

MODELS AND ALGORITHMS FOR WORKFORCE ALLOCATION AND UTILIZATION

by

Ada Yetunde Barlatt

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in The University of Michigan
2009

Doctoral Committee:

Assistant Professor Amy E. M. Cohn, Chair
Professor Brian Talbot
Associate Professor Marina A. Epelman
Oleg Y. Gusikhin, Ford Motor Company

ACKNOWLEDGEMENTS

I would like to thank my family and friends for their support. I would also like to thank my dissertation committee for their feedback and suggestions. I would especially like to thank my research advisor, Amy Cohn, and our industry collaborators at the Ford Motor Company, Oleg Gusikhin, Yakov Fradkin and Craig Morford, without them this dissertation would not have been possible.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	ii
LIST OF FIGURES	v
LIST OF TABLES	vi
LIST OF APPENDICES	vii
CHAPTER	
I. Introduction	1
1.1 Workforce Allocation and Utilization	3
1.1.1 Examples	3
1.1.2 Shift-Workforce Allocation and Utilization (SWA-WU)	4
1.1.3 Modeling Challenges	9
1.2 Solution Techniques Overview	11
1.2.1 Shift Schedule and Extreme Shift Schedule Variables	11
1.2.2 Test-and-Prune (T&P)	12
1.3 Organization of the Dissertation	14
II. Shift Schedule Formulation	16
2.1 Composite Variable Modeling	18
2.2 Shift Schedule Formulation	20
2.2.1 Unsuccessful Modeling Approaches	21
2.2.2 Shift Schedule Definition	22
2.2.3 Shift Schedule Formulation	25
2.2.4 Using Delayed Column Generation to Solve the Formulation	28
2.2.5 Extreme Shift Schedule Formulation	30
2.2.6 Computational Results	35
2.3 Conclusions	35
III. Test-and-Prune (T&P)	39
3.1 Weak Linear Programming Relaxations	40
3.2 T&P Motivation	42
3.3 T&P	42
3.3.1 T&P Algorithm Details	43
3.4 Performance Improvements for T&P	46
3.5 T&P Enhancements	48
3.5.1 T&P and a Non-Linear Cost Function	48
3.5.2 T&P and Feasibility Problems that are not Mathematical Programs	48

3.5.3	T&P Neighborhood when the Workforce Utilization Decisions Impact Cost	49
3.6	T&P and Benders Decomposition	49
3.7	Conclusions	53
IV.	Case Studies	55
4.1	Stamping Problem	56
4.1.1	Assumptions	63
4.1.2	Input Data	64
4.1.3	Modeling Challenges	65
4.2	Single Pressline Decisions	66
4.2.1	Problem Statement	66
4.2.2	Shift Schedule Formulation	66
4.2.3	Extreme Shift Schedule Formulation	72
4.2.4	Computational Results	74
4.3	Facility Operating Decisions	85
4.3.1	Problem Statement	85
4.3.2	T&P Feature 1: Many Feasibility Problems	88
4.3.3	T&P Feature 2: Pre-Processing Non-Viable Workforce Allocations	90
4.3.4	T&P Feature 3: Pruning Feasibility Problems	91
4.4	Pressline Zone Decisions	92
4.4.1	Problem Statement	92
4.4.2	Lower Bound	93
4.4.3	Solution Technique for When Each Pressline Has Two Distinct Direct Labor Requirements	98
4.4.4	Upper Bound	101
4.5	Conclusions	103
V.	Conclusions	105
5.1	Work in Shift-Workforce Allocation and Utilization	105
5.2	Contributions	107
5.3	Future Work	109
	APPENDICES	113
	BIBLIOGRAPHY	131

LIST OF FIGURES

Figure

2.1	Sample Shift Schedules	23
2.2	Sample Shift Schedules: Quantity Parameter	24
2.3	Sample Shift Schedules: First and Last Parameter	24
2.4	Sample Shift Schedules: Non-Zero Quantity Parameter	25
2.5	Feasible Region: One-Task Shift Schedules	31
2.6	Feasible Region: Two-Task Shift Schedule	32
3.1	Test-and-Prune Flowchart	44
4.1	Stamping Facility Flow	57
4.2	Stamping Press [1]	58
4.3	Feasible Changeovers and Off-line Preparation	58
4.4	Pressline Sample Shift Schedules: Changeover Parameter	68
4.5	Feasible Region: Pressline Two-Task Shift Schedule	74

LIST OF TABLES

Table

2.1	Number of Shift Schedules Required Comparison	33
2.2	Single Pressline Initial Results	36
4.1	Sample Pressline Schedule	61
4.2	Pressline Number of Shift Schedules Required Comparison	74
4.3	Single Pressline Characteristics	75
4.4	Single Pressline Initial Results	76
4.5	Single Pressline Computational Results with T&P	83
4.6	Single Pressline T&P Pre-processing Impact	84
4.7	Facility Operating Costs Branch-and-Bound	88
4.8	Facility Operating Costs Disaggregate Feasibility Problems	90
4.9	Facility Operating Costs Pre-processing Feasibility Problems	91
4.10	Facility Operating Costs Pre-processing and Pruning Feasibility Problems	92
4.11	Pressline Zones	96
4.12	Pressline Zones Lower Bound Branch-and-Bound	96
4.13	Pressline Zones Lower Bound T&P	97
4.14	Workforce Distribution Solution Example	98
4.15	Pressline Zones Solutions	100
4.16	Pressline Zones Gap Between Lower and Upper Bound	103

LIST OF APPENDICES

Appendix

A.	Single Machine Scheduling Literature	114
B.	Rotated Case Study Solutions Maintain Feasibility Status	117
	B.1 Feasible solution exists	117
	B.2 Feasible solution does not exist	119
C.	Additional Stamping Extreme Shift Schedules	120
	C.1 Three part type shift schedules	121
	C.2 Two part type shift schedules with a minimum duration of production re- quired directly after each on-line dieset	125
	C.3 Three part type shift schedules with a minimum duration of production re- quired directly after each on-line dieset	127

CHAPTER I

Introduction

This dissertation presents novel mathematical models and algorithms to accurately represent and efficiently solve workforce planning problems. Workforce planning is an important process that: (1) enables organizations to determine the most efficient workforce composition, and (2) provides a basis to recruit and/or reorganize the workforce to achieve organizational goals.

A workforce plan is a framework for making staffing decisions based on an organization's mission, strategic plan, budget, and a set of required worker skills. An effective workforce plan has the right number of workers with the right skills in the right place at the right time [80]. Unfortunately, simultaneously determining the *workforce allocation* – the number of workers with each skill set available during the planning horizon – and the *workforce utilization* – the sequence of tasks scheduled during the planning horizon to meet customer demand – is not a trivial task. Workforce planning problems often have characteristics that are either difficult to model or pose challenges to tractability when using *mathematical programming (MP)* techniques, including:

Non-linear relationships: When a worker must be paid for an entire shift even if she is not assigned enough tasks to fill the shift, the workforce cost is a non-linear

function of the duration of tasks assigned. Non-linear relationships between decisions like this one are quite common in workforce planning problems. Unfortunately, the most successful methods for solving mathematical models are found in cases where the objective function and all constraints are linear. These non-linear relationships add significant difficulty to modeling workforce planning problems.

Weak *linear programming (LP)* relaxation: There is a strong interaction between workforce allocation decisions (i.e., how many workers to hire and what skills each worker should have) and workforce utilization decisions (i.e., what tasks to assign each worker). This interaction suggests that there could be several benefits for solving the two problems simultaneously. However, when a worker must be present for the entire shift even if she is not assigned enough tasks to fill the shift, these problems have a weak *LP* relaxation. Instead of integer workers, the *LP* relaxation allocates fractional workers to avoid paying for unused capacity. These fractional allocation variables lead to large *branch-and-bound* trees and very slow run times. Weak *LP* relaxations present a significant challenge to solving workforce planning problems.

Using an *automotive stamping manufacturing* environment as a demonstrative example, this dissertation presents solution techniques that leverage strengths of both mathematical programming and *search-based algorithms* to overcome these modeling and tractability challenges. Computational results based on data from a major automotive manufacturer demonstrate how the models and algorithms developed provide high-quality, realistic solutions in reasonable run times.

This chapter is organized as follows. Section 1.1 describes integrated workforce allocation and utilization problems. This section presents examples of integrated workforce allocation and utilization problems and describes *Shift-Workforce Allocation and Utilization problems (SWA-WU)*. *SWA-WU* is the class of integrated

workforce allocation and utilization problems addressed in this dissertation. The section concludes with a discussion of the challenges to accurately modeling and efficiently solving *SWA-WU*. A high-level overview of the solution techniques used in the dissertation is discussed in section 1.2. The chapter concludes with the organization of the dissertation in section 1.3.

1.1 Workforce Allocation and Utilization

Many workforce planning problems include two sets of decisions: how many workers to hire (workforce allocation) and how to assign these workers to tasks (workforce utilization).

Often these decisions are made sequentially to achieve tractability (e.g., [20] and [41]). Considering both decisions simultaneously, however, may make it possible to complete all of the tasks in a reduced amount of time or with a reduced number of workers. This can result in significant savings in facility operating costs, overhead costs, etc., which often dominate the cost of performing the actual tasks. The integration of the workforce allocation (*WA*) decisions with the workforce utilization (*WU*) decisions will be referred to as *Workforce Allocation and Utilization (WA-WU)*.

1.1.1 Examples

Examples of *WA-WU* can be found in many applications such as healthcare, manufacturing and call centers.

[23] reviews nurse rostering problems. This review paper addresses producing a periodic (e.g., weekly, bi-weekly, monthly, etc.) duty roster for nursing staff, subject to a variety of objectives (e.g., minimize infeasibility, minimize uncovered shifts, and minimize cost) and constraints (e.g., personnel policies, legal regulations, nurses preferences and many other hospital-specific requirements). Although not all nurse

rostering problems are *WA-WU*, this paper does discuss a set of nurse allocation and utilization problems where nurses are assigned specific tasks throughout the planning horizon (ex. [21] and [57]).

[2], [6], [39] and [64] all are examples of *WA-WU* problems found in manufacturing. In this environment every production schedule (i.e., task start times and durations) corresponds to a daily workforce allocation. Altering the production schedule can change the daily workforce allocation. The goal is to find the lowest-cost workforce allocation that corresponds to a production schedule that meets demand.

WA-WU also occurs in call centers. Unlike the manufacturing applications, the exact type and the number of tasks that need to be completed is not known a priori. Instead the input data for a *WA-WU* in a call center is a workforce requirement pattern (i.e., the number of workers required during every half-hour) and a desired service level. [42] and [82] discuss solution techniques used in this area.

WA-WU can also be found in a variety of other areas such as public transportation ([30]) and construction ([50]).

[38] provides an overview of the solution techniques used to model *WA-WU* and a summary of the areas in which *WA-WU* can be found.

1.1.2 Shift-Workforce Allocation and Utilization (SWA-WU)

This dissertation focuses on a set of problems within *WA-WU* which we refer to as Shift-Workforce Allocation and Utilization.

In *SWA-WU* the planning horizon is broken into a finite set of shifts. In the most basic case, shift-workforce allocation decisions select a set of shifts where workers are available to complete tasks. These shifts might represent time periods in a manufacturing plant, call shifts in a telemarketing facility, or available slots in a set of hospital operating rooms. Once a subset of these shifts has been “turned on”

(i.e., selected for workers to be available), these shifts become available to schedule tasks such as final assembly, customer surveys, or surgical procedures; conversely, workers are unavailable and tasks are prevented from being scheduled during those shifts which have not been selected.

Note that the candidate shifts need not necessarily represent contiguous blocks of time. Examples of this can be found in manufacturing. For instance, the automotive stamping problem considered in this dissertation considers three daily shifts of eight hours each (first, second, and third shift). The planning horizon spans multiple days, however, and if the first shift is “turned on,” then workers must be available for *all* first shifts in that horizon, not just a single day.

This definition of shift-workforce allocation can be broadened to not only include the single binary decision for each shift of whether or not to enable it, but also a wider class of worker characteristics for each shift. For example, a manufacturing facility might have workers with three distinct skill sets (A , B , and C) and workers in each skill set are trained to complete a specific set of tasks. Thus, instead of one decision for each candidate shift s in the planning horizon, this problem might have three: Should workers of skill set A be available during shift s ? Should workers of skill set B be available during shift s ? Should workers of skill set C be available during shift s ? A telemarketing call center may care not only about whether or not calls are being made, but which language skills the employees have (e.g., Can calls be made to Spanish-speaking households?). Likewise in healthcare, surgeons are qualified to complete different procedures and thus the question is no longer “Are surgeons available to preform operations during shift s ?” but becomes “Are type o surgeons available to preform operations during shift s ?” for some finite set of surgeon types.

Moreover, the shift-workforce decision can assign capacity to shifts, where the decision of *how many* is added to turning “on” a characteristic. This is required when more than one worker is required to complete a single task. An example of this would be “Should *five* workers of skill set *A* be available during shift *s*?”

Adding further complexity to the selection of shift-workforce characteristics is the fact that there may be interdependence between these decisions. For example, in the manufacturing context, workers with skill set *A* may never be needed at the same time that a worker with skill set *B* is needed. Alternatively, while a worker with skill set *C* is completing tasks, a worker with either skill set *A* or skill set *B* must be present.

Finally, the cost of the chosen shift-workforce characteristics may not be linear with respect to the number of tasks occurring simultaneously. For example, in surgical scheduling it might be necessary to have an administrator available whenever any one of the operating rooms is scheduled, but this administrator can be shared across all of the enabled operating rooms and thus the associated cost does not increase as the number of scheduled suites increases.

In the *SWA-WU* problem, the goal is to find the minimum-cost workforce allocation such that it provides adequate opportunity to sequence a collection of pre-defined tasks.

Shift-Workforce Allocation (SWA) Decisions

A shift-workforce allocation is the number of workers with each skill set available during each shift. The assumptions concerning *SWA* decisions are as follows:

1. There is a heterogeneous workforce. This workforce can be described with a finite set of worker skill sets.

2. The set of candidate shifts is finite, discrete, and pre-specified.
3. The length of each shift is pre-determined (but shifts are not all required to be the same length).
4. Workers are scheduled to work for entire shifts. A worker must be paid for the entire shift even if she is not assigned enough tasks to fill the shift.
5. At a minimum, there is a decision for each shift of whether to turn it “on” or “off”. More broadly, there may be multiple decisions for each shift corresponding to a variety of characteristics. For example, in manufacturing, there may be decisions about the number of workers of each skill set available during each shift.
6. The fixed cost associated with the size, availability and skills of the workforce allocated (i.e., the *SWA* solution) is substantial. It greatly dominates any additional costs associated with the *workforce utilization (WU)* decisions of how to perform tasks within the chosen shifts.
7. The *SWA-WU* cost function is restricted to be linear in the *SWA* decisions. Note that the objective does not need to be linear with respect to the *WU* decisions. In other words, the workforce cost grows linearly with respect to the number of workers hired, however the workforce cost does not have to be linear function of the duration (or number) of tasks assigned.
8. Each *SWA* decision imposes a capacity limit on the *WU* decisions, which can be written as a linear constraint or set of linear constraints; the *SWA* decision serves as the right-hand side (i.e., upper bound) of these constraints. An example of this is : $\sum_t l_t x_{ts} \leq d_s y_s$, where x_{ts} equals one if task t is assigned during shift

s (zero, otherwise), y_s equals one if s is “turned on” (zero, otherwise), d_s is the duration of shift s , and l_t is the length of task t . In other words, the total time that workers are available (and tasks can be scheduled) within a given shift s is the length of that shift if it is “turned on”, or zero if it is not.

9. There may be interdependencies between *SWA* decisions. For example, there may be restrictions on the minimum and maximum number of shifts to be operated.

Workforce Utilization (WU) Decisions

The workforce utilization is the sequence of tasks scheduled during the planning horizon to meet customer demand. The assumptions concerning *WU* decisions are as follows:

1. The planning horizon is finite.
2. The set of tasks to be completed (e.g., products to be manufactured) during this planning horizon is finite, deterministic, and static.
3. The tasks are to be completed on a fixed set of workstations.
4. Each task has a pre-specified set of requirements (e.g., number and skills required for the workers). Tasks are not required to all be of the same length.
5. There is a set of constraints to be satisfied by the task sequence. These constraints may include due dates, operational restrictions, resource limitations, etc. It is assumed that these constraints can be formulated as a tractable *mixed integer program (MIP)*.
6. The cost of *WU* decisions is dominated by the cost of the *SWA* decisions.

Operational Constraints

Similar to *WA-WU*, *SWA-WU* problems can be found in many applications. Across applications there may be a wide variety of different operational constraints. The models and algorithms in this dissertation are applicable in situations where:

1. Preemption of tasks is possible. As stated in [73], preemption implies that it is not necessary to keep a task on a workstation, once started, until completion. The schedule can interrupt the work on a task at any point in time and put a different task on the workstation instead. The amount of processing of the task is not lost. When a preempted task resumes, it only needs to be processed for the remaining time.
2. Changeovers (i.e., set ups) occur between tasks. The duration of the changeovers is allowed to be a function of other attributes (e.g., preceding task, succeeding task, shift it is assigned to, etc.) If changeovers occur, *these changeovers can not span shift boundaries*. This is so that one group of workers is responsible for the entire changeover.

However, the models and algorithms discussed in this dissertation can be used to provide solutions to problems without these operational constraints.

1.1.3 Modeling Challenges

Workforce utilization problems are often quite challenging to solve by themselves (ex. [12], [43], [70], [74]); integrating workforce allocation and utilization decisions yields even greater challenges [15]. There are several structural aspects of *SWA-WU* that can make such problems difficult to model and solve.

First, there are complex operational constraints that must be captured in the model. Recall that changeover durations are allowed to be a function of other at-

tributes (e.g., task sequence). The model must capture the sequence of tasks in order to enforce the proper changeover duration. In addition, the model must track the start times of the changeovers to ensure that they do not cross between two shifts. Therefore, the challenge is how to define variables and constraints that capture the shift boundary constraints as well as the sequence-dependent changeover durations.

Second, the total workforce cost is non-linear. A worker is scheduled to work for an entire shift and must be paid for entire shift even if she is not assigned enough tasks to fill the shift. Thus, the staffing level for a given shift is defined as the maximum worker requirement at any point of time within the shift. Defining variables and constraints that enforce these non-linear worker constraints poses an additional modeling challenge.

Third, when *SWA-WU* can be formulated as a *MIP*, it typically will pose a weak *LP* relaxation. This is because the objective value can be decreased in a fractional solution by matching the capacity of the *WU* decisions to the exact value required by the *SWA* decisions. This is caused by the constraints that link together *SWA* decisions (which assign capacity) and *WU* decisions (which use this capacity). Whenever capacity is not fully utilized, fractional values for the allocation decisions can reduce cost in the *LP* relaxation.

For example, recall the constraint described earlier of $\sum_t l_t x_{ts} \leq d_s y_s$, where x_{ts} equals one if task t is assigned during shift s (zero, otherwise), y_s equals one if s is “turned on” (zero, otherwise), d_s is the duration of shift s , and l_t is the length of task t . In other words, the total time that workers are available (and tasks can be scheduled) within a given shift s is the length of that shift if it is “turned on” or zero if it is not. In this case, the cost of the *LP* relaxation is minimized by fixing y_s at $\sum_t \frac{l_t x_{ts}}{d_s}$, which may well be fractional. In practice, very poor convergence of

the branch-and-bound tree is often observed in such problems. [See [4], [32] and [60] for examples of this in other problem domains.] Thus, the challenge is how to capture the relationship between *SWA* and *WU* decisions and overcome a weak *LP* relaxation.

1.2 Solution Techniques Overview

To overcome the modeling challenges presented above, this dissertation utilizes variables that represent entire shifts (referred to as *shift schedules*) to accurately model the *SWA-WU*. To overcome the tractability challenges caused by a weak *LP* relaxation, this dissertation presents a novel algorithm, *Test-and-Prune*, to efficiently solve the *SWA-WU*.

1.2.1 Shift Schedule and Extreme Shift Schedule Variables

In the *SWA-WU*, it is important to note that the complexity is largely restricted to individual shifts. For example, changeovers must be fully contained within a shift, and worker calculations are made at the shift level as well. This dissertation presents a modeling approach in which the variables correspond to feasible sequences and durations of tasks completed within a single shift. We refer to these variables as *shift schedules*. For example, one shift schedule might include eight hours of work on task *A*. Another shift schedule might begin with the workers working on task *A* for the first two hours, then a changeover to task *B* takes place and workers work on task *B* for the remainder of the shift.

By defining the variables in this way, many of the constraints are automatically enforced – a shift schedule is not defined if it does not satisfy the changeover operational rules. Additionally, associated with each shift schedule are characteristics – the sequence of tasks, the number of changeovers, etc. – that can be used in the

remaining constraints, as well as the objective function.

These benefits could come at the expense of a (potentially) exponentially large number of variables – one for every feasible shift schedule. However, by recognizing that any shift schedule can be written as a convex combination of a small, carefully-selected set of specialized shift schedules, which we refer to as *extreme shift schedules*, the number of variables required is significantly reduced. For example, when considering eight-hour shifts, a shift schedule in which task A is worked on for four hours (the remaining four hours are idle) can be represented by taking the average of two extreme shift schedules, one with eight hours of A and the other with eight hours of idle. In fact, any combination of work on task A plus idle time can be found by taking a convex combination of these two extreme shift schedules.

In Chapter II, we present a mixed integer program based on the extreme shift schedules. We present computational results in Chapter II using data sets provided by a major automotive manufacturer. Branch-and-bound solves many of those problem instances. However, it is insufficient for others, due to the weakness of the linear programming relaxation.

1.2.2 Test-and-Prune (T&P)

To address this source of intractability, we develop an alternative algorithm called *Test-and-Prune* ($T\mathcal{E}P$) in Chapter III.

Two observations motivate $T\mathcal{E}P$.

1. Number of possible workforce allocations is finite. Recall that a workforce allocation is the number of workers with each skill set available during each shift. A unique workforce allocation can correspond to several feasible workforce utilization solutions (i.e., feasible sequence and duration of tasks). Although there

is possibly an infinite number of distinct feasible workforce utilization solutions, the number of workforce allocations is finite and often quite small.

2. When the workforce allocation is fixed, a feasibility problem remains. Recall that the objective for the *SWA-WU* only considers the workforce costs; therefore when the workforce allocation is fixed (i.e., specified as input parameters, rather than as decision variables), the extreme shift schedule formulation becomes a feasibility problem. This feasibility problem answers the question, “Can a feasible schedule be created with the given worker allocation?”

Therefore, instead of solving a single optimization problem, *T&P* solves several feasibility problems (one for each workforce allocation) and then selects the feasible solution with lowest cost.

We can improve the performance of the *T&P* algorithm by recognizing that information gained when solving one feasibility problem can give a priori information about the results for other feasibility problems. We can reduce the number number of workforce allocations solved by using this information to prune the list. If an allocation is feasible, any allocation with higher cost can be pruned. Any allocation with higher cost than the feasible allocation is sub-optimal. If the allocation is infeasible, any allocation with fewer or the same number of workers in each shift type can be pruned. Any allocation with the same or fewer number of workers has fewer resources and will also be infeasible.

We use several workforce planning problems in an automotive stamping plant to demonstrate the extreme shift schedule model and *T&P* algorithm in Chapter IV.

1.3 Organization of the Dissertation

This dissertation is organized into five chapters. The first is this introduction which describes *WA-WU* and the *SWA-WU* problems that will be explored in this dissertation.

Chapter II presents a shift schedule model for *SWA-WU*. In this model, much of the problem complexity is embedded in the shift schedule variables. The characteristics associated with the shift schedules are used in the constraints and the objective function. The number of shift schedule variables required for the formulation is quite small. This is due to the fact that all shift schedules can be represented as convex combinations of a limited set of extreme shift schedules. The shift schedule formulation extends the literature by allowing for sequence-dependent changeover times and varying due dates. Moreover, this approach does not restrict batch sizes, workforce availability, or sequencing of part types or the number of changeovers.

Chapter III presents a novel algorithmic approach, which we call *Test-and-Prune*, to solve the extreme shift schedule formulation. *T&P* uses mathematical programming as a mechanism for solving simpler feasibility problems that are embedded in a larger search-based algorithm. This search-based algorithm is able to leverage dominance in order to achieve substantial pruning.

Chapter IV describes an automotive stamping plant and presents several automotive stamping case studies using the extreme shift schedule formulation and the *T&P* algorithm. Computational experiments using data provided by a major automotive manufacturer are presented to demonstrate the effectiveness of the *T&P* algorithm and compare the *T&P* algorithm to traditional integer programming methods.

Chapter V concludes by discussing the contributions of this dissertation and sug-

gesting areas of future research. There are three key contributions of this dissertation: (1) accurately modeling and efficiently solving a challenging set of workforce planning problems (i.e., *SWA-WU*), (2) demonstrating that all of the required shift schedules can be represented by a small set of extreme shift schedules, and (3) developing an algorithm, *T \mathcal{E} P*, that can successfully solve a variety of *SWA-WU* problems – this is demonstrated via automotive stamping case studies. In addition, *T \mathcal{E} P* can be applied to other classes of resource allocation of problems.

CHAPTER II

Shift Schedule Formulation

This chapter develops a model for Shift-Workforce Allocation and Utilization problems. *SWA-WU* integrates two sets of decisions: (1) the number of workers of each skill set available during each shift (workforce allocation) and (2) the sequence and duration of tasks to meet demand (workforce utilization). The objective is to determine the lowest-cost workforce allocation that corresponds to a feasible workforce utilization.

This problem is difficult to solve not only because of the general challenges found in most workforce planning problems, but additionally because of the complex operational requirements placed on the changeovers from one task to another in *SWA-WU* (the assumptions regarding changeovers are described in Chapter I).

To accurately model *SWA-WU*, this chapter presents a formulation based on *composite variables (CV)*. Each *CV* represent multiple decisions simultaneously. *Composite variable modeling (CVM)* has been successful in improving realism and tractability for many real-world applications with complex operational constraints in transportation and logistics (for example, [3],[4], [8], [10], [22], [26], [27], and [28]).

