600 family-nodes from which start nodes can be chosen and the cGRASP only finds 2000 feasible solutions. Therefore it is not possible for each node to appear in at least one feasible solution. Therefore Phase 1 of our cGRASP is only able to generate a very small percentage of feasible solutions. In addition, since the initial node selection is completely random, many of the generated feasible solutions may have family-assignments that are so poor (in terms of the best setup costs that could be achieved) that the factory would never put those family groupings on the lines. In contrast, the single line sequencing problem only has 56 family-nodes and the SLGRASP generates 5000 solutions, so each family-node is likely to appear as the initial node in many feasible solutions. And, since

the set of families on the single line has been chosen by the assignment problem to not have a lot of differing part requirements, the randomly chosen initial family grouping on the line is likely to be a reasonable one in reality.

And the third contributing factor, related to the second factor, is that the Phase 2 solution improvement procedures in the cGRASP do not improve the family-to-line assignments. Rather, the improvement procedures only improve the sequences for each line. Therefore, if a bad family-to-line assignment is generated in Phase 1, it is not changed in Phase 2. Although Phase 2 of the SLGRASP also only improves sequencing, in the separated models scheduling method the family-to-line assignments are generated in a more reliable way (through the use of the assignment model).

In addition to the poor performance of the combined models scheduling method in terms of setup costs, the high solve times are also a cause for concern. Although the one exceptionally high solve time seen in scenario 5 may be explained by our method of demand data generation, all of the scenarios took quite a while to solve. This is primarily due to the solution generation in Phase 1. As mentioned previously, with the combined model we can no longer maintain solution feasibility throughout the solution construction process and must therefore check for feasibility, and either fix or discard solutions after they are generated. The run records show that the cGRASP generally discarded about four solutions for every feasible solution it found. Further research may be able to reduce this waste.

Chapter 5 Conclusions and Future Research

In this dissertation we presented three different methods for solving the complex scheduling problem found at Dell's TMC factory. The first solution method, which was implemented in Dell's TMC factory, assumed that the setup costs were sequence independent providing a good first-pass solution to the imminent scheduling problem. The second method explicitly considered the sequence-dependant nature of the setup costs. Although this method modeled the assignment and sequencing aspects of the problem separately (both of which affect sequence-dependant setup costs) and the sequencing problem could not be solved to optimality, the second method significantly outperformed the first method. The third scheduling method combined the assignment and sequencing aspects of the problem into a single optimization model which, when solved to optimality, would produce setup costs at least as good as, or better than, the second method. However, this problem also could not be solved to optimality, and the problem was so large that it resulted in significantly larger setup costs.

Although the first scheduling method was the only one implemented by Dell to date, the second and third methods would also be fairly easy to implement and use in the factory. Both of these methods require data to be input into the procedures in a readily producible format. In addition, both methods can easily accommodate changes in demand, changes in the number of families, changes in the part requirements of any families, and changes in the number of production lines. Although production lines are not readily constructed or removed in the physical factory, this last feature allows the factory to adjust the number of production lines that are running during any production shift.

The poor performance of the third method (the combined models scheduling method) gives rise to a number of further research possibilities. There are two primary ways to improve the cGRASP. One is to make modifications to shorten the run time which would allow more solutions to be generated and, hopefully, lead to better solutions. The other is to modify the procedure to improve the solutions that are generated.

The ideas we have to shorten the runtimes are to improve the logic for the existing routines within the code and to use parameter feedback. One way the logic can be improved is to iteratively cascade demand reassignments from the line with the highest demand through the other lines to the lines with the lowest demand in during Phase 1, Step 8. This would reduce the number of solutions that were rejected due to not being able to fix demand spread issues – our main problem encountered during scenario 5, the scenario with the longest run-time. Another way to improve the logic is to insert nodes into the sequence rather than append them to the end of the sequence to add required node-visits during Phase 1, Step 11.c. This would increase the number of feasible arcs that could be used to fix the node visitation constraints and therefore reduce the number of solutions that are rejected due to being unable to make these constraints feasible. It may also speed up Phase 1, Step 11.

The idea of using parameter feedback consists of allowing the GRASP program to "learn" over the course of the run by improving certain parameters. This could be particularly effective in regard to distributing the demand across the production lines. It may be possible to introduce a weight parameter that alters the probability of each potential arc being selected in step 4 and this weight parameter could be adjusted as solutions are being constructed to help mitigate the situation that happened in scenario 5.

This could result in fewer solutions being rejected due to infeasible demand spreads which would lower run times and possibly result in better solutions.

The ideas we have to improve the solutions generated include using real examples of initial family assignments, developing an alternative hybrid GRASP, and adding another improvement step that can alter the family-to-line assignments. Using real examples of initial family assignments would allow the cGRASP to select realistic initial nodes for each line and start the assignment process with reasonable family combinations on each line. This will prevent the generation of many of the poorest feasible solutions, and likely some of the discarded solutions, and therefore improve the setup costs of the final schedule.