This chapter shows that *CVM* can also be successful for *SWA-WU*. Each *CV* in the formulation presented in this chapter corresponds to a feasible sequence and du-

ration of tasks completed within in a shift. Recall that in *SWA-WU*, the complexity is largely restricted to individual shifts. For example, changeovers must be fully contained within a shift, and worker requirements calculations are made at the shift level as well. We refer to these *CVs* as *shift schedules*. For example, one shift schedule might represent eight hours of work on task *A*. Another shift schedule might begin with the workers working on task *B* for the first hour, then a changeover to task *A* takes place and workers work on task *A* for the remainder of the shift.

The shift schedule variable definition embeds much of the operational complexity within the individual variables themselves, rather than through the use of complicating constraints. Furthermore, this variable definition does not require that batch sizes, number of changeovers, workforce availability, or sequencing of tasks be restricted or pre-defined in order to achieve tractability, as is the case in much of the scheduling literature. In addition, the shift schedule variables enable several changeover policies and varying due dates, also an enhancement over much of the scheduling literature.

There are three contributions of the research presented in this chapter. First, we present a formulation that accurately models *SWA-WU*. Second, we show that there are several benefits to using *CVs* to model *SWA-WU*. In particular, composite variables that represent entire schedules for individual shifts within the planning horizon can capture significant operational complexity. These variables also easily capture sequence-dependent changeover times.

The benefits of a composite variable modeling approach typically come at the cost of a very large number of binary variables and/or restrictions on the solution space such as the discretization of time. The third contribution of this work is in recognizing that all of the feasible shift schedules can be represented as a convex

combination of a small, select subset of specialized shift schedules. This enables the creation of an enhanced model that exhaustively captures all possible schedules (i.e., the set of shift schedules considered is not limited and time is not discretized), while keeping the number of *CVs* small. As an added benefit, the integrality of the remaining composite variables can be relaxed, leaving only a small set of auxiliary variables restricted to be integer.

The chapter is organized as follows. The composite variable modeling literature is reviewed in Section 2.1. The shift schedule formulation, extreme shift schedule formulation, and computational results for the *SWA-WU* are presented in section 2.2. Section 2.3 is the conclusion with a summary of the results and motivation for the next chapter.

2.1 Composite Variable Modeling

A traditional mathematical programming approach is problematic when formulating the *SWA-WU*, largely due to the operational requirements on how and when changeovers take place, as well as the non-linear calculations for workforce requirements. For example, to determine the duration and sequence of part types for a single machine production planning problem, [33] uses heuristics to reduce the problem “to a manageable mathematical programming formulation.”

In a number of applications in transportation and logistics, complicating requirements have been addressed by embedding them within composite variables rather than through the use of constraints. Benefits can include: linearization of constraints and objective function; strengthened *LP* relaxations; and decreased numbers of constraints. In this research, the goal is to extend these benefits to *SWA-WU* as well. The term “composite variable modeling” refers not to a single specific technique,

but rather to a modeling philosophy or approach based on the idea of capturing multiple decisions simultaneously in a single binary variable. The use of composite variable modeling is by no means new. It has its origins in *Dantzig-Wolfe decomposition* [31], in which a “traditional” MP is decomposed into a new model whose variables represent the extreme points of the polyhedra formed by subsets of the original constraints. Modeling approaches which are derived by modifying or adding to the variables in an existing MP are also sometimes known as variable redefinition, inverse projection, or extended reformulation. Extensive discussions of these broad topics appear in [45], [54], [58], [66], and [79]. The particular issue of variable redefinition for integer and mixed-integer programming is addressed in detail in [9] and [81].

In CVM , the variables are not created via the decomposition of an alternative model. Instead, the CV s are defined directly, with the explicit intent of addressing application-specific complexity such as a non-linear objective function, a large block of complicating constraints, or a weak LP relaxation. An early and excellent example of CVM is seen in the cutting stock problem, in which each variable represents a particular pattern in which a board can be cut, and the candidate patterns are generated by solving instances of a knapsack problem [48].

CVM has seen much success in the area of transportation and logistics. One prevalent example is the airline crew scheduling problem (a survey of this literature appears in [7]). In this application, instead of using variables that represent the assignment of a crew to a single flight, each variable represents the assignment of a crew to an entire pairing — a multi-day tour of duty. In doing so, complex rules about limits on the amount of time between flights, overnight rest periods, etc., are addressed implicitly (if a pairing is infeasible, then there is no corresponding vari-

able defined) and explicit constraints are not needed to enforce them. Similarly, the non-linear cost function governing how crews are paid is eliminated because the total pay for a pairing can be computed off-line. A more recent example of *CVM* from the transportation and logistics literature is the work of [4], which solves the express shipment network design problem using composite variables to represent individual routes and the corresponding aircraft type assigned to fly these routes. [25] considers the problem of integrating the aircraft routing and crew scheduling problems in passenger aviation planning. In this case, the composite variables represent complete solutions to the aircraft routing problem. The use of dominance properties enables the candidate set of routing solutions to be greatly reduced. A service parts logistics problem is solved via *CVM* in [26]. This paper considers several different possible choices for the composite variable definition, assessing their relative pros and cons. The most successful approach uses *CVs* representing clusters of customers and their corresponding sources for high-cost, low-failure replacement parts. [34] solves the daily aircraft routing and scheduling problem by defining composite variables that assign fleet types to feasible schedules. Finally, [10] uses *CVM* to develop an itinerary-based formulation of the fleet assignment problem, using composite variables to capture the interdependence between itineraries sharing common flights.

This research extends the *CVM* literature by showing how the benefits *CVM* can be extended from transportation and logistics to *SWA-WU*.

2.2 Shift Schedule Formulation

In *SWA-WU*, one of the key modeling challenges is in how to capture the operating constraints associated with changeovers. Recall, changeover durations are allowed to be a function of other attributes (e.g., task sequence). The model must capture the

sequence of tasks in order to enforce the proper changeover duration. In addition, the model must track the start times of the changeovers to ensure that they do not cross between two shifts. Therefore, the challenge is how to define variables and constraints that capture the shift boundary constraints as well as the sequence-dependent changeover durations.

2.2.1 Unsuccessful Modeling Approaches

Capturing all of the information needed to enforce the changeover constraints in a traditional mathematical programming approach, in which each variable represents a single decision, is difficult if not impossible.

A common modeling approach to sequencing tasks is to have variable x_A equal the start time for task A (e.g., [17]). This variable can not be used to successfully model *SWA-WU*. Using this variable definition would make it difficult to capture the task sequence. Without information about the sequence, it would be a significant challenge to enforce the sequence-dependent changeover durations or capture the shift boundary changeover rules.

To capture the task sequence another index could be added. For example, [74] defines a binary variable x_{AB} , that equals one if task A directly proceeds task B and zero otherwise. Since this variable definition includes information about the task sequence it could be used to capture the sequence-dependant changeovers, however it would be a challenge to capture the sequence boundary rules.

To capture the shift boundary rules a time index could be used. An approach similar to [18] could be used by defining binary variables where x_{At_1} equaled one if task A ends at time t_1 and zero otherwise. This variable would be able to capture the shift boundary rules, because it would be easy to compare the end time of tasks to the time of the next shift boundary. However, capturing the sequence of tasks would

be a challenge using this variable definition (especially since tasks can be preempted). Without being able to capture the sequence, the model could not be able to enforce the sequence-dependent changeovers.

Binary variable $x_{At_1Bt_2}$, which equals one if work on task A begins at time t_1 and then work on task B begins at time t_2 , else zero, addresses these issues. However, using these variables result in a very large number of binary variables as well as a very large number of constraints to ensure continuity (e.g., if $x_{At_1Bt_2} = 1$ then there must exist some task C and some time t_3 for which $x_{Bt_2Ct_3} = 1$. In addition, constraints would be required to prohibit any tasks to begin between t_1 and t_2).

2.2.2 Shift Schedule Definition

In *CVM*, the goal is to embed complexity within the variable definition so as to simplify the objective function and/or complicating constraints. In *SWA-WU* it is helpful to take advantage of the fact that the complexity is largely shift-specific. For example, changeovers must be fully contained within a shift, and workforce calculations are made at the shift level as well. Therefore, we define composite variables for this problem to represent a feasible set of ordered tasks that can be completed in an individual shift. For example, one shift schedule might represent eight hours of work on task A . Another shift schedule might represent two hours of work on task A ; then the changeover to task B takes place and B is worked on for the remainder of the shift. Several feasible shift schedules are presented in Figure 2.1. The letter indicates the task; “Idle” indicates that the workstation is prepared to work on a task, but the workers are currently idle. In production planning, “Idle” may occur when the machine is set up to produce a product, but the machine (and workers) are idle. Δ_{AB} indicates the time to changeover from task A to task B .

By defining the variables in this way, many of the constraints are automatically

enforced — a shift schedule is not defined if it does not satisfy the operational rules associated with changeovers. Additionally, associated with each shift schedule are characteristics — the tasks worked on and for what duration, the number of changeovers, etc. — that can be used in constraints, as well as the objective function. Figures 2.2, 2.3 and 2.4 illustrate the shift schedule parameters used in the formulation.

Each figure includes four shift schedule examples. Each parameter is indexed by the name of the task (A or B in the Figures) and a shift schedule identification number (1, 2, 3 or 4 in the Figures). Figure 2.2 illustrates the parameter q , the quantity (or duration) of a task worked on during the shift schedule. Notice that when a task is “Idle” the quantity is zero (illustrated in shift schedule 2 in the figures). Figure 2.3 illustrates the binary parameters f and l . The f parameter represents the first task and the l task represents the last task in the shift schedule. [The shift schedules considered in this chapter have at most two tasks in a shift. There is a discussion later in the chapter to relax this assumption.] Figure 2.4 illustrates the binary parameter u . The u parameter equals one if the task is worked on for non-zero quantity (or duration).

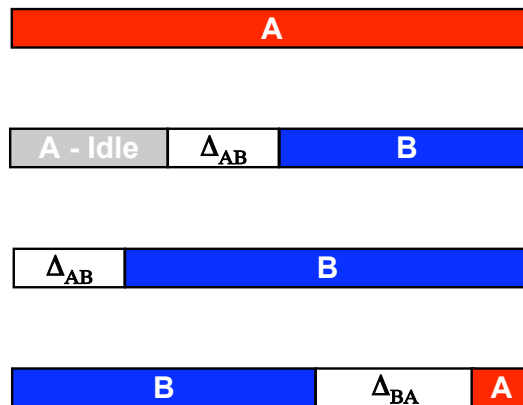


Figure 2.1: Sample Shift Schedules

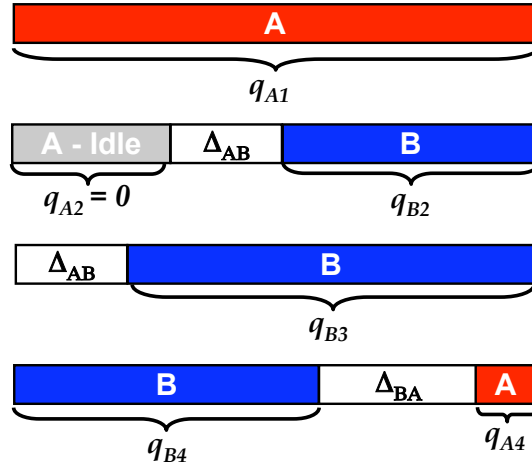


Figure 2.2: Sample Shift Schedules: Quantity Parameter

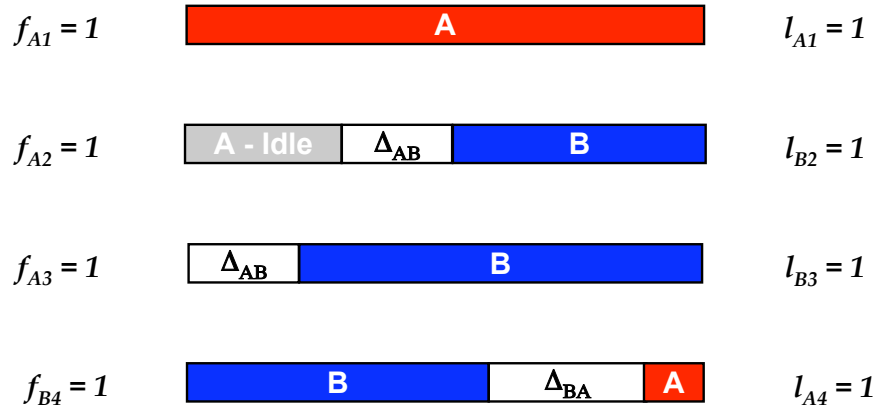


Figure 2.3: Sample Shift Schedules: First and Last Parameter

Unfortunately, using shift schedules to accurately capture the changeover constraints comes at the expense of a (potentially) very large number of variables — one for every feasible shift schedule. This is the case for *CVMs* in general, and is typically addressed through delayed column generation [9], which avoids the explicit enumeration of all variables in advance and instead uses a secondary optimization problem, known as a *pricing problem*, to identify promising candidates (i.e., pivot variables with negative reduced cost).

The shift schedule formulation for *SWA-WU* is presented next, followed by a

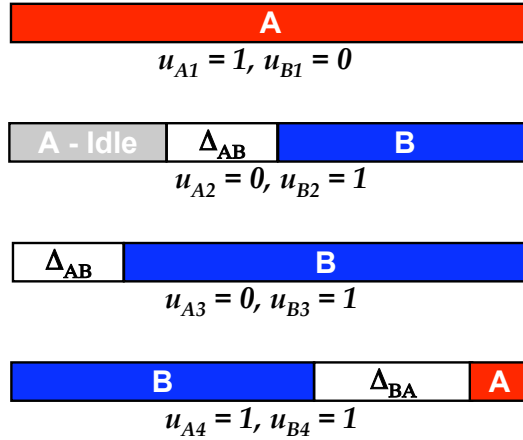


Figure 2.4: Sample Shift Schedules: Non-Zero Quantity Parameter

column generation-based approach for overcoming its large number of variables. This column generation-based approach is used to motivate an extreme shift schedule formulation, which reduces the number of shift schedules that need to be included in the model.

2.2.3 Shift Schedule Formulation

The basis for the formulation is the binary variable x_{ns} which takes value one if shift schedule s is assigned to the n^{th} shift (recall that there is a finite number of shifts in the planning horizon). Given this variable definition, the problem can be formulated as follows.

Sets

- **S** is the set of shift schedules
- **I** is the set of worker types
- **P** is the set of tasks
- **N** is the set of shifts in the horizon

Parameters

- c_n^i is the cost for a type i worker during shift n $\forall i \in \mathbf{I}, n \in \mathbf{N}$
- a_p^i is the number of type i workers required for task p $\forall i \in \mathbf{I}, p \in \mathbf{P}$
- d_p is the demand (i.e., total quantity required), of task p $\forall p \in \mathbf{P}$
- q_{ps} is the duration that task p is worked on in shift schedule s $\forall p \in \mathbf{P}, s \in \mathbf{S}$
(illustrated in Figure 2.2)
- $f_{ps} = 1$ if shift schedule s starts set up to work on task p , 0 otherwise $\forall p \in \mathbf{P}, s \in \mathbf{S}$ (illustrated in Figure 2.3)
- $l_{ps} = 1$ if shift schedule s ends set up to work on task p , 0 otherwise $\forall p \in \mathbf{P}, s \in \mathbf{S}$ (illustrated in Figure 2.3)
- $u_{ps} = 1$ if task p is worked on for a non-zero duration in shift schedule s , 0 otherwise $\forall p \in \mathbf{P}, s \in \mathbf{S}$ (Note that it is possible to have $u_{ps} = 0$ but $f_{ps} = 1$, when the workstation is prepared to work on task p but the workers are idle, this can be observed by comparing the second example in Figures 2.3 and 2.4.)
- K_i is the largest requirement of workers of type i across all tasks
 $(\max_{p \in \mathbf{P}}(a_p^i) \quad \forall i \in \mathbf{I})$

Variables

- $x_{ns} = 1$ if shift schedule s is assigned to shift n , 0 otherwise $\forall n \in \mathbf{N}, s \in \mathbf{S}$
- $y_{nk}^i = 1$ if k workers of type i are required during shift n , 0 otherwise $\forall i \in \mathbf{I}, n \in \mathbf{N}, k \in \{1 \dots K_i\}$
- $w_{np}^F = 1$ if task p is the first task in shift n , 0 otherwise $\forall n \in \mathbf{N}, p \in \mathbf{P}$
- $w_{np}^L = 1$ if task p is the last task in shift n , 0 otherwise $\forall n \in \mathbf{N}, p \in \mathbf{P}$

Shift Schedule Formulation

$$(2.1) \quad \min \sum_{i \in \mathbf{I}} \sum_{n \in \mathbf{N}} c_n^i \sum_{k \in \{1 \dots K_i\}} k y_{nk}^i$$

s.t.

$$(2.2) \quad \sum_{s \in \mathbf{S}} x_{ns} = 1 \quad \forall n \in \mathbf{N}$$

$$(2.3) \quad \sum_{s \in \mathbf{S}} f_{ps} x_{ns} = w_{np}^F \quad \forall p \in \mathbf{P}, n \in \mathbf{N}$$

$$(2.4) \quad \sum_{s \in \mathbf{S}} l_{ps} x_{ns} = w_{np}^L \quad \forall p \in \mathbf{P}, n \in \mathbf{N}$$

$$(2.5) \quad w_{np}^F = w_{(n-1)p}^L \quad \forall p \in \mathbf{P}, n \in \{2 \dots |\mathbf{N}|\}$$

$$(2.6) \quad \sum_{s \in \mathbf{S}} \sum_{n \in \mathbf{N}} q_{ps} x_{ns} = d_p \quad \forall p \in \mathbf{P}$$

$$(2.7) \quad \sum_{s \in \mathbf{S}} u_{ps} x_{ns} \leq \sum_{k \in \{1 \dots K_i\} : k \geq a_p^i} y_{nk}^i \quad \forall n \in \mathbf{N}, i \in \mathbf{I}, p \in \mathbf{P}$$

$$(2.8) \quad \sum_{k \in \{1 \dots K_i\}} y_{nk}^i = 1 \quad \forall i \in \mathbf{I}, n \in \mathbf{N}$$

$$(2.9) \quad x_{ns} \in \{0, 1\} \quad \forall n \in \mathbf{N}, s \in \mathbf{S}$$

$$(2.10) \quad w_{np}^F, w_{np}^L \in \{0, 1\} \quad \forall p \in \mathbf{P}, n \in \mathbf{N}$$

$$(2.11) \quad y_{nk}^i \in \{0, 1\} \quad \forall i \in \mathbf{I}, n \in \mathbf{N}, k \in \{1 \dots K_i\}$$

The objective, 2.1, minimizes the cost of the workforce allocation. Constraints 2.2 assign exactly one shift schedule to each shift. Constraints 2.3 and 2.4 define the

auxiliary variables w_{np}^F and w_{np}^L , which track the task set up on the workstation at the beginning and end of each shift. Constraints 2.5 specify that the task that is set up on the workstation at the end of a shift must be the same task that starts the following shift. Constraints 2.6 ensure that the total quantity (or duration) that each task is worked is equal to the total quantity (or duration) required. Note that the formulation can be easily modified to capture demand due dates throughout the planning horizon, by adding a shift index n to the parameter d_p . [This will be illustrated in Chapter IV]. Constraints 2.7 and 2.8 determine the workforce allocation. These constraints are perhaps best explained with an example. Assume that three workers of type i are required for tasks A or B ($a_A^i = a_B^i = 3$); four workers of type i are required for tasks C or D ($a_C^i = a_D^i = 4$). Then K_i equals four ($\max(3, 4) = 4$). Now consider task B . Any shift where task B is worked on for a non-zero duration (i.e., $\sum_{s \in \mathbf{S}} u_{Bs} x_{ns} = 1$), must have three or more type i workers available – in this case, the possible choices are three or four type i workers (i.e., if $\sum_{s \in \mathbf{S}} u_{Bs} x_{ns} = 1, a_B^i = 3, K_i = 4$, by substitution Constraints 2.7 and 2.8 require that $y_{n3}^i + y_{n4}^i = 1$).

2.2.4 Using Delayed Column Generation to Solve the Formulation

The shift schedule formulation proposed relies heavily on the definition of the set \mathbf{S} , the feasible shift schedules. In theory, there is an infinite number of variables because the set of times that a task can be worked on within a feasible shift schedule is continuous.

One obvious solution, similar to that seen in much of the *CVM* literature, is to limit the number of candidate variables by discretizing time — for example, only considering working on tasks in half-hour increments and then enumerating the corresponding set of valid shift schedules. Even then, however, the number of variables

could be quite large, because of the different durations each task could be worked on during a shift.

Delayed column generation (see, for example, [9]) is commonly applied in such cases. In this approach, only a subset of the variables (in our case, the candidate shift schedules) are initially included in the model, which is referred to as the *restricted master (RM)* problem. The *LP* relaxation of the *RM* is solved to optimality and yields a set of dual values. These duals are then used as input to a secondary optimization problem, known as the *pricing problem*, which seeks the most negative reduced-cost variable from amongst the set of all variables (here, shift schedules).

If a variable with strictly negative reduced cost is found, then this variable is added to *RM* and the process repeats. If the smallest reduced-cost over all variables is nonnegative, then clearly none of the variables outside of *RM* have a negative reduced cost, and thus optimality is established. In an integer program, this process must be repeated at each node of the *branch-and-bound* tree, and the dual variables corresponding to the branching rules must be incorporated into the pricing problems as well.

The challenge is in formulating a pricing problem that can quickly identify promising new shift schedules. Thus, for *SWA-WU* this pricing problem must capture the changeover rules. Unfortunately, this difficulty is exactly what motivated the use a *CVM* approach. During our research, we could not find a tractable formulation for identifying the smallest reduced cost shift schedule, especially given that the pricing problem needs to be solved at each iteration (i.e., simplex pivot), of each node in the branch-and-bound tree.

2.2.5 Extreme Shift Schedule Formulation

As an alternative to discretizing time or a column generation approach, we develop extreme shift schedules. To derive the extreme shift schedules we analyze subsets of shift schedules.

For example, consider the set of shift schedules that include a single task, A , and do not include any changeovers. The only decision remaining is the duration that that task is worked on during this shift and how much to leave idle. For the sake of exposition, assume that every shift is eight hours long. A polyhedron defined by the constraints $t(A) \geq 0$ and $t(A) \leq 8$ can represent the candidate set of these shift schedules, where $t(A)$ is the amount of time task A is worked on.

Now consider two-task shift schedules (those with one changeover). For two tasks A followed by B , the set of valid shift schedules is restricted to those cases that satisfy the following constraints:

$$(2.12) \quad t(A) + t(B) \leq 8 - \Delta_{AB}$$

$$(2.13) \quad 0 \leq t(A)$$

$$(2.14) \quad 0 \leq t(B)$$

where $t(p)$ is the amount of time task p is worked on, and $\Delta_{p_1 p_2}$ is the time to changeover from task p_1 to task p_2 . Constraint 2.12 states that the total time working cannot exceed the length of the shift minus the time required for the changeover (recall the assumption that there are eight-hour shifts). Constraints 2.13 and 2.14 state that the duration of work on a task (A and B) can not be negative. [We assume

that Δ_{AB} is non-negative.] These constraints form a polyhedron that defines the set of all feasible shift schedules containing task A followed by task B .

Instead of iterating between the pricing problem and the master problem or discretizing time, we can define *extreme shift schedules* that can be directly included in the shift schedule formulation in section 2.2.3. The extreme shift schedules are the *basic feasible solutions* of the polyhedra mentioned above (one polyhedron for each individual task and one polyhedron for each ordered pair of tasks). Thus, each feasible shift schedule can be written as the convex combination of the extreme shift schedules.

For example, any single task shift schedule can be formed as the convex combination of two extreme shift schedules (the basic feasible solutions of the single task polyhedron). There is one extreme shift schedule in which A is worked on for eight hours ($t(A) = 8$) and one in which the workers are prepared to work on A but remain idle for the full eight hours ($t(A) = 0$). Figure 2.5 shows the feasible region for shift schedules where there is only one task (task A) and no changeovers. In addition the figure shows a shift schedule created by the a convex combination of the two extreme shift schedules, a shift schedule where the workers work on task A for four hours is a convex combination of the two extreme shift schedules (i.e., $0.5 \cdot 8 \text{ hours} + 0.5 \cdot 0 \text{ hours} = 4 \text{ hours}$).

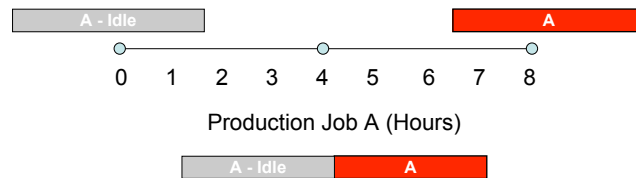


Figure 2.5: Feasible Region: One-Task Shift Schedules

The extreme points of the two-task polyhedron (denoted by $(t(A), t(B))$) are:

$(0, 0)$, $(0, 8 - \Delta_{AB})$, and $(8 - \Delta_{AB}, 0)$. The shift schedules corresponding to these three points are extreme shift schedules. Any convex combination of these three extreme shift schedules will be a feasible shift schedule, and any feasible shift schedule containing task A followed by task B can be represented as a convex combination of these three extreme points. Figure 2.6 shows the feasible region for two-task shift schedules and an example of a convex combination of the extreme shift schedules.

The figure assumes that Δ_{AB} is a half-hour.

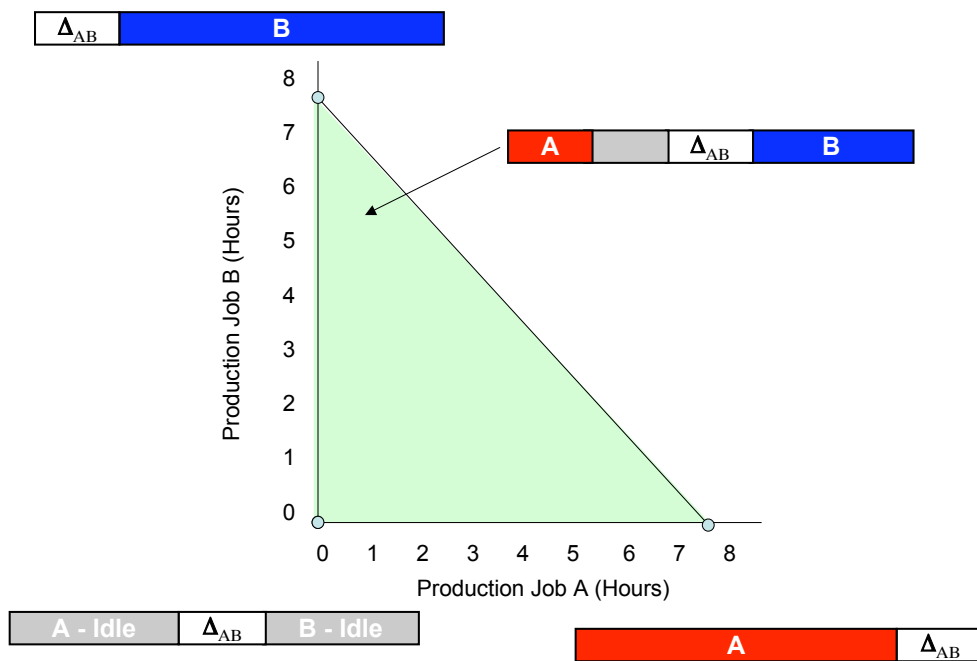


Figure 2.6: Feasible Region: Two-Task Shift Schedule

The fact that any shift schedule for a given task or pair of tasks can be written as a convex combination of the corresponding extreme shift schedules eliminates the need for the pricing problem altogether (see [4] for a similar result). *In addition, this fact can dramatically decrease the number of variables and increase the solution quality by implicitly incorporating all possible shift schedules (without any need for discretizing time).*

This set of extreme shift schedules — the two extreme shift schedules for each

task, as well as the three extreme shift schedules for each ordered pair of tasks — is denoted by \mathbf{S}' . With this set, constraint 2.9

$$x_{ns} \in \{0, 1\} \quad \forall n \in \mathbf{N}, s \in \mathbf{S}$$

can be replaced with the much smaller set of continuous variables

$$(2.15) \quad 0 \leq x_{ns} \leq 1 \quad \forall n \in \mathbf{N}, s \in \mathbf{S}'.$$

This change, in combination with the existing constraints 2.3 through 2.5 (requiring a single task to be at the beginning of each shift and one at the end of each shift), captures all possible shift schedules and ensures an implementable solution. Using the set of extreme shift schedules directly in the model achieves four benefits that are critical in achieving tractability:

1. The number of variables is greatly reduced — for each shift n , there are only two extreme shift schedules per task and three extreme shift schedules for each ordered pair of tasks, as given by the polyhedra presented earlier. Thus, an instance with 10 tasks would have only 290 extreme shift schedules. This is a significant improvement over discretizing time into half-hour increments where over five times the number of shift schedules would be needed. Table 2.1 shows the difference in the number of shift schedules required to represent all feasible shift schedules using the extreme shift schedule versus discretizing time using half-hour and one minute increments.

Table 2.1: Number of Shift Schedules Required Comparison

Tasks	Extreme Shift Schedules	Half-hour Increments	Minute Increments
5	70	405	11,425
10	290	1,610	45,400
15	660	3,615	101,925

2. The x variables become continuous, greatly reducing the number of integer variables in the model and the corresponding amount of branching needed to solve it.
3. This formulation can be solved directly, without delayed column generation. This bypasses the tailing effects commonly seen in such approaches, as well as the negative impact that the branching rules can have on the solvability of the pricing problems.
4. The solution set is exhaustive. Time does not have to be discretized to ensure tractability.

Chapter IV illustrates how this concept can be extended when there are more complex operational constraints.

For the problem considered in this dissertation there is at most one changeover in a single shift; however the logic naturally extends to shifts in which more changeovers are permitted. For example, to consider a shift schedule with two changeovers, a third set of auxiliary variables would need to be created, w_{np}^M , which would take value 1 if task p was the *middle* task set in shift n , and corresponding constraints. Creating another three-task polyhedron by expanding constraints 2.12 — 2.14 would create extreme shift schedules for the set of tasks $\{A, B, C\}$ to include the minimum time required for setting and producing the middle task. [Appendix C presents the constraints and extreme shift schedules corresponding to three-task shift schedules.]

We conclude by noting that when the number of changeovers allowed per shift grows, even the set of extreme shift schedules may be too large to include directly in the model. In this case, delayed column generation could be used, not to identify the specific shift schedules, but rather to identify promising extreme shift schedules,

which would then be added to RM (not as binary variables, but rather as continuous variables from which convex combinations could be formed to construct solution).

2.2.6 Computational Results

To evaluate the performance of this model, twenty-two *stamping pressline* instances (i.e., twenty-two different data sets), provided from a major automotive manufacturer were considered. [This problem will be discussed in greater detail in Chapter IV.]

For each of these instances, the extreme shift schedule formulation was implemented using CPLEX version 11.0 with default branching parameters on a IBM x3455 with an 2x Dual Core AMD Opteron Model 2218 Processors at 2.6GHz (2593.632 MHz), 10 GB of memory running Redhat Enterprise Linux 4 operating system. The optimality gap was set to zero percent and a time limit of 10.5 hours was used to evaluate each instance. The results appear in table 4.3. This table provides the run time in seconds, the number of branch-and-bound nodes solved, and the optimality gap at completion.

In nineteen of the twenty-two instances, a provably-optimal solution was found in under 3 hours and 45 minutes. This was not the case in the remaining three instances where the optimality gap was between 0.49% and 9.68% after 10.5 hours. We discuss why these instances were intractable and present a new algorithm, *Test-and-Prune*, to overcome that source of intractability in the next chapter.

2.3 Conclusions

In this chapter, we presented a composite variable model that incorporates significant real-world detail without the need for a large number of constraints to model $SWA-WU$. By defining shift schedules, variables that represent feasible schedules

Table 2.2: Single Pressline Initial Results

ID	Run Time (sec)	Number of Nodes	Optimality Gap (%)
1	2	523	0.00
2	407	3297	0.00
3	23	350	0.00
4	965	399	0.00
5	2012	1116	0.00
6	4	553	0.00
7	45	259	0.00
8	1	508	0.00
9	37800	577587	9.68
10	1385	1989	0.00
11	102	548	0.00
12	84	81	0.00
13	37800	76057	0.49
14	4076	35249	0.00
15	5258	42370	0.00
16	2	70	0.00
17	230	1273	0.00
18	2394	21551	0.00
19	13219	503858	0.00
20	5101	227024	0.00
21	9	185	0.00
22	37800	26963	8.18

for an individual shift, constraints that capture complex operational policies about changeovers are not needed. The sequence of tasks, duration of time that each task is worked on, changeover between tasks, and number of workers needed in a shift are all represented by a single variable. Furthermore, this approach bypasses many of the limiting assumptions often seen in the literature — there are no limits on the size of the batches, or the number of workers available, and the number of changeovers is not fixed in advance. Additionally, the use of composite variables makes it possible to assume sequence-dependent changeovers.

The shift schedule formulation can be used to easily evaluate many types of scenarios with limited modification. Examples of different questions that could be answered using the shift schedule formulation are:

- “Can we meet demand using a subset of shifts?” This question can be answered

by reducing the number of shifts (i.e., the cardinality of the set \mathbf{N}), considered in the formulation.

- “Can we meet demand if we work on task A only during the first three shifts?” This question can be answered by excluding all shift schedules containing task A from consideration, except on shift one, two and three.
- “What is the cost if we sequence the tasks in decreasing order of demand?” This question can be answered by ranking tasks in decreasing order of demand and excluding all shift schedules with changeovers except those in which the pair of tasks are sequential in the list.
- “What if we allow at most ten changeovers in the planning horizon?” This question can be addressed by including a parameter that states the number of changeovers in each shift schedule and adding a simple constraint summing the changeovers in all chosen shift schedules.

Many other scenarios can be considered as well.

Although composite variable modeling can be quite powerful, it often comes at the expense of a very large number of variables. In the shift schedule model, however, the number of variables is quite contained. The problem structure is exploited by recognizing that any shift schedule can be written as the convex combination of a small number of extreme shift schedules. This not only controls the number of variables, but also enables the integrality of these variables to be relaxed. In the process, the quality of the solution is actually improved, because time is not discretized and the set of feasible shift schedules is not restricted.

The shift schedule and extreme shift schedule formulations were developed to target very specific operational considerations (in particular, the changeover process

and the cost structure in which workers must be paid for all days in the planning horizon), however the shift level may also be the source of many complexities in other workforce allocation and utilization problems. While the details associated with defining the feasible set of shifts and the use of added constraints in the model may vary, the general structure may be applicable in other such applications.

CHAPTER III

Test-and-Prune (T&P)

Chapter II described a formulation based on shift schedule variables for *Shift-Workforce Allocation and Utilization* problems, and presented computational results for twenty-two instances provided by a major automotive manufacturer. For each of the twenty-two instances, the extreme shift schedule formulation was implemented using CPLEX 11.0 and was given a time limit of 10.5 hours. Provably optimal solutions were found for nineteen of the twenty-two instances in under 3 hours and 45 minutes. However, for three of the twenty-two instances the integer programming solver was unable to find provably optimal solutions within 10.5 hours.

This chapter presents an alternative approach to branch-and-bound, *Test-and-Prune*, for efficiently solving *SWA-WU*. The chapter is organized as follows. It begins with a discussion of a source of the intractability, a weak *linear programming* relaxation. Section 3.2 describes the motivation for the *T&P* algorithm. Section 3.3 formally states *T&P*. Section 3.4 discusses ways to improve the performance of the *T&P* algorithm. Section 3.5 discusses how *T&P* can be enhanced so that it can be applied to resource allocation and utilization problems that are not *SWA-WU*. Section 3.6 compares *T&P* to Bender's Decomposition.

The key contribution of this chapter is in developing a tractable solution approach

for *SWA-WU*. It is also important to note that *T&P* has applicability beyond the *SWA-WU* for a much broader class of problems – those in which a discrete and finite set of resource allocation decisions dominate cost while a more substantial set of resource utilization decisions dominate complexity. Section 3.7 further discusses this contribution and motivates the next chapter.

3.1 Weak Linear Programming Relaxations

A *LP* relaxation of an integer program is the result of replacing the integrality constraints by relaxed constraints that allow each integer variable to be continuous. (e.g., when considering binary variables, the constraints that each variable must be *equal to* 0 or 1 are relaxed and variables are allowed to be any value *between* 0 and 1).

Optimal solutions to integer programs are often found by solving a series of related *LP* relaxations in an approach referred to as branch-and-bound [62]. When the feasible region of a *LP* relaxation is not similar to the convex hull of the feasible set of solutions to the integer program, the *LP* relaxation is said to be weak. Due to a weak *LP* relaxation, a significant number iterations of the branch-and-bound algorithm will be required to converge to an integer optimal solution. Thus, weak *LP* relaxations lead to large branch-and-bound trees and long computation times. Weak *LP* relaxations are problematic throughout many integrated workforce allocation and workforce utilization problems, including *SWA-WU*.

In *SWA-WU*, the workforce allocation decisions (i.e., how many workers to hire and what skills each worker should have) and workforce utilization decisions (i.e., what tasks to assign each worker) are being made simultaneously. Unfortunately, the extreme shift schedule formulation (discussed in detail in Chapter II) has a

weak LP relaxation as a result of considering these decisions at the same time. As illustrated in Chapter II Table 4.4, the weak LP relaxation of the extreme shift schedule formulation contributed to large branch-and-bound trees and long run times for several of the instances.

The following simplified example helps to demonstrate why the extreme shift schedule formulation has a weak LP relaxation. Recall the notation discussed in Chapter II where, $w_{np}^F = 1$, if task p is the first task in shift n , 0 otherwise $\forall n \in \mathbf{N}, p \in \mathbf{P}$ and $w_{np}^L = 1$, if task p is the last task in shift n , 0 otherwise $\forall n \in \mathbf{N}, p \in \mathbf{P}$. By setting $w_{np}^F = w_{np}^L = \frac{1}{P}$ for all shifts n and all tasks p , a fractional solution can be constructed that schedules no changeovers (and does not incur any of the cost for the workers responsible for changeovers). This yields a weak lower bound on the optimal cost. The objective value of the LP relaxation is at least as good as the objective value of the integer program, because any feasible solution to the integer program solution would also be a feasible solution to the LP relaxation. Furthermore, when branching on one of the w variables, which determine the sequence of tasks, there is limited impact on the remainder of the solution. In this problem when one decision regarding the sequence of tasks is made, it does not restrict remaining sequence decisions, unlike some integer programs, in which fixing one integer variable automatically leads to integer values for many other variables.

Thus, many branches must be evaluated before an integer feasible solution is found. Even once an integer feasible solution is found, many more branches would be required to prove that a lower-cost solution does not exist. This weak LP relaxation makes solving the extreme shift schedule formulation a challenge for some instances.

3.2 T&P Motivation

Consider the following two key observations.

1. Number of possible workforce allocations is finite. Although there is possibly an infinite number of distinct feasible workforce utilization solutions (i.e., feasible sequence and duration of tasks), the number of unique workforce allocations is finite and often quite small. Several feasible workforce utilization solutions correspond to one unique workforce allocation.
2. When the allocation is fixed, the remaining problem is a feasibility problem.

The objective for *SWA-WU* only considers the workforce costs; when the workforce allocation, captured by the y variables in the extreme shift schedule formulation, are fixed a feasibility problem remains. Computational experiments for the stamping case studies show that the resulting feasibility problem is typically quite easy to solve.

A new algorithm, *T&P*, was motivated by these two observations. *T&P* solves several feasibility problems instead of solving a single optimization problem. Each feasibility problem seeks to answer the question, “Does a feasible workforce utilization exist for the given workforce allocation?” The optimal solution to the problem is the feasible workforce allocation with lowest-cost.

3.3 T&P

Our approach to overcome the weak *LP* relaxation is based on the simple idea of solving many easy feasibility problems instead of one difficult optimization problem. We begin by enumerating all feasible *shift-workforce allocation* solutions. We then compute the cost of each of these solutions. Next, for each *SWA* solution we test

the corresponding *workforce utilization* problem for feasibility. A *SWA* solution is considered feasible if the corresponding instance of the *WU* problem is feasible. Finally, we select the lowest-cost *SWA* solution that corresponds to a feasible *WU*.

To improve the run time of our algorithm, we prune the list, limiting the number of *SWA* solutions that are actually tested. In particular, we take advantage of the fact that we can prune both when a *SWA* is feasible (eliminating all higher-cost solutions) and also when it is infeasible (eliminating all dominated solutions).

By breaking the optimization problem into a series of feasibility problems $T\mathcal{E}P$ overcomes the extreme shift schedule formulation’s weak *LP* relaxation. The objective is not explicitly included in the feasibility problems and thus the incentive to assign fractional workers is significantly reduced. This, together with pre-processing and pruning, results in $T\mathcal{E}P$ efficiently solving all of the twenty-two instances of *SWA-WU* previously discussed as well as additional instances of *SWA-WU* that are described in Chapter IV. The details of the algorithm are outlined in the following section.

3.3.1 T&P Algorithm Details

In addition to the notation defined for the extreme shift schedule formulation in Chapter II, we use the following notation to formalize $T\mathcal{E}P$.

- The set \mathcal{Y} is the set of all possible *SWA* solutions – the y variables in the extreme shift schedule formulation. Note that \mathcal{Y} is not required to be represented as a mathematical formulation.
- The set \mathcal{X} defines any non-negativity constraints, integrality constraints, and bounds on the *WU* decisions – the x (shift schedule) and w (first task and last task) variables in the extreme shift schedule formulation. Recall that the *WU*

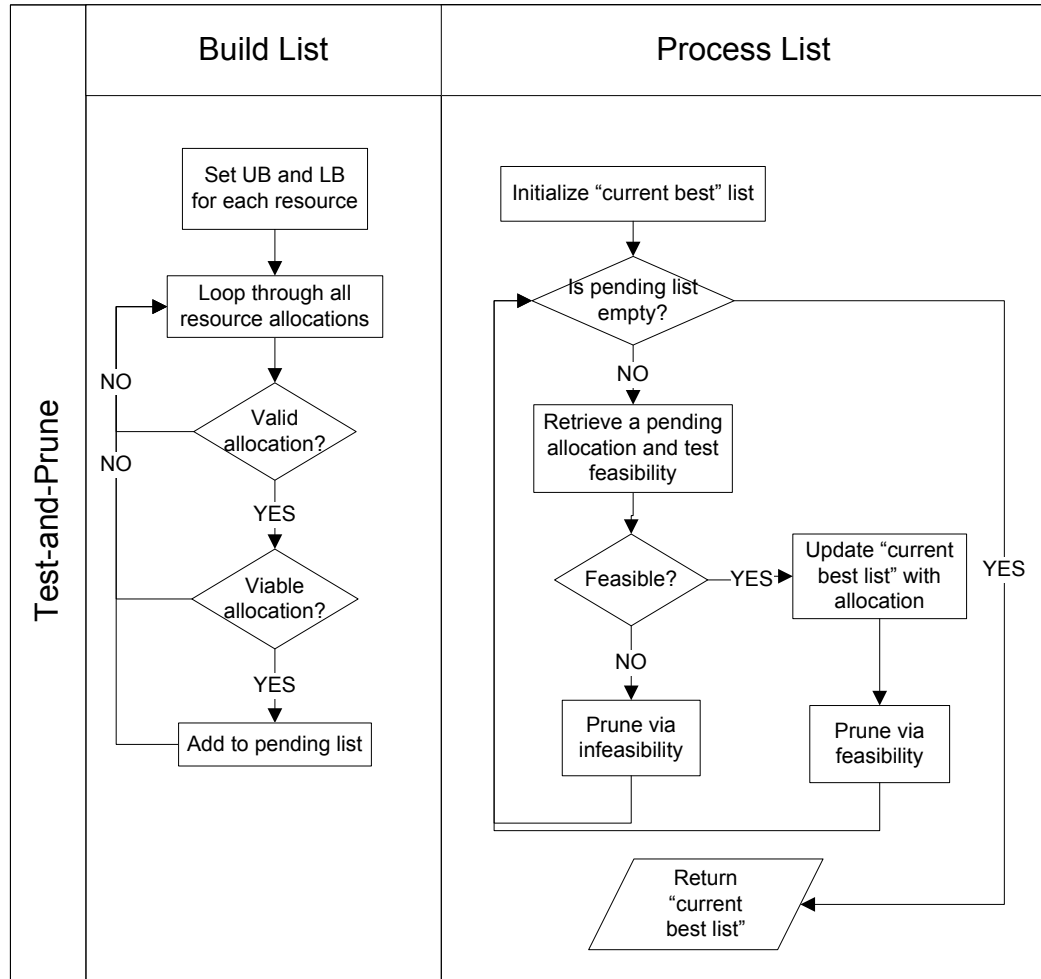


Figure 3.1: Test-and-Prune Flowchart

decisions describe the utilization of the workforce, the sequence and duration of tasks in the planning horizon.

- $f(y)$ is the workforce cost function applied to SWA solution $y \in \mathcal{Y}$.

There are two phases to the $T\&P$ algorithm. First, we *build* the list of candidate SWA solutions; we then *process* this list until it is empty and the optimal solution has been found. These phases are outlined in Figure 3.1.

Build List Phase

In the Build phase, we create a *Pending List* of candidate solutions to the *SWA* problem. Specifically, we begin with an upper and lower bound for each variable y . We loop through all combinations in these ranges and for each such vector y , we check its *validity* – that is, we test to see if it is a member of the set \mathcal{Y} . Validity tests enforce that a *SWA* solution is possible. A validity test might contain a total budgetary constraint or a limit on the total number of workers that can be scheduled across all shifts.

If y is not in \mathcal{Y} then we delete it. Otherwise, we next test its *viability* – that is, we use bounds and other problem-specific structures to check if y is clearly infeasible or sub-optimal. For example, any workforce allocation that does not include any workers is clearly infeasible (without workers tasks cannot be completed). If an *SWA* solution is not viable, then we delete it. Otherwise, we insert it in the Pending List.

Process List Phase

In the Process phase, we evaluate elements of the Pending List until the list is empty and the optimal solution has been identified. Specifically, we begin by removing a candidate solution y from the Pending List and testing its *WU* feasibility. If a feasible solution to *WU* (i.e., $x \in \mathcal{X}$), exists for the given y vector, we prune from the list any pending *SWA* solution outside of the optimality gap defined by this solution; clearly any such a solution is sub-optimal. We then update the Current Best list with solution y . Conversely, if y is infeasible, then we prune from the Pending List any *SWA* solution \hat{y} for which there are the same or fewer number of workers for each shift and each worker type (i.e., $\hat{y} \leq y$). Such *SWA* solutions constrain *WU*

even more tightly and thus will also be infeasible.

Once y has been processed and the Pending List updated, we then choose another *SWA* solution vector from the (reduced) Pending List and repeat until the list is empty.

The Current Best allocation at the end of the algorithm is the optimal solution.

3.4 Performance Improvements for T&P

There are three ways to improve the performance of *T&P* for *SWA-WU*:

1. The number of feasibility problems solved can be significantly reduced as a result of pre-processing and pruning. In the Build phase, pre-processing out invalid and non-viable workforce allocations (i.e., *SWA* solutions) can drastically reduce the number of workforce allocations that are included in the Pending List. The smaller the Pending List is the fewer feasibility problems that need to be solved. Pruning the Pending List based on the result of the feasibility problem further reduces the number of allocations whose feasibility problems are solved. The combination of pre-processing and pruning improves the run time of the algorithm – the fewer feasibility problem solved the faster the algorithm will converge.
2. The size of the feasibility problems can be reduced by recognizing workforce allocation decisions that limit workforce utilization decisions. For example, if the workforce allocation states that there are three workers of type one available during shift one, then any shift schedule that includes a task that requires more than three workers of type one cannot be assigned to shift one. Instead of explicitly enforcing the workforce allocation within the extreme shift schedule formulation (by fixing the y vector retrieved from the Pending List), we can re-

duce the number of variables and constraints considered by implicitly enforcing the workforce allocation. Specifically, instead of enforcing workforce allocation relationships with constraints, we simply do not generate utilization variables that would violate the workforce allocation. Therefore, when the workforce allocation states that there are three workers of type one available during shift one, the variables that would require more than three workers of type one during shift one are not generated. This reduces the number of shift schedules considered as well as eliminate the need for constraints to enforce the workforce allocation.

3. The *T&P* algorithm allows customization by the user in a number of areas.

First, the user can develop a “black box” for the validity and viability checks, which take as input a vector y and specify whether this vector is valid and viable.

Second, the user can provide a method for testing the feasibility of the *WU* problem given a fixed set of *SWA* decisions y . We recognize that in different workforce planning environments there may be better alternatives to the extreme shift schedule formulation.

Finally, the user can specify the strategy for retrieving from the Pending List. For example, in the instances discussed in this dissertation the Pending List is sorted by cost and the workforce allocations are retrieved from the middle of the list. As a result, whenever the candidate y vector is feasible, the list will be pruned in half (because half of the candidates in the Pending List, by definition, have higher cost). Whenever the candidate y vector is infeasible, the list is not automatically cut in half, but in the early iterations several allocations

in the Pending List will be pruned by dominance, because the cost function is increasing in the values of y (i.e., more resources means more cost). We recognize this strategy will not yield the best run-time performance in all cases. Just as branch-and-bound requires careful selection of branching and variable selection strategies, $T\mathcal{E}P$ users can benefit from customizing their search strategy to fit the application at hand.

3.5 T&P Enhancements

In addition to the user-customization discussed above, there are several other enhancements to $T\mathcal{E}P$ that allow the algorithm to be applicable to problems that are not *SWA-WU*.

3.5.1 T&P and a Non-Linear Cost Function

In *SWA-WU*, it was assumed that the cost function is restricted to be linear with respect to the *SWA* decisions. However, there are some cases where the cost may not be linear. These cases can be easily accommodated by $T\mathcal{E}P$. In $T\mathcal{E}P$, the objective value is computed for each workforce allocation vector y and is only used to compare the Current Best objective value to the objective value of the workforce allocations in the Pending List when pruning. Therefore, the only restrictions on the cost function when using $T\mathcal{E}P$ is that it needs to be easy to compute. $T\mathcal{E}P$ can easily accommodate several types of objective functions including those that are non-linear, non-convex, and not closed-form.

3.5.2 T&P and Feasibility Problems that are not Mathematical Programs

For *SWA-WU*, the feasibility test within $T\mathcal{E}P$ is a *MIP*. However within $T\mathcal{E}P$, it is not necessary to solve the feasibility problem as an *MIP* or even as a mathematical program – there can simply be a “black box” which returns the status of the y vector

as either feasible or infeasible (i.e., there exists an $x \in \mathcal{X}$ given y). [5] discusses a related approach; this paper addresses a call center staffing problem with *SWA-WU*-type structure, in which feasibility of the y vector is assessed by Monte-Carlo simulations of the call center.

3.5.3 T&P Neighborhood when the Workforce Utilization Decisions Impact Cost

In *SWA-WU*, it was assumed that the task-sequencing decisions are dominated by workforce allocation decisions. However, there are some cases where *WU* will impact cost relative to the *SWA* decisions. For example, there could be inventory costs to store finished goods or penalties for finishing a task after its due date. In this case, *T&P* can be modified by setting the optimality gap to a small *negative* value. By doing so, whenever a feasible solution is found, only those solutions that are *strictly higher* than the optimal value by this gap will be pruned. In other words, all feasible solutions within a neighborhood, a narrow range of optimality, can be found. Given this set of (near-) optimal solutions to *SWA*, the user can then solve *WU* as an optimization problem (i.e., considering the cost function of utilization decisions) for each of these solutions y to find the best combination. [Note that this presupposes that *WU* can be solved easily not only as a feasibility problem, but as an optimization problem as well.]