An alternative GRASP that could improve the solutions would be a hybrid of the SLGRASP and the cGRASP. This hybrid GRASP (hGRASP) would incorporate steps 1 through 8.b of the cGRASP with the full SLGRASP. In essence, this hGRASP would generate initial solutions in the same way as the cGRASP and fix any demand spread issues, but would not attempt to fix node-visitation issues. Instead, the family-to-line assignments and demand assignments for each line would be fed into the SLGRASP code which would then generate and improve feasible solutions. It is not known if this hGRASP would run faster or slower than the cGRASP, but it should lead to better solutions.

And one other way to generate better solutions is to include a solution improvement procedure that has the power to change the family-to-line assignments. Currently, Phase 2 improvements do not alter family-to-line assignments because of the complexity involved in maintaining solution feasibility while assignment changes are made without completely regenerating each affected line's path. One idea recently

introduced in the literature is path-relinking. This method may be useful in conquering this difficulty and thereby allowing improvement to Phase 2.

One outstanding research question related to our work is how each scheduling method will impact delivery lead-times. Due to the fact that the parts for each product families spend more time on the production line than their expected orders would require, the schedules resulting from the separated modes and the combined model both offer a little less flexibility to the time-scheduling step. As a result, it is anticipated that the delivery lead-times will be larger with both the separated models and combined models approaches than with the initial approach currently in use by Dell. This increase is expected to be very small though, and allow the delivery lead-times to stay well within the requirements set by Dell. This research question can be addressed through a simulation that iteratively runs the scheduling procedure, simulates the random demand experienced and the production carried out according to the schedule during a production shift, and then feeds the remaining demand information back to the scheduling procedure.

One other possible avenue of future research is the investigation of the applicability of stochastic programming to this model. Due to the random nature of demand, stochastic programming seems a logical choice but stochastic programs are often extremely difficult to solve so the first move in that direction would be to develop and solve a stochastic assignment model that addresses the first step of the scheduling problem. Work is currently under way to address this topic area.

References

Agnetis, A., Alfieri, A., Nicosia, G. 2004. A heuristic approach to batching and scheduling a single machine to minimize setup costs. *Computers & Industrial Engineering*, 46, 793-802.

Al-Fawzan, M. A. and K. S. Al-Sultan. 2002. A tabu search based algorithm for minimizing the number of tool switches on a flexible machine. *Computers and Industrial Engineering*, 44, 35-47.

Allahverdi, A., Gupta, J. N.D. and T. Aldowaisan. 1999. A review of scheduling research involving setup considerations. *Omega, The International Journal of Management Science*, 27, 219-239.

Allahverdi, A. Ng, C.T., Cheng, T.C. E., Kovalyov, M. Y. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research* (to appear).

Balakrishnan, A., Vanderbeck, F. 1999. A tactical planning model for mixed-model electronics assembly operations. *Operations Research,* 47(3); 395-409.

Baptiste, P., Le Pape, C. 2005. Scheduling a single machine to minimize a regular objective function under setup constraints. *Discrete Optimization*, 2, 83-99.

Boudia, M., M.A.O. Louly, and C. Prins. 2006. A reactive GRASP and path relinking for a combined production-distribution problem. *Computers & Operations Research*. In Press.

Breen, B. 2004. Living in Dell Time. *Fast Company*, 88, 86.

Cattrysse, D.G. and Van Wassenhove, L.N., 1992. A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, Vol. 60, 260-272.

Cheng, T.C.E., Gupta, J.N.D. and Wang, G., "A review of flowshop scheduling research with setup times", *Production and Operations Management*, Fall 2000, Vol. 9, No. 3, 262-282.

Crama, Y. and J. van de Klundert. 1999. Worst-case performance approximation algorithms for tool management problems. *Naval Research Logistics*, 46, 445-462.

Dell, M. and J. Magretta. 1998. The power of virtual integration: an interview with Dell Computer's Michael Dell. *Harvard Business Review*, March-April 1998, 72-84.

Feo, T. A., Bard, J. F. 1989. Flight scheduling and maintenance base planning. *Management Science,* 35(12); 1415-1432.

Feo, T. A., Resende, Mauricio G. C. 1995. Greedy randomized adaptive search procedures. *Journal of Global Optimization,* 6; 109-134.

Feo, T. A., Venkatraman, K., Bard, J. F. 1991. A GRASP for a difficult single machine scheduling problem. *Computers & Operations Research,* 18(8); 635-643.

Foulds, L.R. and Wilson, J.M. 1997. A variation of the generalized assignment problem arising in the New Zealand dairy industry. *Annals of Operations Research,* Vol. 69, 105-114.