3.6 T&P and Benders Decomposition

There are some similarities between our proposed approach and Benders Decomposition and therefore it is worth comparing the approaches and putting *T&P* in the context of Benders.

Specifically, the main idea in Benders Decomposition ([11], [46]) is to partition the variables in a mathematical program into two sets, \mathbf{Y} and \mathbf{X} . A relaxed master

problem is solved in which only the set \mathbf{Y} is (directly) considered. An optimal $y \in \mathbf{Y}$ for the restricted master is found and then used as the input to a sub-problem, in \mathbf{X} , which either proves the optimality of y (and finds a corresponding optimal $x \in \mathbf{X}$) or provides a cut for the master problem (to eliminate the current y solution). The master is then solved for a new “optimal” value of y and the process repeats until the sub-problem proves the current y to be optimal (or proves the original problem infeasible).

In “traditional” Benders, given a fixed value of y , the sub-problem is assumed to become a linear program in \mathbf{X} . The cuts can then be generated based on solving the dual of this sub-problem. This approach is successful for a very wide range of applications ([13]), but also has limitations. For example, if the sub-problem is not an *LP* and is an integer program, then there can be a duality gap and, as a result, the optimal solution may not be found. This limitation can be overcome by embedding Benders in each node of a branch-and-bound tree, but this can provide sizeable challenges for both implementation and run time. In addition to these challenges related to the sub-problem, it is also assumed that optimizing in \mathbf{Y} in the master problem is straightforward, which again places limitations on the cost function and constraints associated with \mathbf{Y} .

More general implementations of Benders have thus been developed which target the easing of these restrictions. For example, the logic-based dual idea of Hooker and Ottoson ([55]) defines a more general notion of duality which enables cuts to be generated, and the algorithm to achieve an optimal solution, without requiring the sub-problem to be a linear program. The form of the sub-problem instead becomes application-specific, focused on finding a way to eliminate a large set of elements within \mathbf{Y} based on the information learned from considering just one - for example

see ([25], [76]).

There are many ways in which $T\mathcal{E}P$ is similar to Benders. First and foremost, it also partitions the variables (in our case, the \mathbf{Y} 's are the shift-resource allocation variables and the \mathbf{X} 's are the workforce utilization variables). At each iteration, it also identifies a candidate value y , solves a sub-problem over \mathbf{X} , parameterized by this y , and then uses the information to further restrict the feasible solutions in \mathbf{Y} . And like the logic-based approach to Benders, $T\mathcal{E}P$ requires an application-specific sub-problem to be developed by the user – it is not automatically defined (unlike the case in traditional Benders).

Note, however, that in logic-based Benders the application-specific challenge is to find a way to generate the duality cuts. In $T\mathcal{E}P$, the definition of the cuts is application-independent (i.e., either dominance- or cost- based, depending on whether the current y is feasible or infeasible), and the application-specific component is simply to evaluate whether there exists a feasible solution to \mathbf{X} given a fixed value for y . Note that this feasibility problem need not be an LP or even a mathematical program, but simply a black box that returns “Feasible” or “Infeasible.”

The other key way in which our approach is different from logic-based Benders is in the way in which we select y . In Benders, y is found by solving an optimization problem, with the cuts being used to increasingly reduce the feasible region of this relaxed master problem. We select y in a very different way. Specifically, we enumerate \mathbf{Y} (using some preliminary pre-processing measures to limit the set; recall that we have pre-supposed that \mathbf{Y} is discrete and “somewhat” limited) and then choose an element from this pending candidate list according to a user-specified sorting and selection rule (in our case studies, we sort by cost and select from the middle of the

Pending List).

At first glance, this may seem like a small difference. In the right context, however, it can be quite powerful. For example, we do not have to restrict any properties of the cost function or constraints on \mathbf{Y} , so long as they can quickly be evaluated in a “black box.” Furthermore, so long as there is a way to determine if a feasible $x \in \mathbf{X}$ exists for a given y , the feedback loop from the sub-problem to the master problem is not application-specific but rather well-defined. But the key is, of course, in the actual performance, and we suggest that our approach can in some cases outperform logic-based Benders – not for all problems, of course, but for a well-defined set, which we identify as *SWA-WU*.

In fact, for these problems, one can actually view logic-based Benders as a special case of *T \mathcal{E} P*. Specifically, *T \mathcal{E} P* uses two methods to prune the Pending List: If the current y is infeasible, then all dominated y ’s can be pruned, while if the current y is feasible, all y ’s of higher cost can be pruned. Thus, if the Pending List is sorted by cost and the least-cost element of the Pending List is always chosen (i.e., the “optimal” y is selected), then *T \mathcal{E} P* is in fact logic-based Benders, where the idea of dominance is used to define the dual constraints. In our experience, however, this is far less effective than choosing from the middle of the list, and pruning both through dominance (when y is infeasible) and also through cost (when y is feasible). Because of the problem structure of *SWA-WU*, the Benders approach would start from the bottom of the list and work upwards, with very little impact from each of the successive cuts. In those cases where the optimal solution is fairly costly (relative to the total solution space), the list would be considered almost exhaustively, whereas selecting from the middle is much more likely to behave like a log-based search.

To summarize, *T \mathcal{E} P* is similar to logic-based Benders in its partitioning of the

variables into two sets and its use of non- LP based methods for reducing the feasible region of the y variables; the difference is in how the y space is searched. The $T\mathcal{E}P$ search method is a promising alternative when considering problems that have the structure of $SWA-WU$.

3.7 Conclusions

This chapter began with a discussion regarding weak LP relaxations, and described how a weak LP relaxation can cause intractability for the extreme shift schedule formulation presented in Chapter II. $T\mathcal{E}P$ was created to address this tractability challenge.

Two observations motivate $T\mathcal{E}P$.

1. The number of possible workforce allocations is finite.
2. When the workforce allocation is fixed a feasibility problem remains.

Therefore, instead of solving a single optimization problem, $T\mathcal{E}P$ overcomes the weak LP relaxation of the extreme shift schedule formulation by solving several feasibility problems.

We can improve the performance of the $T\mathcal{E}P$ algorithm by recognizing that information gained when solving one feasibility problem can give a priori information about the results for other feasibility problems. This reduces the number of workforce allocations solved through pruning. If the allocation is feasible, any allocation with higher cost can be pruned. Any allocation with higher cost than the feasible allocation is sub-optimal. If the allocation is infeasible, any allocation with fewer or the same number of workers in each shift type can be pruned. Any allocation with the same or fewer number of workers has fewer resources and will also be infeasible.

The framework of $T\mathcal{E}P$ naturally lends itself to a more detailed investigation of a given application's WU feasibility problem. Rather than trying to solve a challenging integrated optimization problem, researchers can, within the $T\mathcal{E}P$ framework, focus on exploiting application-specific structures to test the WU feasibility for a given SWA solution. Likewise, application-specific knowledge can be leveraged in the pre-processing stage (as is the case studies presented in Chapter IV) to reduce the size of the candidate SWA solution list. Thus, this chapter lays the groundwork for a variety of new research in solving WU feasibility problems.

$T\mathcal{E}P$ can naturally be extended to a broader class of problems – specifically, those in which some high level set of (discrete) resources are being allocated (which dominate system cost), while lower level decisions about how to utilize these resources to complete a set of tasks (which dominate system complexity) must be made. In the future, researchers can explore the applicability of $T\mathcal{E}P$ to other problems within this broader category.

CHAPTER IV

Case Studies

Chapter II presented an extreme shift schedule model for Shift Workforce Allocation and Utilization problems. Chapter III presented a novel algorithmic approach, which we call Test-and-Prune, to solve this extreme shift schedule formulation, which is a hybrid of mathematical programming as a mechanism for solving simpler feasibility problems that are then embedded in a larger search-based algorithm. This chapter presents several *SWA-WU* case studies based on problems found in *automotive stamping plants*. This chapter describes an automotive stamping plant and presents a shift schedule formulation for this manufacturing environment. Computational experiments using data provided by a major automotive manufacturer are presented to demonstrate the effectiveness of the $T\mathcal{E}P$ algorithm and compare the $T\mathcal{E}P$ algorithm to traditional *integer programming (IP)* methods.

The chapter is organized as follows. Section 4.1 describes the automotive stamping environment that will be explored in this chapter. Section 4.2 describes the *single pressline* problem, where the workers are assigned to a stamping press throughout the planning horizon. The section concludes with a comparison of the computational results using a commercial *IP* solver to $T\mathcal{E}P$. The remainder of the chapter explores how $T\mathcal{E}P$ and the extreme shift schedule formulation can be used to make workforce

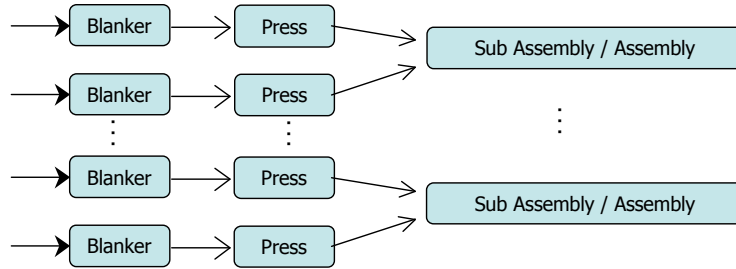
allocation and utilization decisions across presslines. Section 4.3 extends $T\mathcal{E}P$ and the extreme shift schedule formulation to determine the minimum number of shifts types (first, second and third) that a plant must be operating. This section illustrates the improvements that pre-processing and pruning have in $T\mathcal{E}P$. Section 4.4 describes the problem of sharing a pool of workers across groups of presslines called *pressline zones*. This section illustrates how $T\mathcal{E}P$ can be used to establish bounds for a pressline zone problem. Section 4.5 is the conclusion with a summary of the results.

4.1 Stamping Problem

An *automotive stamping plant* is the part of the automotive supply chain that produces the parts that compose the body of vehicles – such as hoods, fenders, and door panels – to be used in the manufacturing and repairing of automobiles. An automotive stamping plant utilizes an *assembly line production system*. An assembly line is a set of sequential workstations in which parts pass from workstation to workstation until the product is fully assembled. There are four types of workstations in a stamping plant: *blanking presses*, *stamping presslines*, *sub-assembly workstations*, and *assembly workstations*. There are part-specific containers that transfer parts between the presses, presslines and workstations. These containers are called *racks*.

The process begins by inserting a large roll of sheet steel into a blanking press. This press cuts the sheet steel into pieces that are slightly larger than the final parts, called *blanks*. The blanks are then passed to a stamping pressline that contains matching upper and lower *dies*. As a blank moves through the stamping pressline, the dies shape the blank into a three-dimensional part. After the stamping pressline, some of these parts are considered final products and are ready to be shipped, while others stay in the plant and move on to sub-assembly and/or assembly workstations.

At these workstations, parts can be combined to create final products or undergo additional operations – such as welding, or adding nuts and bolts – to create final products. The final products are shipped from the stamping facility to downstream facilities and/or customers. Each final product has a pre-defined routing through the plant that must be followed. Figure 4.1 illustrates the flow of parts in a facility.



Example Plant Layout

Figure 4.1: Stamping Facility Flow

This research focuses on creating effective workforce plans for the presslines stage of production. This is the bottleneck operation, with the most binding capacity constraints.

Figure 4.2 depicts a pressline; recall that each part corresponds to a specific set of upper and lower dies. Each pressline is assigned a specific set of part types to produce. The key modeling challenge in this problem stems from the requirements governing the changing of those upper and lower dies.

An *off-line preparation* and *on-line dieset* are required to change the pressline from the production of part type A to that of type B .

First, the dies for part type B must be prepared. This off-line preparation can take place while part type A is being produced, but it cannot begin before production of part type A starts. During off-line preparation, dies from the previous on-line dieset (these dies correspond to the part type produced before part type A) are moved to

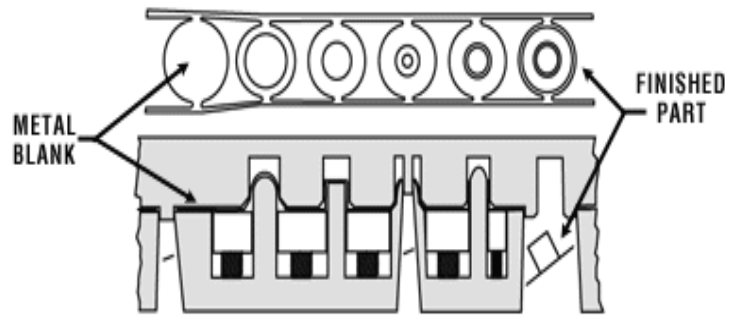


Figure 4.2: Stamping Press [1]

their storage location or cleaning and maintenance area. Then, the dies for part type *B* are picked up and moved to the pressline and the dies are visually checked.

Second, once the off-line preparation (which can take as much as two hours) has been completed and after the production of part type *A* ends, the on-line dieset to the dies for part type *B* may take place. Specifically, once the production has been completed, the area is cleaned and the dies for part type *A* are unclamped or unbolted. The dies for part type *A* are moved out, and the dies for part type *B* are moved in, secured and then inspected. During this period (which, depending on the technology used, can take from as little as 30 minutes to as much as several hours), no parts can be produced on the pressline.

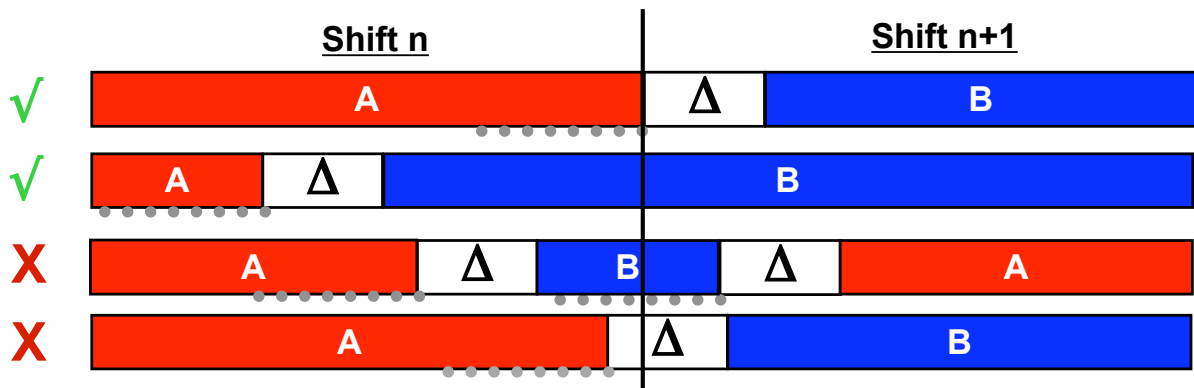


Figure 4.3: Feasible Changeovers and Off-line Preparation

It is important to note two operational policies regarding changing the pressline from the production of part type A to that of part type B . First, the on-line dieset to A should be fully contained within a single shift. Second, the subsequent off-line preparation time for B should also be fully contained in a single shift. This is so a single group of workers maintains full responsibility over each activity. Figure 4.3 illustrates proper and improper off-line preparation and on-line dieset scheduling across two shifts n and $n+1$. The Δ represents an on-line dieset, a letter denotes production of a particular part type and the dots represent an off-line preparation. The “X” denotes an improper on-line dieset or off-line preparation and the check mark denotes a proper on-line dieset and off-line preparation. The top two examples are feasible, both the off-line preparation and on-line dieset are fully contained in a shift. It is important to note that the off-line preparation and the subsequent on-line dieset do not have to be in the same shift. The remaining examples are improper; the off-line preparation or the on-line dieset span across the shifts.

Lastly, it is important to note that the workforce costs dominate all other costs in the facility. There are two types of workers for presslines, *direct laborers* – responsible for operating the presslines – and *indirect labor crews* – responsible for the off-line preparation and conducting the on-line diesets. Each part type requires a specific number of direct laborers to be present during production. One indirect labor crew is required during off-line preparation. Laborers must be hired for the entire planning horizon within a given *shift type* (first, second, or third shift in the day). Therefore, the daily direct labor staffing level for a given shift type is the maximum direct labor requirement across all such shifts in the planning horizon and the daily indirect labor staffing level for a given shift type is the maximum indirect labor requirement across all such shifts in the planning horizon. For each shift type, there is an hourly cost

per direct laborer, as well as a separate hourly cost for indirect labor crews.

This chapter develops algorithms and models to assist managers in determining the workforce allocation (the number of direct laborers and indirect labor crews in each shift type) and utilization (the schedule of part types on each pressline) in a stamping plant.

A schedule for a pressline is defined by the sequence of the part types on the workstations and the duration that each part type is produced on each workstation. Each schedule has a corresponding workforce allocation.

Table 4.1 illustrates a sample schedule for a pressline. This table includes the shift number, the shift type, a figure to represent the tasks completed in the shift, the number of direct laborers required (denoted by D Req.) to complete the task and the number of indirect labor crews required (denoted by I Req.) to complete the task. The last two columns represent the number of direct laborers scheduled in the shift (denoted by D Sched.) and the number of indirect crews scheduled in the shift (denoted by I Sched.). For the shift tasks column the letter indicates the part type; “Idle” indicates that pressline is set up to produce a part type, but the workers are currently idle. Δ_{AB} indicates the time to complete the on-line dieset from part type A to part type B .







In the example, part type A requires three direct laborers for production, part type B requires four direct laborers for production and part type C requires one direct laborer for production. Each dieset requires one indirect labor crew. The planning horizon begins by producing part type C for eight hours in shift one, followed by a completely idle shift (the pressline remains set up for part type C). The third shift begins by producing part type C for one hour. Next, the pressline is set up to produce part type A , and part type A is produced for the rest of the shift. Part

type A is produced for all of shift four. To begin shift five, the pressline is set up to produce part type B , after the dieset is complete part type B is produced for the remainder of the shift. Part type B is produced for a half-hour to begin shift six, then there is a dieset to produce part type C and part type C is produced for the remainder of the shift.

The number of laborers for each shift is the maximum across all tasks scheduled in that shift. Thus, one direct laborer is required in the first shift, zero direct laborers are required in the second shift, three direct laborers are required in the next two shifts and four direct laborers are required in the last two shifts. These values are shown in the direct laborers required (D Req.) column. One indirect labor crew is required in the third, fifth and sixth shifts, this is indicated in the indirect labor crews required (I Req.) column.

With this information we can determine the number of direct laborers and indirect labor crews required for each shift type. Recall, the number of laborers for each shift type is the maximum across all shifts in that shift type. Thus, three direct laborers are required in the first shift type, and four direct laborers are required in the second and third shift type – these values are shown in the direct laborers scheduled (D Sched.) column. One indirect labor crew is required in the second and third shift type – these values are shown the indirect labor crews scheduled (I Sched.) column.

Table 4.1: Sample Pressline Schedule

Shift	Shift Type	Shift Tasks	D Req.	D Sched.	I Req.	I Sched.
1	First		1	$\max(1,3) = 3$	0	$\max(0,0) = 0$
2	Second		0	$\max(0,4) = 4$	0	$\max(0,1) = 1$
3	Third		3	$\max(3,4) = 4$	1	$\max(1,1) = 1$
4	First		3	$\max(1,3) = 3$	0	$\max(0,0) = 0$
5	Second		4	$\max(0,4) = 4$	1	$\max(0,1) = 1$
6	Third		4	$\max(3,4) = 4$	1	$\max(1,1) = 1$

The objective of the pressline workforce planning problem is to minimize the cost of labor subject to the following constraints.

1. There are three eight-hour shifts in each day.
2. Each pressline is assigned a set of part types to produce.
3. All daily demands must be met on time.
4. Production of a given part type can only take place when the pressline is set for that part.
5. The on-line dieset for a given part type cannot occur until the off-line preparation has been completed.
6. Off-line preparation for a pressline cannot begin until the prior on-line dieset on the pressline has been completed.
7. Production for a pressline cannot occur while an on-line dieset is taking place on the pressline.
8. Each on-line dieset must be conducted completely within a single shift.
9. Each off-line preparation must be conducted completely within a single shift.
10. Production cannot occur unless the correct number of direct laborers are present at the pressline.
11. On-line diesets cannot occur unless the correct number of indirect labor crews are present at the pressline.
12. Off-line preparation cannot occur unless the correct number of indirect labor crews are present at the pressline.
13. Laborers must be hired for the entire planning horizon within a given shift type.

4.1.1 Assumptions

The assumptions for the stamping workforce planning problem are as follows:

- All daily demand requirements must be met. The stamping facility feeds the downstream final assembly plants and service facilities. It is essential that these final assemblies not be disrupted due to lack of materials from the stamping facilities.
- Adequate raw materials are always available. It is assumed that the pressline is the bottleneck operation, therefore blanks are always available to feed the pressline.
- Adequate buffer is always available. It is also assumed that there are no capacity limitations for storing completed output from the presslines.
- Daily demand must be met at the end of each day's third shift. This is also the time at which inventory calculations are made.
- Changeover operating policy constraints are enforced. When changing to part type B from part type A , the on-line dieset for part type A must immediately be followed by the beginning of the off-line preparation for part type B . This is to ensure that the pressline can be switched over from A to B as quickly as possible if there is a problem with the production of A . Furthermore, the dieset to A and subsequent off-line preparation time for B should be fully contained within a single shift. This is so a set of laborers maintains responsibility for the complete process.
- At most one changeover can take place per shift. For the sake of exposition, we assume at most one changeover per shift. In many cases this is reasonable

operationally, except when demand levels are very small. Section 2.2.4 notes how this assumption can be relaxed.

- Labor requirements: Laborers must be hired for the entire planning horizon within a given shift type (first, second, third).
- The problem is static, deterministic, and repeating. The goal is to develop a two week schedule, to be repeated over an extended period of time, comprised of at most three shifts per weekday. Although demand may vary from day to day over the two-week horizon, it is assumed that the inventory levels at the start and end of the planning horizon are the same.

The production environment is not fully deterministic – there will be fluctuations in daily demand and yield, as well as machine failures and other sources of disruption. In practice, these issues are addressed through the input data, by establishing a priori appropriate levels of safety stock, being conservative in yield estimates, and leveraging over-time opportunities and the use of premium transportation modes to recover from disruptions [52]. Furthermore, day-to-day modifications can be made by the scheduler or a manufacturing execution system to recover from minor deviations from plan [53]. The purpose of this work is to construct an initial plan which will typically be implemented in conjunction with these systems.

4.1.2 Input Data

The following defines the input data for the pressline stamping problem:

- Time horizon and shifts: In the computational experiments, there are three shifts per weekday for a two-week planning horizon, resulting in a total of thirty shifts.

- Set of part types (also known as jobs): Associated with each part type is a daily demand for each day in the planning horizon (as output by *Manufacturing Resource Planning II (MRP II)*) [84]; the number of direct laborers needed to run that part type on the pressline; the production time per unit; and the daily per-unit inventory cost for that part type.
- Changeover characteristics: The off-line preparation time and on-line dreset time for each pair of part types are known.
- Labor costs: For each shift type (first, second, or third), there is an hourly cost per direct laborer, as well as a separate hourly cost per an indirect labor crew.

4.1.3 Modeling Challenges

The stamping workforce planning problem is a *SWA-WU*. A solution to the automotive stamping problem includes the shift-workforce allocation (the number of direct laborers and indirect labor crews available during each shift in the planning horizon) and the workforce utilization (the schedule of part types on the presses, presslines and workstations). The *SWA-WU* class of problems are discussed in detail in Chapter I. Like other *SWA-WU*, the stamping workforce planning problem is difficult to model because of the complex changeover constraints and non-linear relationships between decisions.

In addition, workers (both direct laborers operating the presslines, and indirect labor crews conducting changeovers) must be hired for the entire planning horizon within a given shift *type*. Thus, the daily staffing level for a given shift type (first, second, or third) is the maximum labor requirement across all such shifts in the planning horizon (recall the example in table 4.1). This adds complexity to the stamping problem.

This chapter shows how an extreme shift schedule formulation and $T\mathcal{E}P$ can be used to overcome these challenges to find high-quality implementable solutions to workforce planning problems in an automotive stamping plant.

4.2 Single Pressline Decisions

The stamping workforce planning problem addressed in this section considers a single pressline.

4.2.1 Problem Statement

In this section, the goal is to build a schedule – what order to sequence the part types in, how many parts to produce during each run – for a single pressline that minimizes the cost of direct laborers and indirect labor crews.

Although there is little if any pre-existing research directly addressing the scheduling of automotive stamping operations, there is a vast body of related research on single-machine scheduling problems. Please see Appendix A for a detailed review on this literature. The application that we consider is different from the standard literature in two ways. The first difference is the objective function, which focuses on labor costs, assuming all demand must be met on or before its due date; in particular, the calculation of labor costs is fairly unique, relative to what is commonly seen in the literature, in that a laborer needed on any given day must be paid for all days in the time horizon. The second difference is in the two-staged changeover requirements, which include both off-line preparation and the on-line dieset itself.

4.2.2 Shift Schedule Formulation

Similar to the formulation presented in Chapter II, the basis for the formulation is the binary variable x_{sn} which takes value one if shift schedule s is assigned to the

n^{th} shift (recall that there are 30 shifts in the problem instances considered - three shifts per weekday for two weeks).

Recall that for *SWA-WU* it is helpful to take advantage of the fact that the complexity is largely shift-specific. For example, changeovers must be fully contained within a shift, and workforce calculations are made at the shift level as well. Therefore, we define variables for this problem to represent a feasible set of ordered tasks that can be completed in an individual shift. Recall that within shift schedule figures, the letter indicates the part type; “Idle” indicates that pressline is set up to produce a part type, but the workers are currently idle. Δ_{AB} indicates the time to complete the on-line dieset from part type A to part type B . The dots represent the time to complete the off-line preparation.

By defining the variables in this way, many of the constraints are automatically enforced — a shift schedule is not defined if it does not satisfy the operational rules associated with changeovers. Additionally, associated with each shift schedule are characteristics — the tasks worked on and for what duration, the number of changeovers (on-line dieset and off-line preparation), etc. — that can be used in constraints, as well as the objective function.

The shift schedule parameters (f , l , q and u) are the same as in the shift schedule formulation in Chapter II, however, there is one additional parameter, g illustrated in Figure 4.4. The figure has four shift schedule examples. The g parameter equals the number of changeovers within each shift schedule. Recall, each parameter is indexed by the name of the part type (A or B in the Figures) and the shift schedule identification number (1, 2, 3 or 4 in the Figures).

In addition to the notation defined in Chapter II we will need to following notation to formulate the single pressline problem.

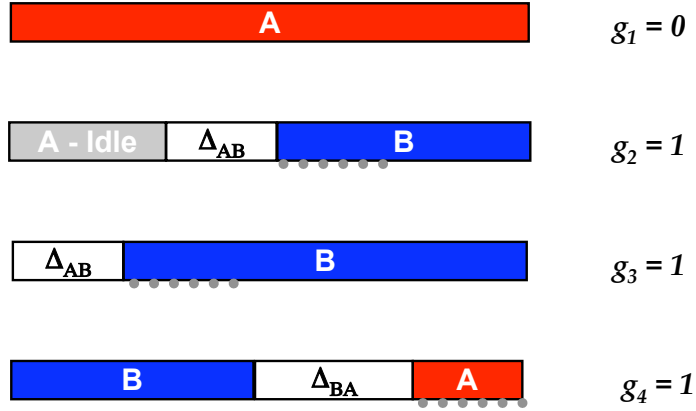


Figure 4.4: Pressline Sample Shift Schedules: Changeover Parameter

Sets

- \mathbf{H} is the set of shift types (i.e., first, second and third)
- $\mathbf{T} \subset \mathbf{N}$ is the set of third shifts – $\{3, 6, 9, \dots, 30\}$ - during which demand must be met and inventory is computed
- \mathbf{S}' is the set of extreme shift schedules for the single pressline problem (discussed in the next section)
- \mathbf{K} is the set of unique direct labor requirements across all part types, plus the option to not staff a shift type (e.g., if part types A and B each require 2 laborers during production and part types C and D require 4, then \mathbf{K} would be $\{0, 2, 4\}$)

Parameters

- $g_s = 1$ if there is a changeover (on-line dieset and off-line preparation) in shift schedule s , else 0 $\forall s \in \mathbf{S}$ (illustrated in Figure 4.4)
- d_{pn} is the demand for part type p on shift n $\forall p \in \mathbf{P}, n \in \mathbf{T}$ (where we only think of demand per day, i.e., for every third shift)

- c_h^I is the cost for the entire planning horizon of an indirect labor crew (responsible for changeovers) in shift type $h \quad \forall h \in \mathbf{H}$
- c_h^D is the cost for the entire planning horizon of a single direct laborer (responsible for production) in shift type $h \quad \forall h \in \mathbf{H}$
- a_p is the number of direct laborers needed during production of part type $p \quad \forall p \in \mathbf{P}$ (in the example, $a_A = a_B = 2$ and $a_C = a_D = 4$)
- $h(n)$ is the shift type of shift $n \quad \forall n \in \mathbf{N}$ (for example, $h(1) = h(4) = h(7) = first$)

Variables

- $i_{np} \geq 0$ is the inventory of part type p at the end of shift $n \quad \forall p \in \mathbf{P}, n \in \{0\} \cup \mathbf{T}$ (again, only once per day, i.e., for every third shift)
- $y_{hk}^D = 1$ if direct labor requirement k is scheduled for shift type h , else 0 $\quad \forall k \in \mathbf{K}, h \in \mathbf{H}$
- $y_h^I = 1$ if an indirect labor crew is required for shift type h , else 0 $\quad \forall h \in \mathbf{H}$

Formulation

$$(4.1) \quad \min \sum_{h \in \mathbf{H}} c_h^I y_h^I + \sum_{h \in \mathbf{H}} c_h^D \sum_{k \in \mathbf{K}} k y_{hk}^D$$

s.t.

$$(4.2) \quad \sum_{s \in \mathbf{S}} x_{ns} = 1 \quad \forall n \in \mathbf{N}$$

$$(4.3) \quad w_{np}^F = \sum_{s \in \mathbf{S}} f_{ps} x_{ns} \quad \forall p \in \mathbf{P}, n \in \mathbf{N}$$

$$(4.4) \quad w_{np}^L = \sum_{s \in \mathbf{S}} l_{ps} x_{ns} \quad \forall p \in \mathbf{P}, n \in \mathbf{N}$$

$$(4.5) \quad w_{np}^F = w_{(n-1)p}^L \quad \forall p \in \mathbf{P}, n \in \{2 \dots |\mathbf{N}|\}$$

$$(4.6) \quad w_{1p}^F = w_{|\mathbf{N}|p}^L \quad \forall p \in \mathbf{P}$$

$$(4.7) \quad \sum_{n'=n-2}^n \sum_{s \in \hat{\mathbf{S}}_m} q_{ps} x_{n's} + i_{(n-3)p} - d_{np} = i_{np} \quad \forall p \in \mathbf{P}, n \in \mathbf{T}$$

$$(4.8) \quad i_{0p} = i_{|\mathbf{N}|p} \quad \forall p \in \mathbf{P}$$

$$(4.9) \quad \sum_{s \in \mathbf{S}} g_s x_{ns} \leq y_{h(n)}^I \quad \forall n \in \mathbf{N}$$

$$(4.10) \quad \sum_{s \in \mathbf{S}} u_{ps} x_{ns} \leq \sum_{k \in \mathbf{K}: k \geq a_p} y_{h(n)k}^D \quad \forall p \in \mathbf{P}, n \in \mathbf{N}$$

$$(4.11) \quad \sum_{k \in \mathbf{K}} y_{hk}^D = 1 \quad \forall h \in \mathbf{H}$$

$$(4.12) \quad 0 \leq x_{ns} \leq 1 \quad \forall n \in \mathbf{N}, s \in \mathbf{S}'$$

$$(4.13) \quad i_{np} \geq 0 \quad \forall p \in \mathbf{P}, n \in \mathbf{T} \cup \{0\}$$

$$(4.14) \quad w_{np}^F, w_{np}^L \in \{0, 1\} \quad \forall p \in \mathbf{P}, n \in \mathbf{N}$$

$$(4.15) \quad y_{hk}^D \in \{0, 1\} \quad \forall k \in \mathbf{K}, h \in \mathbf{H}$$

$$(4.16) \quad y_h^I \in \{0, 1\} \quad \forall h \in \mathbf{H}$$

The objective, 4.1, minimizes the cost of indirect labor crews and direct laborers. Constraints 4.2, 4.3, 4.4, 4.10 and 4.11 are like those found in Chapter II – these constraints assign shift schedules, define the auxiliary variables and calculate the number of direct laborers. Constraints 4.5 and 4.6 specify that the part type for which the pressline is currently set at the end of one shift must also be the part type for which that pressline is set when it begins the next shift (recall the planning horizon is cyclic). Constraints 4.7 and 4.8 calculate each day’s inventory - the production over all three shifts in a day, plus the preceding day’s inventory, minus the current day’s demand, yields the current day’s inventory. These constraints ensure that demand is met on time. Constraints 4.9 enforce that shift type h must be assigned an indirect labor crew if any shift of that type is assigned a schedule containing a off-line preparation.

This formulation is very similar to the formulation presented in Chapter II, there are shift schedule variables, and several sets of constraints are similar. However, there are some differences. First, there are several due dates throughout the horizon for each part type. To capture this the demand parameter, d , has two indices to denote when in the planning horizon the parts must be ready to be sent to customers. To ensure the due dates are satisfied, variables and constraints are introduced to track the daily inventory for each part type. The second difference is that the planning horizon is cyclic. Constraints that make the end of the planning horizon match the start of the planning horizon ensure that the schedules created will be cyclic.

Note that the assumption that starting and ending inventory must be the same can be relaxed by eliminating constraint set 4.14 and instead setting i_{0p} and $i_{|N|p}$ to the desired values (or leaving them as decision variables to be fixed by the model).

4.2.3 Extreme Shift Schedule Formulation

We can derive a set of extreme shift schedules in this formulation in a similar way that we derived the extreme shift schedules for the formulation presented in Chapter II. *Recall that extreme shift schedules dramatically decrease the number of variables and increase the solution quality by implicitly incorporating all possible shift schedules (without any need for discretizing time).*

Similar to the extreme shift schedules presented in Chapter II, we derive the extreme shift schedules for the single pressline problem by considering two different cases – shift schedules that do not include a changeover, and shift schedules that do include a changeover.

The single part type shift schedules are exactly the same. Recall that in the automotive stamping plants considered every shift is eight hours long. A polyhedron defined by the constraints $t(A) \geq 0$ and $t(A) \leq 8$ can represent the candidate set of these shift schedules. Recall, $t(p)$ is the duration task p is produced. The extreme shift schedules are the the basic feasible solutions of the polyhedron $t(A) = 8$ (production of part type A for the entire shift) and $t(A) = 0$ (the workers are prepared to work on A but they remain idle for the entire shift).

The two-part type shift schedules are slightly different. Two-part type shift schedules include a single changeover (on-line dieset and off-line preparation). For two part types A followed by B , the set of valid shift schedules is restricted to those cases that satisfy the following constraints:

$$(4.17) \quad t(A) + t(B) \leq 8 - \Delta_{AB}$$

$$(4.18) \quad t(A) \leq 8 - \Delta_{AB} - \Gamma$$

$$(4.19) \quad 0 \leq t(A)$$

$$(4.20) \quad 0 \leq t(B)$$

where $\Delta_{p_1 p_2}$ is the time to changeover from part type p_1 to part type p_2 and Γ is the duration of the off-line preparation. Constraint 4.17 states that the total production time cannot exceed the length of the shift minus the time required for the on-line dieset (recall the assumption that there are eight-hour shifts). Constraint 4.18 ensures that production of part type A ends with enough time for the changeover (on-line dieset and off-line preparation) to finish within the shift. Constraint 4.19 and 4.20 ensure that the production time of part type A and B is not negative. [We assume that Δ_{AB} and Γ are non-negative and that their sum is less than or equal to the shift duration.]

These constraints form a polyhedron that defines the set of all feasible shift schedules containing part type A followed by part type B . Thus the extreme shift schedules (expressed as $(t(A), t(B))$) are: $(0, 0)$, $(0, 8 - \Delta_{AB})$, $(8 - \Delta_{AB} - \Gamma, \Gamma)$ and $(8 - \Delta_{AB} - \Gamma, 0)$. Any feasible shift schedule containing part type A followed by part type B can be represented as a convex combination of these four extreme points. This polyhedron and an example of a convex combination of the extreme shift schedules are illustrated in Figure 4.5. The figure assumes that Δ_{AB} is a half-hour and Γ is two hours.

Just as in the general *SWA-WU* formulation, the number of variables is greatly reduced by using extreme shift schedules — for each shift n , there are only two extreme shift schedules per part type and four extreme shift schedules for each ordered pair of part types, as given by the polyhedra presented earlier. Thus, an instance with 10 part types would have only 380 extreme shift schedules. This is a significant

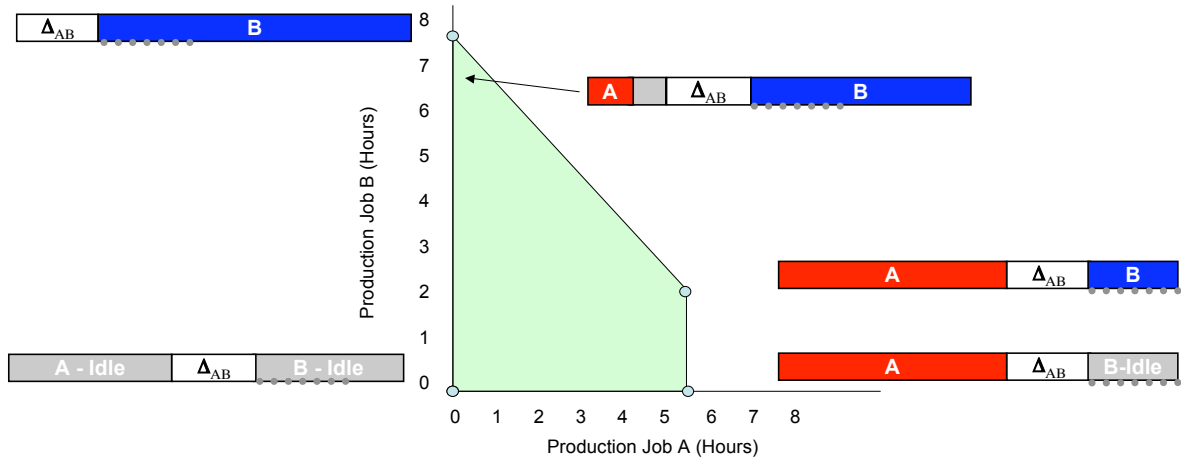


Figure 4.5: Feasible Region: Pressline Two-Task Shift Schedule

improvement over discretizing time into half-hour increments where over *fourteen* times the number of shift schedules would be needed. Table 4.2 shows the difference in the number of shift schedules required to represent all feasible stamping shift schedules using the extreme shift schedule versus discretizing time using half-hour and one minute increments. [Note that due to the additional constraints regarding changeovers, the stamping problem requires more extreme shift schedule than the formulation presented in Chapter II, Table 2.1. However, it is important to also note that the number of discretized shift schedules that would be required has grown at a significantly faster rate.]

Table 4.2: Pressline Number of Shift Schedules Required Comparison

Tasks	Extreme Shift Schedules	Half-hour Increments	Minute Increments
5	90	1,285	803,425
10	380	5,570	3,609,400
15	870	12,855	8,417,925

4.2.4 Computational Results

To evaluate the performance of this model, twenty-two single pressline instances (i.e., twenty-two different pressline data sets) provided from a major automotive

manufacturer were considered.

Branch-and-Bound Computational Results

Table 4.3 provides the number of distinct part types and total number of individual parts to be produced, as well as the number of unique direct labor requirements, $|\mathbf{K}|$, for each pressline.

Table 4.3: Single Pressline Characteristics

ID	# Part Types	Total	# of Units	Unique Direct Labor Requirements
1	2		18,290	3
2	5		52,910	3
3	5		37,780	4
4	12		38,190	2
5	11		38,750	2
6	3		35,710	4
7	7		35,350	3
8	2		43,180	2
9	5		39,440	3
10	9		56,540	5
11	7		85,860	3
12	7		29,900	3
13	8		44,170	3
14	5		94,360	4
15	6		34,730	4
16	3		21,600	3
17	5		36,760	4
18	5		30,730	2
19	4		40,250	4
20	3		21,980	2
21	3		28,850	3
22	13		158,110	4

For each of these instances, the extreme shift schedule formulation was implemented using CPLEX version 11.0 with default branching parameters on a IBM x3455 with an 2x Dual Core AMD Opteron Model 2218 Processors at 2.6GHz (2593.632 MHz), 10 GB of memory running Redhat Enterprise Linux 4 operating system. A time limit of 10.5 hours was used to evaluate each instance. The results appear in table 4.4. This table provides the run time in seconds, the number of branch-and-bound nodes solved, and the optimality gap at completion.

Table 4.4: Single Pressline Initial Results

ID	Run Time (sec)	Number of Nodes	Optimality Gap (%)
1	2	523	0.00
2	407	3297	0.00
3	23	350	0.00
4	965	399	0.00
5	2012	1116	0.00
6	4	553	0.00
7	45	259	0.00
8	1	508	0.00
9	37800	577587	9.68
10	1385	1989	0.00
11	102	548	0.00
12	84	81	0.00
13	37800	76057	0.49
14	4076	35249	0.00
15	5258	42370	0.00
16	2	70	0.00
17	230	1273	0.00
18	2394	21551	0.00
19	13219	503858	0.00
20	5101	227024	0.00
21	9	185	0.00
22	37800	26963	8.18

In nineteen of the twenty-two instances, a provably-optimal solution was found in under 3 hours and 45 minutes. This was not the case in the remaining three instances where the optimality gap was between 0.49% and 9.68% after 10.5 hours.

T&P Implementation

The $T\mathcal{E}P$ algorithm (discussed in detail in Chapter III) was developed to overcome the tractability challenges of the extreme shift schedule formulation. The $T\mathcal{E}P$ approach is based on the idea of solving many easy feasibility problems instead of one difficult optimization problem.

The algorithm begins by enumerating all feasible workforce allocation possibilities. During the enumeration the cost of each workforce allocation is calculated. For the stamping problem, the workforce allocation is the number of direct laborers and indirect laborer crews available during each shift type. Next, for each workforce

allocation the feasibility problem that corresponds to fixing the workforce allocation is solved. The optimal solution is lowest-cost workforce allocation that corresponds to a feasible workforce utilization (i.e., schedule of part types on the pressline). To improve the performance of the algorithm and reduce the number of feasibility problems that need to be solved, $T\mathcal{E}P$ can be modified to include user-specified pre-processing and pruning.

For the single pressline problem a workforce allocation is described by the number of indirect labor crews available for off-line preparations (either 0 or 1 for a single pressline) and then by its direct labor staffing. When considering a single pressline, there are only eight possible allocations for indirect labor crews. Recall that the choice of whether or not to schedule an indirect labor crew is required in each of the three shift types (first, second, and third). Similarly, for direct labor there are $|\mathbf{K}|$ choices of staffing for each of the three shift types, where \mathbf{K} is the set of unique labor requirements (including the choice of not staffing any direct laborers a given shift type). Thus, the total number of unique workforce allocations is merely: $2^3|\mathbf{K}|^3$. When considering individual presslines, the largest number of distinct direct labor requirements for the twenty-two instances considered in table 4.3 is five, therefore there are at most one thousand ($2^3 * 5^3 = 1000$) unique workforce allocations for any of the twenty-two instances.

Just as in Chapter III, y will be used to represent a workforce allocation ($\{y\}$ and $\{y^I \mid y^D\}$ will also be used to represent a workforce allocation). For example, if $\mathbf{K} = \{0, 4, 5, 6\}$ labor allocation $\{0, 1, 1 \mid 4, 6, 5\}$ for a pressline states that there is no indirect labor crew assigned to the first shift of each day in the horizon ($y_1^I = 0$), but both the second and third shift of each day have an indirect labor crew ($y_2^I = y_3^I = 1$). The first shift type has four direct laborers assigned ($y_{14}^D = 1$), the second shift type

has two direct laborers assigned ($y_{26}^D = 1$), and the third shift type has three direct laborers assigned ($y_{35}^D = 1$).

Thus, a workforce allocation could be represented as

$\{y_1^I, y_2^I, y_3^I, | \sum_{k \in \mathbf{K}} ky_{1h}^D, \sum_{k \in \mathbf{K}} ky_{2k}^D, \sum_{k \in \mathbf{K}} ky_{3k}^D\}$ using the notation from the single pressline formulation. Recall the single pressline formulation where $y_h^I = 1$ if an indirect labor crew is required in shift type h and $y_{kh}^D = 1$ if direct labor requirement k is required for shift type h .

Recall, the *T&P* algorithm has two phases. The first phase is the *Build* phase, where the *Pending List* is created by enumerating all feasible workforce allocation possibilities. The second phase is the *Process* phase, where the list is searched until the optimal solution is found.

The *Build* phase begins with an upper and lower bound for each element of the workforce allocation. The *Pending List* is created by looping through all combinations in these ranges. As the list is built, each workforce allocation is pre-processed to eliminate clearly infeasible and sub-optimal solutions.

Pre-processing the set of workforce allocations to remove clearly infeasible and sub-optimal allocations from consideration before solving any feasibility problems can significantly reduce the number of workforce allocations solved. Two types of pre-processing checks were used for the single pressline problem.

1. Pre-process by labor lower bounds Lower bounds for the number of direct laborers and indirect labor crews required in a feasible schedule can be calculated based on demand data. These bounds can help reduce the number of workforce allocations that are included in the *Pending List*. If workforce allocation $\{y\}$ does not meet the lower bounds for indirect or direct laborers then $\{y\}$ is infeasible and is not included in the *Pending List*.

- *Indirect Labor Crews Lower Bound* - Recall that there is at most one changeover per shift, ten days in the planning horizon and each day has one shift of each shift type (first, second, third). Thus in a feasible schedule, there can be up to ten changeovers in any shift type – one changeover in each shift (i.e., there can be up to ten changeovers if an indirect labor crew is assigned to only one shift type, up to twenty changeovers if indirect labor crews are assigned to two shift types, and up to thirty changeovers of indirect labor crews are assigned to all three shift types). In addition, recall that there must be at least one changeover per part type (i.e., every part type must be set up at least once so it can be produced). Therefore at a minimum the number of off-line preparations in a feasible solution is equal to the number of part types. Using these observations, we can quickly determine a lower bound on the number of shift types that must have indirect labor crews available using the expression, $\lceil \frac{|\mathbf{P}|}{10} \rceil$, where $|\mathbf{P}|$ is the number of part types produced on the pressline. The expression enforces the relationship between the number of part types and the number of shift types that require indirect labor crews. If a workforce allocation does not meet these lower bounds (i.e., $y_1^I + y_2^I + y_3^I < \lceil \frac{|\mathbf{P}|}{10} \rceil$) it will be eliminated from consideration.
- *Direct Laborers Lower Bound* - A bound can be calculated in a similar manner for each unique direct labor requirement k . Recall that each shift is eight hour long. We can use the expression, $\lceil \frac{d_k}{8} \rceil$, to calculate the lower bound on the number of shifts that require k or more direct laborers. In the expression d_k is the total demand (measured in hours) of all part types to be produced on the pressline that require k or more direct laborers ($d_k =$

$\sum_{p \in \mathbf{P}: a_p \geq k} \sum_{t \in \mathbf{T} \cup \{0\}} d_{pt} \quad \forall k \in K$). Using this expression we can quickly check to see if a workforce allocation covers the shift needs for each unique direct labor requirement. If a workforce allocation does not cover the shift needs it will be eliminated from consideration, otherwise it will move to the next pre-processing test. For example, assume that $K = \{0, 2, 4\}$ and $d_0 = 20, d_2 = 20, d_4 = 5$. Therefore, the minimum number of shifts required to meet demand is twenty (this is the minimum number of shifts required to produce all part types – every part type requires zero or more workers), the minimum number of shifts required to produce all part types that require two or more direct laborers is twenty and the minimum number of shifts required to produce all part types that require four direct laborers is five. Workforce allocations $\{y^I \mid 0, 2, 2\}$ would be eliminated from consideration – these workforce allocations provide twenty shifts with two or more direct laborers but they do not provide five shifts of four direct laborers. This calculation is a lower bound because it disregards the rules concerning changeovers and, in particular, the boundaries provided between shifts that cannot be spanned by certain tasks.

2. Pre-process by symmetry Note that pre-processing by symmetry is only applicable when the planning horizon is cyclic. When considering a cyclic planning horizon, workforce allocations are symmetric when they are equal to another when all of the shifts are rotated either one or two shift types forward, with the first shift becoming the second or third, the second becoming the third or first, and the third becoming the first or second. For example, assignment $\{0, 1, 1 \mid 4, 6, 5\}$ is symmetric to assignments $\{1, 0, 1 \mid 5, 4, 6\}$ and $\{1, 1, 0 \mid 6, 5, 4\}$. The costs of these schedules will not necessarily be the same, because

different shifts have different labor costs, but the schedules are equivalent in terms of timing and thus in terms of feasibility. [See Appendix B for a detailed discussion.] Therefore, if workforce allocation $\{y\}$ has a symmetric allocation $\{y'\}$ with cheaper cost then $\{y\}$ is sub-optimal and should not be included in the Pending List. The number of workforce allocations that are included in the Pending List can be significantly reduced by eliminating expensive symmetric workforce allocations during pre-processing.

Each workforce allocation that passes the pre-processing tests will be inserted in the Pending List.

In the Process phase, workforce allocations in the Pending List are evaluated until the list is empty and the optimal solution has been identified. First, a workforce allocation, $\{y\}$, in the Pending list is selected. Next, the appropriate variables in the shift schedule formulation are fixed to reflect the allocation and the feasibility of the allocation is tested. [Refer to Chapter III, section 3.4 for a discussion on how to implicitly fix the workforce allocation.] If a feasible solution exists for the given workforce allocation, $\{y\}$, any workforce allocation with higher cost can be eliminated; clearly any such solution is sub-optimal. Conversely, if $\{y\}$ is infeasible, any workforce allocation (or a symmetric workforce allocation) $\{\hat{y}\}$ in the Pending List for which $\hat{y} \leq y$, is removed from the Pending List; such workforce allocations have fewer labor resources in every shift type and thus will also be infeasible. In all of the computational results presented in this chapter, the workforce allocations are sorted from high-cost to low-cost and selected from the middle of the Pending List. As a result, whenever the workforce allocation, $\{y\}$ is feasible, the list will be pruned in half (because half of the candidates in the Pending List, by definition, have higher cost). When the workforce allocation $\{y\}$ is infeasible, the list is not automatically

cut in half, but in the early iterations several workforce allocations in the Pending List will be pruned by dominance, because there are several workforce allocations still in the list that have fewer resources.

Once $\{y\}$ has been processed and the Pending List updated, the workforce allocation at the middle of the (reduced) Pending List is processed. This is repeated until the list is empty. The lowest-cost feasible allocation (and its corresponding pressline schedule) is the optimal solution.

T&P Computational Results

To evaluate the performance of $T\&P$ algorithm, the same twenty-two instances as described in section 4.2.4 were considered using the same computer and CPLEX 11.0 settings to solve the feasibility problems. The results appear in Table 4.5. This table provides the run time (in seconds), the run time of the branch-and-bound approach presented earlier (for the sake of comparison), the number of workforce allocations (the maximum number of feasibility problems that could require solving), the number of workforce allocations included in the Pending List (those that passed the pre-processing checks), and the number of feasible and infeasible workforce allocation feasibility problems actually solved (the others were eliminated by pruning). If the instance did not solve within the 10.5 hour time limit it is denoted by “—”.