Jaillet, P. and A. Odoni. 1988. The probabilistic vehicle routing problem. In *Vehicle Routing: Methods and Studies*. B. L. Golden, B. L. and A. A. Assad, editors. North Holland, Amsterdam, Netherlands; 293-318.

Laporte, G., Riera-Ledesma, J., Salazar-González, J. J. 2003. A branch-and-cut algorithm for the undirected traveling purchaser problem. *Operations Research*, 51(6); 940-951.

Lawler, E.L., Lenstra, J.K., Rinnooy, Kan A.H.G. and Shmoys D.B. 1993. Sequencing and scheduling: algorithms and complexity. In *Logistics of Production and Inventory*. Graves S.C., Rinnooy Kan A.H.G. and Zipkin P.H., editors. North Holland, Amsterdam, Netherlands; 445-522.

Liu, C.Y., Chang, S.C. 2000. Scheduling flexible flow shops with sequence-dependent setup effects. *IEEE Transactions on Robotics and Automation*. 16, 408-419.

Loveland, J. L., Monkman S. K., Morrice, D. J. 2006. Dell uses new production scheduling heuristics to accommodate increased product variety. Submitted to Interfaces in 2005.

Monkman, S. K., D. J. Morrice and J. F. Bard. 2006. A production scheduling heuristic for an electronics manufacturer with sequence dependent setup costs. Forthcoming in *European Journal of Operational Research*.

Miller, D.M., Chen, H.C., Matson, J., Liu, Q. 1999. A hybrid genetic algorithm for the single machine scheduling problem. *Journal of Heuristics,* 5, 437-454.

Mittenthal, J., Noon, C. E. 1992. An insert/delete heuristic for the traveling salesman subset-tour problem with one additional constraint. *Journal of the Operational Research Society,* 33(3); 277-283.

Pinto, J. M. and I. E. Grossmann. 1988. Assignment and sequencing models for the scheduling of process systems. *Annals of Operations Research*, 81, 433-466.

Ramesh, T., "Traveling purchaser problem", *Opsearch*, 1981, Vol. 18, 78-91.

Renaud, J., Boctor, F. F. 1998. An efficient composite heuristic for the symmetric generalized traveling salesman problem. *European Journal of Operations Research,* 108; 571-584.

Stevenson, W. J. 2001. *Operations Management*. 7th Edition. Irwin/McGraw Hill.

Toth, P. and D. Vigo, eds. 2002. *The Vehicle Routing Problem*. SAIM. Philadelphia.

Tranrisever, F. and D. J. Morrice. 2005. Designing and operating a high-volume, make-to-order assembly line under uncertain demand. Working Paper. The University of Texas at Austin.

Vignier, A., Sonntag, B., Portmann, M.C. 1999. Hybrid method for a parallel-machine scheduling problem. *IEEE Symposium on Emerging Technologies and Factory Automation, ETFA,* 1, 671-678.

Webster, S. and K. R. Baker. 1995. Scheduling groups of jobs on a single machine. *Operations Research*. Vol. 43, No. 4, 692-703.

Vita

Dr. Susan Monkman was born Susan Kathleen Heath to Judith Ellen Heath and Dr. David Clay Heath on February 12th, 1972 in St. Paul, Minnesota. She grew up primarily in Ithaca, NY under the shadow of Cornell University. In January of 1995 she earned a Bachelor of Arts degree in Psychology from Cornell University. After a brief venture into the working world, she returned to Cornell and earned a Master of Engineering degree in Operations Research and Industrial Engineering in May of 1997. Following this she worked in Information Systems supporting manufacturing operations for Kraft Foods, Inc. in their Champaign, IL plant. In 2000, she moved to Austin, TX to attend the University of Texas at Austin and work on a doctoral degree in Operations Management.

REFEREED PROCEEDING

1) Monkman, Susan K., Douglas J. Morrice and Jonathan F. Bard. "Scheduling Product Families in a High Volume, Flexible, Assemble-to-Order Factory." 2005. Proceedings of The 2nd Multidisciplinary International Conference on Scheduling: Theory & Applications. Graham Kendall, Lei Lei, Michael Pinedo, eds. pp 394-395.

WORKING PAPERS

1) Loveland, Jennifer, Susan K. Monkman, and Douglas J. Morrice. "Dell Uses New Production Scheduling Heuristics to Accommodate Increased Product Variety." 2005. (submitted to Interfaces)
2) Monkman, Susan K., Douglas J. Morrice and Jonathan F. Bard. "A Production Scheduling Heuristic for an Electronics Manufacturer with Sequence Dependent Setup Costs." 2005. (forthcoming in European Journal of Operational Research)
3) Monkman, Susan K., Douglas J. Morrice and Jonathan F. Bard. "Scheduling Product Families in a High Volume, Flexible, Assemble-to-Order Factory." 2005 working paper.

Permanent address: 9805 Marlborough Drive, Austin, TX 78753

This dissertation was typed by the author.