Table 4.6 summarizes the impact of the pre-processing techniques. The first column in the table is the pressline identification number, followed by the number pre-processed (the number possible minus the number in the pending list in the previous table), next is the number of workforce allocations that are eliminated just using the lower bound calculations, the last column lists the number of workforce allocations that are eliminated only using the symmetry pre-processing rule. Note that the sum of values in the last two columns is greater than the second column for

Table 4.5: Single Pressline Computational Results with T&P

ID	Trad Run	T&P	# Possible	# Viable	# Feasible	# Infeasible
1	2	1	216	38	5	0
2	407	221	216	17	1	5
3	23	6	512	80	2	16
4	965	369	64	6	1	2
5	2012	40	64	2	1	0
6	4	1	512	80	4	5
7	45	11	216	17	4	0
8	1	1	64	10	2	2
9	–	7308	216	17	2	4
10	1385	76	1000	136	5	6
11	102	10	216	10	2	3
12	84	2	216	38	5	0
13	–	7248	216	17	2	4
14	4076	332	512	38	1	8
15	5258	85	512	59	3	9
16	2	1	216	38	5	0
17	230	33	512	59	2	12
18	2394	7	64	10	1	4
19	13219	4509	512	59	2	12
20	5101	1051	64	17	3	1
21	9	1	216	38	4	0
22	–	582	512	26	3	3

all presslines this indicates that some workforce allocations are eliminated by both of the techniques.

As an alternative to traditional mathematical programming techniques, *T&P* finds optimal solutions to the single pressline problem by solving several feasibility problems in place of a single (but more challenging) optimization problem. The ability to prune both when the current solution is feasible and also when it is infeasible greatly enhances computational performance. Results from the single pressline demonstrates the effectiveness of this algorithm. In fifteen of the twenty-two instances, the algorithm terminated in less than five minutes. Four instances had run times between five and sixty-five minutes, and three instances had run times between sixty-five and 125 minutes. Interestingly, for each of the three instances that took over one hour to solve, almost all of the run time can be attributed to a single

Table 4.6: Single Pressline T&P Pre-processing Impact

ID	# Pre-processed	# Lower Bounds	# Symmetry
1	178	104	140
2	199	167	140
3	432	274	336
4	58	48	40
5	62	60	40
6	432	274	336
7	199	167	140
8	54	36	40
9	199	167	140
10	864	594	660
11	206	188	140
12	178	104	140
13	199	167	140
14	474	400	336
15	453	337	336
16	178	104	140
17	453	337	336
18	54	36	40
19	453	337	336
20	47	15	40
21	178	104	140
22	486	436	336

workforce allocation feasibility problem. In each of the three instances there was one workforce allocation that took a significant amount of time (over one hour) to return infeasible. This suggests the value of identifying additional mechanisms to detect infeasibility a priori.

It is also interesting to note that for those problem instances in which the branch-and-bound approach failed to find an optimal solution, the optimality gap at termination was not a reflection of a poor-quality solution but rather of a weak LP relaxation, as conjectured in Chapter III (i.e., for the instances that did not converge within 10.5 hours, $T\&P$ proved that the last integer feasible solution found by CPLEX was optimal).

4.3 Facility Operating Decisions

$T\mathcal{E}P$ and the extreme shift schedule formulation can be used to solve other stamping workforce planning problems, in addition to the single pressline problem. This section explores another workforce planning problem in the automotive stamping manufacturing environment that considers multiple presslines simultaneously to minimize facility operating costs.

4.3.1 Problem Statement

We began this chapter focusing on the detailed task of scheduling individual presslines, itself a complex and challenging problem. For some presslines, we observed that it was possible to schedule all tasks in fewer shift types than had been allocated. Careful planning may make it possible to complete all tasks in a reduced number of shifts for all of the presslines in the facility. The savings associated with eliminating a shift type (and therefore eliminating the corresponding cost of operating the facility during that time period) can be significant. This motivates the question: Can production be scheduled and changeovers be coordinated so as to reduce the number of shifts types that the plant must be open? We therefore expanded our view from considering the single pressline problem to considering all of the presslines in a given stamping facility concurrently, as they share common shifts.

For this problem, there are also six decisions that compose the workforce allocation. However, in this problem the decisions are binary and represent whether or not to permit production (i.e., make the direct laborers available) in the first, second, and third shift of each day, and whether to permit changeovers (i.e., make the indirect labor crews available) in the first, second, and third shift of each day. Thus, there are $2^3 * 2^3 = 64$ candidate workforce allocation solutions to be considered. In this section

we will continue to use $\{y\}$ or $\{y^I \mid y^D\}$ to represent a workforce allocation (e.g., “indirect labor crews are available in the third shift type only, and direct laborers are available in the first and third shift types but not the second shift type” will be represented by $\{0, 0, 1 \mid 1, 0, 1\}$).

We began by attempting to solve this problem as a traditional mixed integer program. The *MIP* is a modification of the shift schedule formulation that includes constraints for each of presslines in the facility. The notation in this formulation is very similar to that presented in Section 4.2.2, however there is an additional set, \mathbf{M} , that represents the set of presslines in the facility. To distinguish between presslines, the sets \mathbf{P} (part types) and \mathbf{S}' (extreme shift schedules), are indexed by m . The formulation is as follows:

Formulation

$$(4.21) \quad \min \sum_{h \in \mathbf{H}} c_h^I y_h^I + \sum_{h \in \mathbf{H}} c_h^D y_h^D$$

s.t.

$$(4.22) \quad \sum_{s \in \hat{\mathbf{S}}_m} x_{ns} = 1 \quad \forall n \in \mathbf{N}, \forall m \in \mathbf{M}$$

$$(4.23) \quad w_{np}^F = \sum_{s \in \hat{\mathbf{S}}_m} f_{ps} x_{ns} \quad \forall p \in \mathbf{P}_m, n \in \mathbf{N}, m \in \mathbf{M}$$

$$(4.24) \quad w_{np}^L = \sum_{s \in \hat{\mathbf{S}}_m} l_{ps} x_{ns} \quad \forall p \in \mathbf{P}_m, n \in \mathbf{N}, m \in \mathbf{M}$$

$$(4.25) \quad w_{np}^F = w_{(n-1)p}^L \quad \forall p \in \mathbf{P}_m, n \in \{2 \dots |\mathbf{N}|\}, m \in \mathbf{M}$$

$$(4.26) \quad w_{1p}^F = w_{|\mathbf{N}|p}^L \quad \forall p \in \mathbf{P}_m, m \in \mathbf{M}$$

$$(4.27) \quad \sum_{n'=n-2}^n \sum_{s \in \hat{\mathbf{S}}_{\mathbf{m}}} q_{ps} x_{n's} + i_{(n-3)p} - d_{np} = i_{np} \quad \forall p \in \mathbf{P}_{\mathbf{m}}, n \in \mathbf{T}, m \in \mathbf{M}$$

$$(4.28) \quad i_{0p} = i_{|\mathbf{N}|p} \quad \forall p \in \mathbf{P}_{\mathbf{m}}, m \in \mathbf{M}$$

$$(4.29) \quad \sum_{s \in \hat{\mathbf{S}}_{\mathbf{m}}} g_s x_{ns} \leq y_h^I \quad \forall n \in \mathbf{N}, m \in \mathbf{M}$$

$$(4.30) \quad \sum_{s \in \hat{\mathbf{S}}_{\mathbf{m}}} u_s x_{ns} \leq y_h^D \quad \forall n \in \mathbf{N}, m \in \mathbf{M}$$

$$(4.31) \quad 0 \leq x_{ns} \leq 1 \quad \forall n \in \mathbf{N}, s \in \hat{\mathbf{S}}_{\mathbf{m}}, m \in \mathbf{M}$$

$$(4.32) \quad i_{np} \geq 0 \quad \forall p \in \mathbf{P}, n \in \mathbf{T} \cup \{0\}, m \in \mathbf{M}$$

$$(4.33) \quad w_{np}^F, w_{np}^L \in \{0, 1\} \quad \forall p \in \mathbf{P}, n \in \mathbf{N}, m \in \mathbf{M}$$

$$(4.34) \quad y_h^D \in \{0, 1\} \quad \forall h \in \mathbf{H}$$

$$(4.35) \quad y_h^I \in \{0, 1\} \quad \forall h \in \mathbf{H}$$

The objective, 4.21, calculates the total cost for the facility. This formulation has constraints for each pressline to enforce the operational rules (identical to constraints 4.2 through 4.8 in the single pressline formulation) with linking constraints 4.29 and 4.30 that calculate the number of indirect and direct laborers required. Note that if $y_h^I = 0$, significant cost savings can be achieved – no indirect labor is required for that shift type. Similarly, if $y_h^D = 0$, significant cost savings can be achieved, no direct labor is required for that shift type. The largest cost savings are achieved when $y_h^I = y_h^D = 0$. Here, the plant does not require workers and the facility does not need to be open.

For each of three demand levels (High, Medium and Low), we allowed the *MIP* to run for 10.5 hours, using CPLEX 11.0 default parameters. Table 4.7 presents the results for each demand level. This table begins with the demand level followed by the run time. The next column presents the number of nodes explored in the branch-and-bound tree followed by the gap of the initial *LP* relaxation relative to the optimal solution (found later in the experiments). Next is the optimality gap of the best integer solution found relative to the lower bound from the branch-and-bound tree, and then the true optimality gap (i.e., the gap relative to the optimal integer solution). “—” denotes when no integer-feasible solution is found. Observe that the *LP* gap is quite large (45.40% to 46.96%) and that this lower bound grows slowly, leading to a significant amount of branching. In addition, observe that it was quite difficult to prove optimality with this traditional approach – only the Low instance found the optimal solution, the Medium instance had a final tree gap that was quite large and an integer-feasible instance was not found in the High instance.

Table 4.7: Facility Operating Costs Branch-and-Bound

Demand	Run Time	#Nodes	LP Relax Gap	Tree Gap	Best Integer Opt Gap
Low	4.25 hours	2150	45.40%	0.00%	0.00%
Medium	10.5 hours	6223	46.00%	21.15%	20.19%
High	10.5 hours	22765	46.96%	—	—

The remainder of this section shows how each aspect of *T&P* helps improve the computational time of this facility operating problem.

4.3.2 T&P Feature 1: Many Feasibility Problems

One of the key challenges observed in trying to solve this problem with a traditional *MIP* approach is the weak *LP* relaxation and corresponding amount of branching. This in turn is largely caused by the interaction between the workforce allocation decisions and the workforce utilization decisions. In particular, the work-

force utilization decisions (i.e the x variables) provide incentive for fractional values of the workforce allocation variables (i.e., the y variables), resulting in very large branch-and-bound trees. We therefore use $T\mathcal{E}P$ to enumerate workforce allocations and, for each of these, assess the feasibility of the corresponding workforce utilization. Clearly, the lowest cost workforce allocation for which a feasible workforce utilization exists will be the optimal solution to the problem.

Although breaking the problem into a series of feasibility problems reduces its size, many of these feasibility problems are still challenging, due in part to their large size (on the order of 115,000 variables and 15,000 constraints). We can overcome this challenge by leveraging another important benefit of pre-defining the workforce allocation decisions. Specifically, the workforce allocation decisions are the only decisions that link the presslines together, and thus when these decisions are fixed, the presslines can be *decoupled*. For a given workforce allocation solution vector, we can therefore solve a separate feasibility problem for each pressline – the workforce allocation is feasible only if the corresponding workforce utilization problem is feasible for *every* pressline. Furthermore, as soon as one pressline is shown to be infeasible, the current workforce allocation vector is known to be infeasible.

The results of this approach are shown in Table 4.8. This table provides the total run times encompassing all sixty-four feasibility problems for each of the three demand levels. It also provides information on the number of feasible and infeasible workforce utilization problems. Observe that this approach enables us to find provably optimal solutions to all three instances, with all instances completing in under 20 minutes.

Table 4.8: Facility Operating Costs Disaggregate Feasibility Problems

Demand	Run Time	# Solved	# Feasible	#Infeasible
Low	410 seconds	64	25	39
Medium	1157 seconds	64	13	51
High	816 seconds	64	4	60

4.3.3 T&P Feature 2: Pre-Processing Non-Viable Workforce Allocations

Clearly, there is a significant improvement in performance to be gained by solving each workforce utilization feasibility problem individually and disaggregating the problem across presslines, as we see in the changes in run time in the previous sets of computational results. Further improvements can be made by bypassing some of the workforce utilization feasibility problems altogether. For example, consider the workforce allocation in which none of the three shift types enable production (i.e. $\{y^I \mid 0, 0, 0\}$) or changeovers (i.e. $\{0, 0, 0 \mid y^D\}$). Clearly, then, no production can occur and thus demand cannot be met. It is not necessary to solve an optimization problem to determine that this workforce utilization problem is infeasible.

More generally, by exploiting problem structure, it is possible to eliminate many of the workforce utilization problems, recognizing that the corresponding workforce allocation solution vectors are not viable. Specifically, we determine bounds on the number of shift types required as described in Section 4.2.4. We sum the total hours of production time for all part types, which gives us a lower bound on the amount of production capacity that needs to be available. [This is a lower bound because it disregards rules concerning changeovers and, in particular, the boundaries provided between shifts that cannot be spanned by certain tasks.] We then compare this lower bound to the amount of production time available given a specific workforce allocation vector. If the workforce allocation solution is below the lower bound in terms of production capacity, we can disregard this vector. In addition, we can use

symmetry to further reduce the number of allocations in the Pending List, see section 4.2.4 for a discussion on pre-processing by symmetry.

Table 4.9 shows the run time, number of viable workforce allocation solution vectors and the number of feasible and infeasible workforce utilization problems. Observe that the number of workforce utilization problems to be solved decreases by roughly an order of magnitude. For two of the three instances, this substantially decreases the run time.

Table 4.9: Facility Operating Costs Pre-processing Feasibility Problems

Demand	Run Time (sec)	#Viable	# Feasible	#Infeasible
Low	194 seconds	10	9	1
Medium	430 seconds	6	5	1
High	741 seconds	2	1	1

4.3.4 T&P Feature 3: Pruning Feasibility Problems

The number of workforce utilization feasibility problems that need to be solved to find the optimal solution to the integrated problem can be reduced even further by recognizing that information gained in one feasibility problem often proves relevant to other feasibility problems as well. For example, if the workforce allocation where indirect labor crews are available for the first and third shift of every day is infeasible, and direct laborers are available during the first and third shift of every day (i.e., $\{1, 0, 1 \mid 1, 0, 1\}$), then clearly the workforce allocation in which direct laborers and indirect labor crews are available only during the first shift of each day will also be infeasible (i.e., $\{1, 0, 0 \mid 1, 0, 0\}$), because this is even more tightly constrained. Likewise, if that problem is feasible, then it is unnecessary to solve the instance in which direct laborers and indirect labor crews are available in every shift (i.e., $\{1, 1, 1 \mid 1, 1, 1\}$), because this solution has higher cost and is thus sub-optimal.

We therefore conducted another set of runs in which this pruning was imple-

mented. Table 4.10 shows the the run time, number of viable workforce allocation solutions after pre-processing, the number that were actually solved, whether they were feasible or infeasible, and the number that were pruned. Observe that the two problem instances which could not be solved to optimality (and, in one case, did not even yield an integer feasible solution) in 10.5 hours using a traditional branch-and-bound approach are all solved in under five minutes.

Table 4.10: Facility Operating Costs Pre-processing and Pruning Feasibility Problems

Demand	Run Time	#Viable	# Solved	# Feasible	#Infeasible	# Pruned
Low	63 seconds	10	3	3	0	7
Medium	273 seconds	6	2	1	1	4
High	291 seconds	2	1	1	0	1

4.4 Pressline Zone Decisions

The previous sections show how $T\&P$ and the extreme shift schedule formulation can be used to solve stamping workforce planning problems minimizing the cost of running a single machine as well as minimizing the cost of high-level facility operating costs. This section explores another workforce planning problem in the automotive stamping manufacturing environment where the goal is to minimize the workforce cost of pressline *zones*. A pressline zone is a group of presslines that share a pool of workers.

4.4.1 Problem Statement

To extend the previous work discussed in this chapter, we focus this section on workforce allocation and utilization for pressline zones. The goal is to determine the minimum zone-workforce cost subject to the same operating constraints discussed in the beginning of the chapter. For this problem we will assume that we know the zone assignment for each pressline. We will assume that workers can only switch presslines

at the end of a shift. Although laborers switching between presslines within a single shift is not permitted, workers can work on different presslines within the zone each day of the planning horizon.

For the sake of exposition we will only consider direct laborers in this section. Therefore, there are three decisions that compose the workforce allocation. Here, the decisions are integer and represent the number of direct laborers available in the first, second, and third shift of each day. Thus, the number of candidate workforce allocation solutions to be considered is a function of the number of presslines and the unique direct labor requirements on the presslines. In this section we will continue to use $\{y\}$ to represent a workforce allocation (e.g., “five direct laborers are available in the first and third shift types and ten direct laborers are available in the second shift type” will be represented by $\{5, 10, 5\}$).

This section first develops a new formulation and describes how to use $T\mathcal{E}P$ to determine a lower bound for the problem. The section then describes how to use the extreme shift schedule formulation and $T\mathcal{E}P$ to find solutions for when the number of distinct direct labor options on each pressline is equal to two (i.e., $K_m = 2 \quad \forall m \in \mathbf{M}$). The section concludes with a discussion on how to use $T\mathcal{E}P$ within a heuristic to develop upper bounds when there are more than two distinct direct labor options on each pressline (i.e., $\exists m \in \mathbf{M} : K_m > 2$).

4.4.2 Lower Bound

The basis for the lower bound formulation is what we refer to as a *worker distribution*. A worker distribution represents a feasible distribution of direct laborers across the presslines during a shift. A similar variable definition is used in [68] where the variable represents which tasks are being processed in parallel.

For a pressline zone with five presslines, worker distribution $[2, 2, 3, 6, 1]$ would

represent fourteen direct laborers assigned ($2 + 2 + 3 + 6 + 1 = 14$), two direct laborers assigned to the first pressline, two direct laborers assigned to the second pressline, three direct laborers assigned to the third pressline, six direct laborers assigned to the fourth pressline and one direct laborer assigned to the fifth pressline. We can develop a formulation based on the worker distribution variable definition to calculate lower bounds for the pressline zone problem. The worker distribution formulation uses much of the same notation introduced earlier in this chapter, the additional notation is as follows:

Sets

- \mathbf{J} is the set of possible worker distributions of labor across the presslines

Parameters

- b_{mk} is the number of shifts in which k direct laborers are required on pressline m (Note that b_{mk} is set equal to the number of shifts lower bound described in section 4.2.4.)
- a_{mj} is the number of laborers assigned to pressline m in worker distribution j

Variables

- $v_{nj} = 1$ if worker distribution j is assigned to shift n , else 0 $\forall n \in \mathbf{N}, j \in \mathbf{J}$

Formulation

$$(4.36) \quad \min \sum_{h \in \mathbf{H}} c_h^D y_h^D$$

s.t.

$$(4.37) \quad \sum_{j \in \mathbf{J}} v_{nj} = 1 \quad \forall n \in \mathbf{N}$$

$$(4.38) \quad \sum_{n \in \mathbf{N}} \sum_{j \in \mathbf{J}: a_{mj} \geq k} v_{nj} \geq b_{mk} \quad \forall m \in \mathbf{M}, k \in \mathbf{K}_m$$

$$(4.39) \quad \sum_{j \in \mathbf{J}} \sum_{m \in \mathbf{M}} a_{mj} v_{nj} \leq y_{h(n)}^D \quad \forall n \in \mathbf{N}$$

$$(4.40) \quad v_{nj} \in \{0, 1\} \quad \forall n \in \mathbf{N}, j \in \mathbf{J}$$

$$(4.41) \quad y_h^D \geq 0 \quad \forall n \in \mathbf{H}$$

The objective, 4.36, minimizes the cost of direct laborers. Constraints 4.37 assign a worker distribution for each shift. Constraints 4.38 enforces the minimum number of shifts required for each unique direct labor requirement k . Constraints 4.39 determines the number of direct laborers required by taking the maximum total number of direct laborers required for each shift in the shift type. The solution to the worker distribution formulation is a lower bound because there is no guarantee that a feasible schedule exists for the sequence of direct laborers assigned to each pressline.

This formulation can be modified to calculate bounds on when workers can switch between presslines *during* a shift by making the v variables continuous. The variables would then represent the amount of time during shift n that is assigned to worker distribution p . The constraints to calculate the number of workers would need to be modified to resemble Constraints 4.10 and 4.11 presented in section 4.2.2.

Branch-and-Bound Results

We began by attempting to solve this problem as a traditional *IP*. For each of the four pressline zones, we allowed the *IP* to run for 5.5 hours, using CPLEX 11.0 default parameters. Table 4.11 describes the zones considered in the computational results. This table begins with the zone identification number, followed by the presslines

Table 4.11: Pressline Zones

Zone	Presslines Included	Number of Worker Distributions
1	1, 2, 3, 4, 5	144
2	6, 7, 8, 9, 10	360
3	11,12,13,14,15,16	1296
4	17,18,19,20,21,22	768

included in the zone (these are the same presslines used in the previous sections), and the number of worker distributions for the pressline zone. Table 4.12 presents the results for each pressline zone. The table begins with the pressline zone identification number, followed by the run time (in seconds), the number of nodes, the LP relaxation gap relative to the optimal solution (found later in the experiments) and optimality gap of the best integer solution found relative to the lower bound from the branch-and-bound tree, and then the true optimality gap (i.e., the gap relative to the optimal integer solution). The run times for the zones varied significantly, and zone 2 did not find the optimal solution within the 5.5 hours. However for all four zones the best integer solution found was the optimal solution.

Table 4.12: Pressline Zones Lower Bound Branch-and-Bound

Zone	Run Time	# Nodes	LP Relax Gap	Tree Gap	Best Opt Gap
1	33 seconds	1100	3.75%	0.00%	0.00%
2	19800 seconds	2158700	3.64%	0.35%	0.00%
3	2685 seconds	27000	0.75%	0.00%	0.00%
4	94 seconds	180	0.47%	0.00%	0.00%

T&P Implementation

This problem can be solved more efficiently using $T\mathcal{E}P$. We used the same preprocessing and pruning techniques described in previous sections. Just as in the extreme shift schedule formulation we can implicitly enforce the workforce allocation to reduce the number of variables and constraints in the feasibility problems. To implicitly enforce the workforce allocation, we set each v variable to

zero if it distributes more workers than allowed in the workforce allocation (i.e., $v_{nj} = 0 \quad \forall j \in \mathbf{J}, n \in \mathbf{N} : \sum_{m \in \mathbf{M}} a_{mj} > y_{h(n)}^D$, where y_h^D is the number of direct laborers in the shift type h in the workforce allocation considered). As a result we can eliminate the objective and constraints 4.39 (the constraints that calculate the number of direct laborers required).

The computational results for finding the pressline zone lower bound with $T\mathcal{E}P$ are in table 4.13. This table shows the run time, number of possible workforce allocations, number of viable workforce allocations after pre-processing, the number of workforce allocations that were actually solved, whether they were feasible or infeasible, and the number of workforce allocations that were pruned. The run times were significantly reduced for zones 1, 2 and 3 by using $T\mathcal{E}P$, and all instances found the optimal solution within 16 minutes. In addition, this example illustrates that $T\mathcal{E}P$ can be used in problems with significantly larger number of workforce allocations. Despite the large number of possible workforce allocations a small number of feasibility problems were solved as a result of pre-processing and pruning, similarly to the instances discussed in previous sections. $T\mathcal{E}P$ was developed to overcome weak LP relaxations, however as we can see from this problem, $T\mathcal{E}P$ can also significantly improve run time in those problems that have strong LP relaxations.

Table 4.13: Pressline Zones Lower Bound T&P

Zone	Run Time	#Possible	#Viable	# Solved	# Feasible	#Infeasible	# Pruned
1	7 seconds	4096	1376	63	3	60	1313
2	59 seconds	15625	5225	196	2	194	5029
3	910 seconds	32768	10944	329	3	326	10615
4	167 seconds	21952	7336	164	4	160	7172

4.4.3 Solution Technique for When Each Pressline Has Two Distinct Direct Labor Requirements

Table 4.14 shows an example solution to the worker distribution formulation for a two-pressline zone. In the table, the workforce allocation is $\{5, 10, 11\}$, five direct laborers required in the first shift type, ten in the second shift type and eleven in the third shift type). In addition to the workforce allocation, the solution to the worker distribution formulation provides the sequence of direct laborers on pressline during each shift. In the example, for pressline 1, there are five direct laborers in shifts one, four, five and six. There are six direct laborers on pressline 1 during shifts two and three.

Table 4.14: Workforce Distribution Solution Example

Shift	Workforce Distribution
1	[5, 0]
2	[6, 3]
3	[6, 5]
4	[5, 0]
5	[5, 5]
6	[5, 3]

Let α represent that sequence of direct laborers provided by the solution to the worker distribution formulation, where α_{mn} is the number of direct laborers assigned to pressline m in shift n . In the example above, $\alpha_{1,1} = 5$, $\alpha_{1,2} = 6$, $\alpha_{1,3} = 6$, $\alpha_{1,4} = 5$, $\alpha_{1,5} = 5$ and $\alpha_{1,6} = 5$.

For each pressline, the sequence of the number of direct laborers available during each shift can be enforced in the extreme shift schedule formulation by not including shift schedules that violate the sequence (i.e., $x_{ns} = 0$ if shift schedule $s \in \hat{\mathbf{S}}_m$ requires more than α_{mn} direct laborers). If α_m corresponds to feasible workforce utilization for every pressline in the zone, then the we have found a feasible solution to the pressline zone problem.

In our experiments, α did not correspond to a feasible workforce utilization for every pressline in the zone. [Recall that we were using lower bounds on the number of shifts for each unique direct laborer requirement k in the workforce distribution formulation.]

However, for instances of the pressline zone problem where $|\mathbf{K}_m| = 2 \quad \forall m \in \mathbf{M}$ (e.g., $K_1 = \{0, 6\}$, and $K_2 = \{0, 5\}$), we can use the worker distribution formulation in conjunction with the extreme shift schedule formulation and $T\mathcal{E}P$ to find the optimal solution. Instead of solving a single feasibility problem for each workforce allocation during the Process phase, we would implement the following algorithm.

1. Given the workforce allocation, solve the worker distribution formulation:
 - If feasible, $m = 1$, go to step 2,
 - If infeasible, the $T\mathcal{E}P$ allocation is *infeasible* – prune by infeasibility.
2. For pressline m , given the sequence of direct laborers (provided by the worker distribution formulation solution), solve the extreme shift schedule formulation:
 - If feasible and $m < |\mathbf{M}|$, $m = m + 1$ go to step 2,
 - If feasible and $m = |\mathbf{M}|$, the $T\mathcal{E}P$ allocation is *feasible* – prune by feasibility.
 - If infeasible, add a cut to the worker distribution formulation for pressline m and go to step 1,

If the sequence of direct laborers is infeasible for pressline m , the cut included in the worker distribution formulation should increase number of shifts that require direct laborers, i.e., $b_{mk} = 1 + \sum_n \sum_{d: a_{mj} > 0} v'_{nj}$, where v' is the solution to the infeasible worker distribution instance. This is a valid cut because the only way for the sequence

to become feasible is to include more direct laborers, and the only way to include more direct laborers is to require that more shifts have direct laborers available. Recall that using this cut will only yield an optimal solution when every pressline only has one unique direct laborer requirement besides zero ($|\mathbf{K}_m| = 2 \quad \forall m \in \mathbf{M}$). [Otherwise the solution to this problem serves as an upper bound to the pressline zone problem (i.e., $|\mathbf{K}_m| \geq 2$ for at least one $m \in \mathbf{M}$). Unfortunately, our efforts to use a similar *cut generation* technique to expand this algorithm to find the optimal solution to the pressline zone problem with presslines with more than two distinct direct labor requirements were unsuccessful.]

We implemented this algorithm for the four pressline zones, with $\mathbf{K}_m = \{0, \max_{p \in \mathbf{P}_m}(a_p)\} \quad \forall m \in \mathbf{M}$ using CPLEX 11.0 default parameters. The computational results for finding the pressline zone upper bound with $T\mathcal{E}P$ are in table 4.15. This table shows the run time, number of possible workforce allocations, number of viable workforce allocations after pre-processing, the number of workforce allocations that were actually solved, whether they were feasible or infeasible, and the number of workforce allocations that were pruned. The run times for these instances ranged between 1.5 to 16.5 hours, and the majority of this time was used to prove that instances were infeasible.

Table 4.15: Pressline Zones Solutions

Zone	Run Time	#Possible	#Viable	# Solved	# Feasible	#Infeasible	# Pruned
1	35963 seconds	1000	340	23	3	20	317
2	7813 seconds	729	249	25	2	23	224
3	57734 seconds	8000	2680	84	2	82	2596
4	4902 seconds	15625	5225	178	2	176	5047

4.4.4 Upper Bound

We also considered heuristic approach to calculate an upper bound for zones that include machines that have more than two distinct direct laborer requirements. This approach also uses $T\mathcal{E}P$ in conjunction with the extreme shift schedule formulation. The heuristic is as follows:

1. For each pressline in the zone, solve $T\mathcal{E}P$ to minimize the direct labor cost
2. Given a solution for each pressline, solve an IP to minimize the direct labor cost of the zone

The goal of the IP is to rotate the single pressline solutions to find the best-cost solution for the zone. Recall that the single pressline solutions are cyclic, therefore the solution stays feasible even if it is rotated several shifts. The IP minimizes the direct labor cost of the zone given a feasible solution for each pressline. The formulation is below:

Parameters

- α_{mn} is the number of direct laborers needed during shift n on pressline m according to the solution of the single pressline problem $\forall n \in \mathbf{N}, m \in \mathbf{M}$ (these values come from the single pressline solutions found during step 1 of the algorithm)

Variables

- $q_{mn} = 1$ if the first shift in the pressline m solution is assigned to shift n , else 0 $\forall n \in \mathbf{N}, m \in \mathbf{M}$ (recall that the goal of this formulation is to rotate the single pressline solutions to find the best solution for the zone).

Formulation

$$(4.42) \quad \min \sum_{h \in \mathbf{H}} c_h^D y_h^D$$

s.t.

$$(4.43) \quad \sum_{n \in \mathbf{N}} q_{mn} = 1 \quad \forall m \in \mathbf{M}$$

$$(4.44) \quad q_{11} + q_{12} + q_{13} = 1$$

$$(4.45) \quad \sum_{m \in \mathbf{M}, n' \in \mathbf{N}: n \geq n'} a_{m(n-n'+1)} q_{mn} + \sum_{m \in \mathbf{M}, n' \in \mathbf{N}: n < n'} a_{m(|N|+n-n'-1)} q_{mn} \leq y_{h(n)}^D \quad \forall n \in \mathbf{N}$$

$$(4.46) \quad q_{mn} \in \{0, 1\} \quad \forall n \in \mathbf{N}, \forall m \in \mathbf{M}$$

$$(4.47) \quad y_h^D \geq 0 \quad \forall h \in \mathbf{H}$$

The objective, 4.42, minimizes the direct labor costs. Constraints 4.43 assigns exactly one shift to correspond to the start of the direct labor sequence for each pressline. Constraints 4.44 eliminates some symmetric solutions by only allowing the first pressline's solution to begin during the first three shifts in the horizon. Constraints 4.45 calculates the number of direct laborers required while preserving the sequence of shifts in the pressline solutions (i.e., if the pressline solution begins on the third shift ($q_{m3} = 1$) the first shift in the pressline solution is actually the second shift for the pressline zone solution, the second shift in the pressline solution is actually the third shift for the pressline zone solution, third shift in the pressline solution is actually the fourth shift for the pressline zone solution, ..., the thirtieth shift in the pressline solution is actually the first shift for the pressline zone solution).

Table 4.16 shows how the solution compares to the lower bound discussed in Section 4.4.2. The solution to this problem serves as an upper bound, this new upper bound is between 22% and 29% percent of the lower bound.

Table 4.16: Pressline Zones Gap Between Lower and Upper Bound

Zone	Run Time	% Gap to Lower Bound
1	490	28.25 %
2	1871	22.01%
3	199	23.09%
4	3326	23.12%

4.5 Conclusions

This chapter began with a description of an automotive stamping plant and throughout the chapter we presented case studies based on an automotive stamping plant where the extreme shift schedule formulation in conjunction with $T\&P$ provided implementable solutions in reasonable run times.

Using variables that represent a feasible sequence of tasks in a shift has allowed us to formulate a complex production planning problem, incorporating significant real-world detail, without the need for a large number of constraints. By using shift schedule variables, we are able to capture complex operational policies about changeovers, as well as a non-linear cost function in which laborers hired on any day in the planning horizon must be paid for all days in the horizon. The sequence of part types, quantity of each type produced, changeover between part types, and number of laborers needed in a shift are all represented by a single variable. Furthermore, this approach bypasses many of the limiting assumptions often seen in the literature – there are no limits on the size of the batches, the inventory level when a changeover occurs, or the amount of labor available, and the number of changeovers is not fixed in advance.

The extreme shift schedule model presented is quite flexible, with potential use in day-to-day operations in addition to longer-term planning. For example, the model may be used to return to the existing plan after a disruption such as a pressline

failure or sudden demand change has occurred. This could be done by solving a variation of the model in which the starting inventory levels are set to the current inventory levels and the ending inventory levels are set to the planned values (the constraints that enforce a cyclic schedule would also be removed from the model). The objective could be to minimize the cost of returning to this level, but also could be to minimize the time to return to the plan, implemented simply by checking the feasibility of progressively longer planning horizons. In addition to being able to assist decision makers when presslines fail and demand changes, the single pressline formulation served as a start point to finding solutions to problems that consider multiple presslines simultaneously. The solution approach to minimize the facility operating costs as well as minimizing the direct labor costs for a zone each used the single pressline formulation as a building block.

$T\mathcal{E}P$ also was quite flexible when solving the case studies. We were able to easily incorporate application-specific pre-processing rules and we effectively solved instances with up to 32,768 possible workforce allocations. Moreover, we were able to demonstrate that $T\mathcal{E}P$ can easily work with more than one type of feasibility “black box”. In this chapter, we illustrated that $T\mathcal{E}P$ is effective when the feasibility “black box” is a single integer program (in Section 4.2), a series of integer programs (in Section 4.3), and an algorithm (in Section 4.4). In addition, in section 4.4 we showed how $T\mathcal{E}P$ can be used as part of a larger algorithm to quickly find a high-quality feasible solution for the pressline zone problem.

CHAPTER V

Conclusions

In this dissertation we discussed Shift-Workforce Allocation and Utilization problems. *SWA-WU* is described in detail in the Introduction. Recall, *SWA-WU* integrates two sets of decisions: (1) the number of workers of each skill set available during each shift (shift-workforce allocation) and (2) the sequence and duration of tasks to meet demand (workforce utilization). The objective is to determine the lowest-cost workforce allocation that corresponds to a feasible workforce utilization. In *SWA-WU*, there are non-linear relationships and complex changeover constraints that makes these problems difficult to model and solve. In this dissertation, we presented a formulation to accurately model *SWA-WU* and developed an algorithm, called *Test-and-Prune* to efficiently solve the formulation. We used problems found in automotive stamping manufacturing environment to demonstrate the effectiveness of the model and algorithm.

5.1 Work in Shift-Workforce Allocation and Utilization

In Chapter II, we discussed how traditional mathematical modeling approaches could not capture all of the constraints in *SWA-WU*. As an alternative, we developed a formulation based on composite variables, variables that represent multiple decisions simultaneously. The *CVs* used to model *SWA-WU* are called shift sched-

ules. Each shift schedule represents a feasible order and duration of tasks to be completed in a single shift. By using this definition, constraints to enforce the complex changeover rules are not needed in the model, these constraints are captured implicitly within the variables. The remaining constraints are captured using the characteristics of the shift schedules (e.g. first task, last task, number of changeovers, etc.). These characteristics are also used to linearize the objective function. Instead of including a very large number of binary variables in the model, we showed that all of the feasible shift schedules can be represented as a convex combination of a small, select subset of specialized shift schedules, called extreme shift schedules.

The computational results for the shift schedule formulation were very promising with nineteen of twenty-two problem instances solving in under 3 hours and 45 minutes. Unfortunately, there were three instances that had large run times. These computational results motivated us to develop a new algorithm to effectively solve the extreme shift schedule formulation.

The algorithm, $T\mathcal{E}P$, is discussed in detail in Chapter III. $T\mathcal{E}P$ finds optimal solutions to the extreme shift schedule formulation by solving a series of feasibility problems instead of solving a single optimization problem. There are two phases to the $T\mathcal{E}P$ algorithm. First, we *build* the list of candidate workforce allocation decisions (i.e. enumerate all of the possible workforce allocations); we then *process* the list (i.e. select a workforce allocation in the list and determine if a feasible workforce utilization exists for the workforce allocation) until the list is empty. The lowest-cost workforce allocation with a corresponding feasible workforce utilization is the optimal solution. The performance of the algorithm can be improved by pre-processing and pruning. Pre-processing occurs during the Build phase. During pre-processing, a series of user-specified rules (e.g. lower bounds on the number of

workers) are used to eliminate clearly infeasible or sub-optimal solutions from the list. Pruning occurs during the Process phase. In pruning, we use information gained by solving one feasibility problem to give us information about the results for other feasibility problems. For example, if the workforce allocation is feasible, any workforce allocation with higher cost can be pruned. Any workforce allocation with higher cost than the feasible allocation is sub-optimal. If the workforce allocation is infeasible, any workforce allocation with fewer or the same number of workers in each shift type can be pruned. Any workforce allocation with the same or fewer number of workers has fewer resources and will also be infeasible.

We use the shift schedule formulation and $T\mathcal{E}P$ to address workforce planning problems for automotive stamping plants in Chapter IV. We focus on three different case studies (1) a single pressline problem – where workers are restricted to work on a single pressline, (2) an operating costs problem – where the goal is to minimize the number of shifts the plant is open, and (3) a pressline zone problem – where workers are shared across a group of presslines. We show how we can extend the shift schedule model to the stamping problem. Within these case studies, we also show how to incorporate pre-processing rules in $T\mathcal{E}P$ to significantly improve the run time of the algorithm. Through these case studies, we are able to demonstrate that $T\mathcal{E}P$ can solve instances of $SWA-WU$ significantly faster than traditional integer programming techniques and $T\mathcal{E}P$ can effectively solve instances with tens of thousands of workforce allocations.

5.2 Contributions

This work makes a number of contributions. The first set of contributions are with respect to accurately modeling $SWA-WU$.

CVs that represent entire schedules for individual shifts within the planning horizon can capture significant operational complexity. These variables also easily capture sequence-dependent changeover times. The automotive case studies demonstrated the effectiveness of the extreme shift schedule model. We were able to incorporate significant operational complexities, without the need for a large number of constraints and variables. Moreover, this work does not include many of the assumptions often seen in the scheduling literature – there are no limits on the size of the batches, the inventory level when a changeover occurs, or the amount of labor available, and the number of changeovers is not fixed in advance.

The benefits of a *CV* modeling approach typically come at the cost of a very large number of binary variables and/or restrictions on the solution space such as the discretization of time. A significant contribution of this work is in recognizing that all of the feasible shift schedules can be represented as a convex combination of a small, select subset of specialized extreme shift schedules. This enables the creation of an extreme shift schedule model that exhaustively captures all possible schedules (i.e. the set of shift schedules considered is not limited and time is not discretized) while keeping the number of *CVs* small. As an added benefit, the integrality of the remaining composite variables can be relaxed, leaving only a small set of auxiliary variables restricted to be integer.

An additional contribution is that the extreme shift schedule formulation can be used to easily evaluate many types of scenarios with limited modification. For example the extreme shift schedule model could be used in day-to-day operations as well as longer-term planning. The model can also easily accommodate alternative objectives. For example, the objective could be to minimize the cost of the workforce or to minimize the number of shifts that require workers.

The second group of contributions of this work is with respect to developing $T\mathcal{E}P$ – a tractable solution approach for $SWA-WU$. For the automotive stamping case studies, $T\mathcal{E}P$ provided implementable solutions in reasonable run times. In addition, $T\mathcal{E}P$ also was quite flexible when solving the automotive stamping case studies. We were able to easily incorporate pre-processing rules and we effectively solved instances with up to 32,768 possible workforce allocations. Our last contribution is that $T\mathcal{E}P$ allows customization by the user in a number of areas (e.g. pre-processing and pruning rules, feasibility problems). Due to this contribution, $T\mathcal{E}P$ has applicability beyond the $SWA-WU$ for a much broader class of problems - those in which a discrete and finite set of resource allocation decisions dominate cost while a more substantial set of resource utilization decisions dominate complexity.

5.3 Future Work

There are several extensions of this work that present interesting opportunities for future research. One area for future research is relaxing our assumptions.

In the automotive stamping problem there are a few opportunities to relax assumptions including:

1. Increasing the maximum number of changeovers allowed in a single shift

In this dissertation we assume that there is at most one changeover per shift. In Chapter IV we discussed how extreme shift schedules can be created for shifts with two changeovers (i.e. shifts that include three part types). In addition, Appendix C includes a detailed description of two changeover shift schedules. In the future, we could include these two changeover shift schedules in the model and run experiments to see how including the additional variables would effect the cost of the solution and the run time of the $T\mathcal{E}P$ algorithm.

2. Decoupling the on-line diesel and off-line preparation in the shift schedules

In practice the on-line diesel is directly followed by the off-line preparation, however these two events do not need to occur back-to-back in the same shift. It would be an interesting extension to develop a model that allows the two activities to occur in different shifts to see if costs savings can be achieved.

There are also opportunities to extend this work so that it can be used to address other *SWA-WU* problems such as:

1. Incorporating worker breaks within shift schedules

Worker breaks is a topic that is not addressed in this dissertation. If *every* worker breaks at the same time we can simply adjust the duration of the shift schedule. For example if the shift duration is eight hours and *all* workers get a half-hour break at the same time then our shift schedules would be 7.5 hours long. Unfortunately in many applications workers do not all take breaks at the same time, additional work would be required to extend the shift schedule formulation to capture worker breaks for those applications.

2. Developing techniques to find better lower bounds during pre-processing

We discussed in Chapter IV how we calculated lower bounds on the number of workers required throughout the horizon. We could extend this work by improving these lower bounds and generalizing the techniques for *SWA-WU*. If we could determine better lower bounds, we could reduce the number of infeasible workforce allocations processed during *T&P* (i.e. with a better bound we would pre-process more workforce allocations). Improving the lower bounds could result in significantly faster run times, because infeasible allocations often take significantly longer to process than feasible allocations.

3. Exploring alternative search techniques when processing the list

In all of the $T\&P$ computational results presented in this work we select from the middle of the list during the Process phase. It would be interesting to see how alternative selection rules (e.g. random, most-dominated workforce allocation first) would effect the run time. In addition, it would be interesting to see if graph theory could be used to develop selection techniques for the Process phase.

While working on this dissertation it became clear that it is very beneficial to have one model that can be easily modified to evaluate different operational constraints. The shift schedule model presented in this dissertation falls in this category. This model can be easily modified to evaluate several scenarios (e.g. restricting the placement of particular tasks, changing labor responsibilities, etc.). Another area of future research is to investigate how models with this characteristic can be created for other domains.

There is an opportunity for researchers to investigate how quantitative models and algorithms can be used to help understand the tradeoffs when making human resources decisions. A significant number of qualitative studies discuss how productive and satisfied workers are with different types of schedules (e.g Monday through Friday 9 am to 5 pm, varying start times and fixed duration, varying shift durations and fixed start times, 10 hours 4 day per week), but this information has yet to be coupled with quantitative models to help decision makers understand potential tradeoffs (e.g. workforce costs, number of workers required). The work in this dissertation can serve as a foundation for new models and algorithms to explore this important area.

There are a large number of resource allocation and utilization problems where the

resources are not workers. Nevertheless in these problems there are many constraints of the same form, therefore an additional area of future research is exploring how the extreme shift schedule concept can be applied to other resource allocation problems. Can the extreme shift schedule concept be modified for resource allocation and utilization problems when the resources are buses or trucks or dollars? Identifying these opportunities and modifying our model to address them is also an interesting area for future research.

APPENDICES

APPENDIX A

Single Machine Scheduling Literature

In the broadest sense, single machine scheduling problems are problems which plan the production of a group of jobs or part types on a single machine which can only produce one part type at a time and which may incur downtime when changing over from one part type to the next.

Within this class of problems are many variations. Relevant characteristics include:

- Changeover time: Is there a delay when changing from one part type to another?
- Static vs. dynamic: Is the demand fixed in advance or do new demands arise after planning has been completed?
- Stochastic vs. deterministic: Is the demand known in advance? Are production times known in advance? Are machine failures taken into consideration?
- Objective function: These include: minimize weighted completion time; minimize maximum tardiness; minimize inventory and changeover costs; etc.
- Solution approaches: These include: mathematical programming; dynamic programming; local search heuristics; etc.

Virtually all combinations of these characteristics are considered in the literature. For example, the static, deterministic problem with changeover times is solved by [29] using local search heuristics to minimize weighted completion time. A stochastic and dynamic version of the problem, which uses feedback control in a heuristic framework, appears in [63]. [19] considers the problem of sequencing products so as to minimize the makespan, with the added requirement of restrictions (upper and lower bounds) on the difference in start times between fixed pairs of events. Research on the problem without changeover costs is considered in [67]. The work of [24] considers a somewhat less common objective function – in their work, jobs are grouped into batches, with changeovers between batches. Each job in a batch is released when the batch is complete; the objective is to minimize the sum of all release times. [35] considers the related problem of deciding which orders to fill in a given time horizon, given that each order contains multiple part types and the changeover time between part types is high. A related problem is considered in [47], which uses a dynamic programming approach. The dynamic scheduling problem considered by [44] also takes a dynamic programming approach, focusing on minimizing changeover and inventory costs. [83] includes a discussion of key properties of several single-machine scheduling problems. [16] considers the problem in which demands are known but time-varying over an infinite horizon; they use an analytical approach to reduce the feasible region and then solve the problem via branch-and-bound techniques. A similar version of the problem is considered by [85], with the objective of minimizing the sum of completion times; they assume a common due date for all jobs. Sequence-dependent changeovers are considered in [36], with a constant demand rate and zero-switch rule. An early example of the stochastic scheduling problem appears in [14]. [37] minimizes flow time in a static,

deterministic problem environment. A more unusual version of the problem appears in [56], which considers a 0/1, sequence-dependent setup cost. The problem in which changeovers are sequence dependent is considered by [61]. Changeovers that are sequence dependent in that they depend on whether the two part types come from the same family are considered by [71]. [51] considers the scheduling of planned maintenance within pre-specified time windows. Changeovers are minimized in [49]. [65] uses cutting planes to solve a dynamic version of the problem. [86] provides a survey of much of the single-machine scheduling literature as part of a broader body of work. Other interesting survey papers include [51], [59], [72], and [75]. The problem of determining lot sizes and sequences considered in [33] is perhaps the most similar to the pressline scheduling problem that we consider; in their case, changeover costs are fixed and the horizon is infinite (a steady-state optimum is sought). This paper discusses the challenges in developing a mathematical program for the problem, and uses a heuristic that generates sequences and then solves a mathematical program to find lot sizes for these sequences. The same problem is considered more recently by [40]; they use the threshold accepting local search algorithm to find solutions without requiring a pre-defined time period. Another paper similar to our research is the work of [77], which includes efficient algorithms for solving special cases of the discrete lot-sizing and scheduling problem. Also similar is the work of [78], which uses a branch-and-bound based heuristic to minimize the maximum lateness. This objective is considered in [69] as well.

APPENDIX B

Rotated Case Study Solutions Maintain Feasibility Status

If a solution to one of the case studies is rotated r shifts it maintains its feasibility status (feasible or infeasible). See Chapter IV for the assumptions and constraints for the automotive stamping case studies. It is important to note that this result only holds when the planning horizon is cyclic.

Let $\text{solution}(r)$ denote that the solution is rotated r shifts forward where $\text{solution}(0)$ denotes the initial solution (i.e. shift 2 in $\text{solution}(1)$ corresponds to shift 1 in $\text{solution}(0)$, shift 3 in $\text{solution}(1)$ corresponds with shift 2 in $\text{solution}(1)$... shift 1 in $\text{solution}(0)$ corresponds to shift 30 in $\text{solution}(0)$). The remainder of this Appendix explains why $\text{solution}(r)$ has the same feasibility status as $\text{solution}(0)$.

B.1 Feasible solution exists

A feasible schedule is one where *total demand equals total production* for each part type (this relationship derived below). Thus when $\text{solution}(0)$ is feasible (i.e. the total production equals to total demand), $\text{solution}(r)$ is also feasible, rotating the solution r shifts does not change the total production or the total demand. The initial inventory can be calculated such that $\text{solution}(r)$ will always have non-negative inventory (e.g. the initial inventory equals the total demand for the planning horizon) thus we can pre-process by symmetry within $T\&P$.

We use an example with a three-day planning horizon, $n \in \{1, 2, \dots, 9\}, t \in \{3, 6, 9\}$, to show that the total production must equal the total demand in a feasible solution. For a feasible solution the following constraints from the single pressline formulation must be satisfied for each part type p (from constraint sets 4.7 and 4.8):

$$(B.1) \quad \sum_{n \in \{1, 2, 3\}} \left(\sum_{s \in \hat{\mathbf{S}}_{\mathbf{m}}} q_{ps} x_{ns} \right) + i_{0p} - d_{3p} = i_{3p}$$

$$(B.2) \quad \sum_{n \in \{4, 5, 6\}} \left(\sum_{s \in \hat{\mathbf{S}}_{\mathbf{m}}} q_{ps} x_{ns} \right) + i_{3p} - d_{6p} = i_{6p}$$

$$(B.3) \quad \sum_{n \in \{7, 8, 9\}} \left(\sum_{s \in \hat{\mathbf{S}}_{\mathbf{m}}} q_{ps} x_{ns} \right) + i_{6p} - d_{9p} = i_{9p}$$

$$(B.4) \quad i_{0p} = i_{9p}$$

Constraint B.1, B.2, and B.3 are the inventory constraints for the first, middle and last day of the planning horizon respectively. Constraint B.4 enforces that the planning horizon is cyclic.

By rearranging the constraint B.3 and B.2:

$$(B.5) \quad i_{3p} = i_{6p} + d_{6n} - \sum_{n \in \{4, 5, 6\}} \left(\sum_{s \in \hat{\mathbf{S}}_{\mathbf{m}}} q_{ps} x_{ns} \right)$$

$$(B.6) \quad i_{6p} = i_{9p} + d_{9n} - \sum_{n \in \{7, 8, 9\}} \left(\sum_{s \in \hat{\mathbf{S}}_{\mathbf{m}}} q_{ps} x_{ns} \right).$$

By substituting constraint B.6 in constraint B.5 and setting constraint B.1 equal to constraint B.5:

$$(B.7) \quad \sum_{n \in \{1,2,3\}} \left(\sum_{s \in \hat{\mathbf{S}}_m} q_{ps} x_{ns} \right) + i_{0p} - d_{3n} = d_{6n} + d_{9n} + i_{9p} - \sum_{n \in \{4,5,6,7,8,9\}} \left(\sum_{s \in \hat{\mathbf{S}}_m} q_{ps} x_{ns} \right).$$

By rearranging constraint B.7:

$$(B.8) \quad \sum_{n \in N} \left(\sum_{s \in \hat{\mathbf{S}}_m} q_{ps} x_{ns} \right) = i_{9p} - i_{0p} + d_{6n} + d_{3n} + d_{9n}.$$

Recall that the planning horizon the ending inventory equals the starting inventory, thus $i_{9p} - i_{0p} = 0$. By substitution, it is clear that the total production equals the total demand:

$$(B.9) \quad \sum_{n \in N} \left(\sum_{s \in \hat{\mathbf{S}}_m} q_{ps} x_{ns} \right) = d_{3n} + d_{6n} + d_{9n}.$$

The argument holds for the n -day planning horizon – any day between the first and last day of the planning horizon can be represented as the middle day.

B.2 Feasible solution does not exist

An infeasible schedule is one where *total demand is not equal to the total production*. When an infeasible schedule is rotated by any value of r it remains infeasible because the demand will not decrease nor will the production increase in the schedule. Thus we can prune by symmetric dominance when in the Process phase of *T&P*.

APPENDIX C

Additional Stamping Extreme Shift Schedules

There were several types of extreme shift schedules that were derived as part of this work. This Appendix discusses extreme shift schedules that were derived but include operational capabilities that were not needed for the case studies.

This Appendix is divided into three sections. The title of each section describes the type of shift schedules derived. Each section includes constraints that describe the feasible region for a type of shift schedules. The extreme points each feasible region are the extreme shift schedules (recall, all other feasible shift schedules will be convex combinations of the extreme shift schedules).

There is an active constraints table in each section that illustrates where the extreme points are found. Each table has three columns:

- *Constraints*: Lists the active constraints are considered in that row.
- *Result*: States the result when the active constraints are solved. The options are basic-feasible, basic-infeasible, and non-basic.
- *Solution*: States the extreme point found if the result is basic-feasible, lists which constraint is violated if the result is basic-infeasible or is blank if the result is non-basic.

Each section concludes with a summary of the extreme shift schedules (extreme points) for each type of shift schedule considered. We assume that the duration of the shift is Ω hours.

Assume the following notation regarding shift schedules:

Sets

- \mathbf{P} is the set of part types to be produced
- $\hat{\mathbf{S}}$ is the set of extreme shift schedules
- \mathbf{K} is the set of unique direct labor requirements across all part types, plus the option to not staff a shift type (e.g. if part types A and B each require 2 laborers during production and part types C and D require 4, then \mathbf{K} would be $\{0, 2, 4\}$)

Parameters

- $f_{ps} = 1$ if shift schedule s starts with the pressline set for part type p , else 0
 $\forall p \in \mathbf{P}, s \in \hat{\mathbf{S}}$ (illustrated in Figure 2.3)
- $l_{ps} = 1$ if shift schedule s ends with the pressline set for part type p , else 0
 $\forall p \in \mathbf{P}, s \in \hat{\mathbf{S}}$ (illustrated in Figure 2.3)
- $m_{ps} = 1$ if shift schedule s has the pressline set with part type p in between the first and last part type, else 0
 $\forall p \in \mathbf{P}, s \in \hat{\mathbf{S}}$
- g_s is the number of changeovers (on-line dieset and off-line preparation) in shift schedule s $\forall s \in \hat{\mathbf{S}}$

C.1 Three part type shift schedules

These shift schedules have three part types set up during the shift. In this scenario, a part type can not be directly followed by itself and there are exactly two on-

line diesets. Therefore a shift schedule, s , of this type would have the following parameters: $g_s = 2$, $f_{j_s} \neq m_{j_s} \quad \forall j \in \mathbf{J}$ and $m_{j_s} \neq l_{j_s} \quad \forall j \in \mathbf{J}$. Let $t(A)$ denote the amount of time that part type p is produced. Assume that A is the first part type produced, B is the middle part type produced and C is the last part type produced. Therefore $f_{A_s} = 1$, $m_{B_s} = 1$, and $l_{C_s} = 1$. Let Δ_{ij} represent amount of time to complete the on-line dieset from part type i to part type j and Γ represent amount of time to complete the off-line preparation. Just as in Chapter IV, in this section, we will continue to assume that a changeover must be fully contained in a single shift. In addition, we assume that there is enough time within the shift to complete two changeovers – each with an on-line dieset and off-line preparation (i.e. $\Delta_{AB} + \Delta_{BC} + 2\Gamma \leq \Omega$). The constraints on these three part type shift schedules are as follows:

$$(C.1a) \quad t(A) \geq 0$$

$$(C.1b) \quad t(B) \geq 0$$

$$(C.1c) \quad t(C) \geq 0$$

$$(C.1d) \quad t(A) \leq \Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma$$

$$(C.1e) \quad t(A) + t(B) \leq \Omega - \Delta_{AB} - \Delta_{BC} - \Gamma$$

$$(C.1f) \quad t(A) + t(C) \leq \Omega - \Delta_{AB} - \Delta_{BC} - \Gamma$$

$$(C.1g) \quad t(A) + t(B) + t(C) \leq \Omega - \Delta_{AB} - \Delta_{BC}$$

Constraints C.1a, C.1b and C.1c state that the duration of work on a task (A, B , and C) can not be negative. Constraint C.1e requires that the duration of tasks A and B leaves enough time for the changeover (on-line dieset and off-line preparation) to task C . Constraint C.1f requires that the duration of tasks A and C leaves enough time for the changeover to task B . Constraint C.1g states that the total time working

cannot exceed the length of the shift minus the time required for both of the on-line diesets and both off-line preparations. (recall the assumption that the shift is Ω hours long).

Active Constraint Table

Constraints	Result	Solution $[t(A), t(B), t(C)]$
(C.1a), (C.1b), (C.1c)	Basic-Feasible	$[0, 0, 0]$
(C.1a), (C.1b), (C.1d)	Non-Basic	
(C.1a), (C.1b), (C.1e)	Non-Basic	
(C.1a), (C.1b), (C.1f)	Basic-Feasible	$[0, 0, \Omega - \Delta_{AB} - \Delta_{BC} - \Gamma]$
(C.1a), (C.1b), (C.1g)	Basic - Infeasible	Violates (C.1f)
(C.1a), (C.1c), (C.1d)	Non-Basic	
(C.1a), (C.1c), (C.1e)	Basic - Feasible	$[0, \Omega - \Delta_{AB} - \Delta_{BC} - \Gamma, 0]$
(C.1a), (C.1c), (C.1f)	Non-Basic	
(C.1a), (C.1c), (C.1g)	Basic - Infeasible	Violates (C.1e)
(C.1a), (C.1d), (C.1e)	Non-Basic	
(C.1a), (C.1d), (C.1f)	Non-Basic	
(C.1a), (C.1d), (C.1g)	Non-Basic	
(C.1a), (C.1e), (C.1f)	Basic - Infeasible	Violates (C.1g)
(C.1a), (C.1e), (C.1g)	Basic - Feasible	$[0, \Omega - \Delta_{AB} - \Delta_{BC} - \Gamma, \Gamma]$
(C.1a), (C.1f), (C.1g)	Basic - Feasible	$[0, \Gamma, \Omega - \Delta_{AB} - \Delta_{BC} - \Gamma]$
(C.1b), (C.1c), (C.1d)	Basic - Feasible	$[\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma, 0, 0]$
(C.1b), (C.1c), (C.1e)	Basic - Infeasible	Violates (C.1d)
(C.1b), (C.1c), (C.1f)	Basic - Infeasible	Violates (C.1d)
(C.1b), (C.1c), (C.1g)	Basic - Infeasible	Violates (C.1d)
(C.1b), (C.1d), (C.1e)	Non-Basic	

(C.1b), (C.1d), (C.1f)	Basic - Feasible	$[\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma, 0, \Gamma]$
(C.1b), (C.1d), (C.1g)	Basic - Infeasible	Violates (C.1f)
(C.1b), (C.1e), (C.1f)	Basic - Infeasible	Violates (C.1d)
(C.1b), (C.1e), (C.1g)	Basic - Infeasible	Violates (C.1d)
(C.1b), (C.1f), (C.1g)	Non-Basic	
(C.1c), (C.1d), (C.1e)	Basic - Feasible	$[\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma, \Gamma, 0]$
(C.1c), (C.1d), (C.1f)	Non-Basic	
(C.1c), (C.1d), (C.1g)	Basic - Infeasible	Violates (C.1e)
(C.1c), (C.1e), (C.1f)	Basic - Infeasible	Violates (C.1d)
(C.1c), (C.1e), (C.1g)	Non-Basic	
(C.1c), (C.1f), (C.1g)	Basic - Infeasible	Violates (C.1d)
(C.1d), (C.1e), (C.1f)	Basic - Feasible	$[\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma, \Gamma, \Gamma]$
(C.1d), (C.1e), (C.1g)	Basic - Feasible	$[\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma, \Gamma, \Gamma]$
(C.1d), (C.1f), (C.1g)	Basic - Feasible	$[\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma, \Gamma, \Gamma]$
(C.1e), (C.1f), (C.1g)	Basic - Feasible	$[\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma, \Gamma, \Gamma]$

Summary

The three part type extreme shift schedules are as follows:

$t(A)$	$t(B)$	$t(C)$
0	0	0
0	0	$\Omega - \Delta_{AB} - \Delta_{BC} - \Gamma$
0	$\Omega - \Delta_{AB} - \Delta_{BC} - \Gamma$	0
0	$\Omega - \Delta_{AB} - \Delta_{BC} - \Gamma$	Γ
0	Γ	$\Omega - \Delta_{AB} - \Delta_{BC} - \Gamma$
$\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma$	0	0
$\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma$	0	Γ
$\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma$	Γ	Γ

C.2 Two part type shift schedules with a minimum duration of production required directly after each on-line dieset

We will assume that the shortest amount of time the last part type can be produced for is a μ , so that the indirect labor crew that completed the on-line dieset can ensure that the pressline is running properly. This section will address two part type shift schedules in this situation.

Two part type shift schedules start and finish with the pressline set up with different part types. These shift schedules contain exactly one changeover. Therefore a shift schedule, s , of this type would have the following parameters: $g_s = 1$, if $f_{js} = 1$ then $l_{js} = 0 \quad \forall j \in \mathbf{J}$, if $l_{js} = 1$ then $f_{js} = 0 \quad \forall j \in \mathbf{J}$, and $m_{js} = 0 \quad \forall j \in \mathbf{J}$. Let $t(p)$ denote the amount of time that part type p is produced. Part type A is always the first part type produced and B is always the last part type produced. Let Δ_{AB} represent amount of time to complete the on-line dieset from part type A to part type B. Let Γ denote the length of the off-line preparation. Just as in Chapter IV, in this section, we will continue to assume that a changeover must be fully contained

in a single shift. We will also assume that a changeover – on-line dieset and off-line preparation – can be contained within one shift (i.e. $\Delta_{AB} + \Gamma \leq \Omega$) and that the shortest amount of time the last part type can be produced is a shorter duration than the on-line preparation (i.e. $\mu \leq \Gamma$).

The constraints on these two part type shift schedule are as follows:

$$(C.2a) \quad t(A) \geq 0$$

$$(C.2b) \quad t(B) \geq \mu$$

$$(C.2c) \quad t(A) \leq \Omega - \Delta_{AB} - \Gamma$$

$$(C.2d) \quad t(A) + t(B) \leq \Omega - \Delta_{AB}$$

Constraint C.2a ensures that the production time of part type A is not negative. Constraint C.2b ensures that the production time of part type B is greater than μ . Constraint C.2c ensures that production of part type A ends with enough time for the changeover (on-line dieset and off-line preparation) to finish within the shift. Constraint C.2d states that the total production time cannot exceed the length of the shift minus the time required for the on-line dieset.

Active Constraints Table

Constraints	Result	Solution $[t(A), t(B)]$
(C.2a),(C.2b)	Basic - Feasible	$[0, \mu]$
(C.2a),(C.2c)	Non-Basic	
(C.2a),(C.2d)	Basic - Feasible	$[0, \Omega - \Delta_{AB}]$
(C.2b),(C.2c)	Basic - Feasible	$[\Omega - \Delta_{AB} - \Gamma, \mu]$
(C.2b),(C.2d)	Basic - Infeasible	Violates (C.2c)
(C.2c),(C.2d)	Basic - Feasible	$[\Omega - \Delta_{AB} - \Gamma, \Gamma]$

Summary

The two part type shift schedules are as follows:

$t(A)$	$t(B)$
0	μ
0	$\Omega - \Delta_{AB}$
$\Omega - \Delta_{AB} - \Gamma$	μ
$\Omega - \Delta_{AB} - \Gamma$	Γ

C.3 Three part type shift schedules with a minimum duration of production required directly after each on-line dieset

We will assume that the shortest amount of time the last part type can be produced for is a μ , so that the indirect labor crew that completed the on-line dieset can ensure that the pressline is running properly. This section will address three part type shift schedules in this situation.

These shift schedules have three part types set up during the shift. In this scenario, a part type can not be directly followed by itself and there are exactly two on-line diesets. Therefore a shift schedule, s , of this type would have the following parameters: $g_s = 2$, $f_{js} \neq m_{js} \quad \forall j \in \mathbf{J}$ and $m_{js} \neq l_{js} \quad \forall j \in \mathbf{J}$. Let $t(p)$ denote

the amount of time that part type p is produced. Assume that A is the first part type produced, B is the middle part type produced and C is the last part type produced. Therefore $f_{As} = 1$, $m_{Bs} = 1$, and $l_{Cs} = 1$. Let Δ_{ij} represent amount of time to complete the on-line dieset from part type i to part type j and Γ represent amount of time to complete the off-line preparation. Just as in Chapter IV, in this section, we will continue to assume that a changeover must be fully contained in a single shift. In addition, we assume that there is enough time within the shift to complete two changeovers – each with an on-line dieset and off-line preparation (i.e. $\Delta_{AB} + \Delta_{BC} + 2\Gamma \leq \Omega$) and that the shortest amount of time the last part type can be produced is a shorter duration than the on-line preparation (i.e. $\mu \leq \Gamma$).

The constraints on these three part type shift schedules are as follows:

$$(C.3a) \quad t(A) \geq 0$$

$$(C.3b) \quad t(B) \geq \mu$$

$$(C.3c) \quad t(C) \geq \mu$$

$$(C.3d) \quad t(A) \leq \Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma$$

$$(C.3e) \quad t(A) + t(B) \leq \Omega - \Delta_{AB} - \Delta_{BC} - \Gamma$$

$$(C.3f) \quad t(A) + t(C) \leq \Omega - \Delta_{AB} - \Delta_{BC} - \Gamma$$

$$(C.3g) \quad t(A) + t(B) + t(C) \leq \Omega - \Delta_{AB} - \Delta_{BC}$$

Many of these constraints are similar to those in Section C.1. Only constraints C.3b and C.3c are different. They state that the duration of work on a task (B , and C) must be greater than μ .

Active Constraint Table

Constraints	Result	Solution $[t(A), t(B), t(C)]$
(C.3a), (C.3b), (C.3c)	Basic-Feasible	$[0, \mu, \mu]$
(C.3a), (C.3b), (C.3d)	Non-Basic	
(C.3a), (C.3b), (C.3e)	Non-Basic	
(C.3a), (C.3b), (C.3f)	Basic-Feasible	$[0, \mu, \Omega - \Delta_{AB} - \Delta_{BC} - \Gamma]$
(C.3a), (C.3b), (C.3g)	Basic - Infeasible	Violates (C.3f)
(C.3a), (C.3c), (C.3d)	Non-Basic	
(C.3a), (C.3c), (C.3e)	Basic - Feasible	$[0, \Omega - \Delta_{AB} - \Delta_{BC} - \Gamma, \mu]$
(C.3a), (C.3c), (C.3f)	Non-Basic	
(C.3a), (C.3c), (C.3g)	Basic - Infeasible	Violates (C.3e)
(C.3a), (C.3d), (C.3e)	Non-Basic	
(C.3a), (C.3d), (C.3f)	Non-Basic	
(C.3a), (C.3d), (C.3g)	Non-Basic	
(C.3a), (C.3e), (C.3f)	Basic - Infeasible	Violates (C.3g)
(C.3a), (C.3e), (C.3g)	Basic - Feasible	$[0, \Omega - \Delta_{AB} - \Delta_{BC} - \Gamma, \Gamma]$
(C.3a), (C.3f), (C.3g)	Basic - Feasible	$[0, \Gamma, \Omega - \Delta_{AB} - \Delta_{BC} - \Gamma]$
(C.3b), (C.3c), (C.3d)	Basic - Feasible	$[\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma, \mu, \mu]$
(C.3b), (C.3c), (C.3e)	Basic - Infeasible	Violates (C.3d)
(C.3b), (C.3c), (C.3f)	Basic - Infeasible	Violates (C.3d)
(C.3b), (C.3c), (C.3g)	Basic - Infeasible	Violates (C.3d)
(C.3b), (C.3d), (C.3e)	Non-Basic	
(C.3b), (C.3d), (C.3f)	Basic - Feasible	$[\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma, \mu, \Gamma]$
(C.3b), (C.3d), (C.3g)	Basic - Infeasible	Violates (C.3f)
(C.3b), (C.3e), (C.3f)	Basic - Infeasible	Violates (C.3d)

(C.3b), (C.3e), (C.3g)	Basic - Infeasible	Violates (C.3d)
(C.3b), (C.3f), (C.3g)	Non-Basic	
(C.3c), (C.3d), (C.3e)	Basic - Feasible	$[\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma, \Gamma, \mu]$
(C.3c), (C.3d), (C.3f)	Non-Basic	
(C.3c), (C.3d), (C.3g)	Basic - Infeasible	Violates (C.3e)
(C.3c), (C.3e), (C.3f)	Basic - Infeasible	Violates (C.3d)
(C.3c), (C.3e), (C.3g)	Non-Basic	
(C.3c), (C.3f), (C.3g)	Basic - Infeasible	Violates (C.3d)
(C.3d), (C.3e), (C.3f)	Basic - Feasible	$[\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma, \Gamma, \Gamma]$
(C.3d), (C.3e), (C.3g)	Basic - Feasible	$[\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma, \Gamma, \Gamma]$
(C.3d), (C.3f), (C.3g)	Basic - Feasible	$[\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma, \Gamma, \Gamma]$
(C.3e), (C.3f), (C.3g)	Basic - Feasible	$[\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma, \Gamma, \Gamma]$

Summary

The three part type extreme shift schedules are as follows:

$t(A)$	$t(B)$	$t(C)$
0	μ	μ
0	μ	$\Omega - \Delta_{AB} - \Delta_{BC} - \Gamma$
0	$\Omega - \Delta_{AB} - \Delta_{BC} - \Gamma$	μ
0	$\Omega - \Delta_{AB} - \Delta_{BC} - \Gamma$	Γ
0	Γ	$\Omega - \Delta_{AB} - \Delta_{BC} - \Gamma$
$\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma$	μ	μ
$\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma$	μ	Γ
$\Omega - \Delta_{AB} - \Delta_{BC} - 2\Gamma$	Γ	Γ

BIBLIOGRAPHY

BIBLIOGRAPHY

- [1] Advantage Fabricated Metals, 3575 Morreim Drive, Belvidere, Illinois, USA 61008, Stamping Process.
- [2] H. K. Alfares, J. E. Bailey, Integrated project task and manpower scheduling, *IIE Transactions* 29 (9) (1997) 711–717.
- [3] L. H. Appelgren, A column generation algorithm for a ship scheduling problem, *Transportation Science* 3 (1) (1969) 53–68.
- [4] A. Armacost, C. Barnhart, K. Ware, Composite variable formulations for express shipment service network design, *Transportation Science* 35 (1) (2002) 1–29.
- [5] J. Atlason, M. A. Epelman, S. G. Henderson, Call center staffing with simulation and cutting plane methods, *Annals of Operations Research* 127 (1) (2004) 333–358.
- [6] K. Baker, Workforce allocation in cyclical scheduling problems: A survey, *Operational Research Quarterly* 27 (1) (1976) 155–167.
- [7] C. Barnhart, A. Cohn, E. Johnson, D. Klabjan, G. Nemhauser, P. Vance, Airline Crew Scheduling, chap. 14, 2nd ed., Kluwer’s International Series, 2003, pp. 493–521.
- [8] C. Barnhart, A. Farahat, M. Lohatepanont, Airline fleet assignment with enhanced revenue modeling, *Operations Research* (2008) 1–14.
- [9] C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, P. Vance, Branch-and-price: Column generation for solving huge integer programs, *Operations Research* (1998) 316–329.
- [10] C. Barnhart, T. Kniker, M. Lohatepanont, Itinerary-based fleet assignment, *Transportation Science* (2002) 199–217.
- [11] J. F. Benders, Partitioning procedures for solving mixed-variables programming problems, *Numerische Mathematik* 4 (1) (1962) 238 – 252.
- [12] D. Bernstein, M. Rodeh, I. Gertner, On the complexity of scheduling problems for parallel/pipelined machines, *IEEE Transactions on Computers* 38 (9) (1989) 1308–1313.
- [13] S. Binato, M. Pereira, S. Granville, A new benders decomposition approach to solve power transmission network design problems, *IEEE Transactions on Power Systems* 16 (2) (2001) 235–240.
- [14] R. Blau, N-job, one machine sequencing problems under uncertainty, *Management Science* (1973) 101–109.
- [15] J. Blazewicz, J. Lenstra, A. Rinnooy Kan, Scheduling subject to resource constraints: Classification and complexity, *Discrete Applied Mathematics* 5 (1) (1983) 11–24.
- [16] J. Blocher, S. Chand, K. Sengupta, The changeover scheduling problem with time and cost considerations: Analytical results and a forward algorithm, *Operations Research* (1999) 559–569.

- [17] P. Brucker, A. Drexl, R. Möhring, K. Neumann, E. Pesch, Resource-constrained project scheduling: Notation, classification, models, and methods, *European Journal of Operational Research* 112 (1) (1999) 3–41.
- [18] P. Brucker, A. Gladky, H. Hoogeveen, M. Y. Kovalyov, C. N. Potts, T. Tautenhahn, S. L. van de Velde, Scheduling a batching machine, *Journal of Scheduling* 1 (1) (1998) 31–54.
- [19] P. Brucker, T. Hilbig, J. Hurink, A branch and bound algorithm for a single-machine scheduling problem with positive and negative time-lags, *Discrete Applied Mathematics* (1999) 77–99.
- [20] E. Burch, M. Qiu, Hierarchical production planning and scheduling in a multi-product, multi-machine environment, *International Journal of Production Research* 35 (11) (1997) 3023–3042.
- [21] R. N. Burns, G. J. Koop, A modular approach to optimal multiple-shift manpower scheduling, *Oper. Res.* 35 (1) (1987) 100–110.
- [22] V. Caraffa, S. Ianes, T. Bagchi, C. Sriskandarajah, Minimizing makespan in an blocking flow-shop using genetic algorithms, *International Journal of Production Economics* (2001) 101–115.
- [23] B. Cheang, H. Li, A. Lim, B. Rodrigues, Nurse rostering problems—a bibliographic survey, *European Journal of Operational Research* 151 (3) (2003) 447 – 460.
- [24] E. Coffman, M. Yannakakis, M. Magazine, C. Santos, Batch sizing and job sequencing on a single machine, *Annals of Operations Research* (1990) 135–147.
- [25] A. Cohn, C. Barnhart, Improving crew scheduling by incorporating key maintenance routing decisions, *Operations Research* 51 (3) (2003) 387–396.
- [26] A. Cohn, C. Barnhart, Composite-variable modeling for service parts logistics, *Annals of Operations Research* 144 (1) (2006) 17–32.
- [27] A. Cohn, S. Root, A. Wang, D. Mohr, Integration of the load matching and routing problem with equipment balancing for small package carriers, *To appear in Transportation Science*.
- [28] T. Crainic, K. Rousseau, The column generation principle and the airline crew scheduling problem, *INFOR* (1987) 136–151.
- [29] H. Crauwels, C. Potts, L. Van Wassenhove, Local search heuristics for single machine scheduling with batch set-up times to minimize total weighted completion time, *Annals of Operations Research* (1997) 261–279.
- [30] J. R. Daduna, I. Branco, J. M. P. Paixio (eds.), Computer-aided transit scheduling : proceedings of the Sixth International Workshop on Computer-aided Scheduling of Public Transport, vol. 430, Springer, Berlin; New York, 1995 (1995).
- [31] G. Dantzig, P. Wolfe, The decomposition principle for linear programs, *Operations Research* (1960) 101–111.
- [32] M. Daskin, S. Melkote, An integrated model of facility location and transportation network design, *Transportation Research Part A: Policy and Practice* 35 (6) (2001) 515–538.
- [33] C. Delporte, L. Thomas, Lot sizing and sequencing for n products on one facility, *Management Science* (1977) 1070–1079.
- [34] G. Desaulniers, J. Desrosiers, Y. Dumas, M. Solomon, F. Soumis, Daily aircraft routing and scheduling, *Management Science* (1997) 841–855.
- [35] B. Dietrich, J. Lee, Y. Lee, Order selection on a single machine with high set-up costs, *Annals of Operations Research* (1993) 379–396.

- [36] G. Dobson, The cyclic lot scheduling problem with sequence-dependent setups, *Operations Research* (1992) 736–649.
- [37] G. Dobson, U. Karmarker, J. Rummel, Batching to minimize flow times on one machine, *Management Science* (1987) 784–799.
- [38] A. T. Ernst, H. Jiang, M. Krishnamoorthy, Staff scheduling and rostering: A review of applications, methods and models, *European Journal of Operational Research* 153 (2004) 3–27.
- [39] B. Faaland, T. Schmitt, Cost-based scheduling of workers and equipment in a fabrication and assembly shop, *Operations Research* 41 (2) (1993) 253–268.
- [40] B. Fleischmann, H. Meyr, The general lotsizing and scheduling problem, *OR Spektrum* (1997) 11–21.
- [41] H. Gabbay, Multi-stage production planning, *Management Science* 25 (11) (1979) 1138–1148.
- [42] N. Gans, G. Koole, A. Mandelbaum, Commissioned paper: Telephone call centers: Tutorial, review, and research prospects, *Manufacturing & Service Operations Management* 5 (2) (2003) 79–141.
- [43] M. Garey, D. Johnson, R. Sethi, The complexity of flowshop and jobshop scheduling, *Mathematics of Operations Research* 1 (2) (1976) 117–129.
- [44] A. Gascon, R. Leachman, A dynamic programming solution to the dynamic, multi-item, single-machine scheduling problem, *Operations Research* (1988) 50–56.
- [45] A. Geoffrion, Elements of large-scale mathematical programming, *Management Science* (1970) 652–691.
- [46] A. M. Geoffrion, Generalized benders decomposition, *Journal of Optimization Theory and Applications* 10 (4) (1972) 237–260.
- [47] A. Gerodimos, C. Glass, C. Potts, T. Tautenhahn, Scheduling multi-operation jobs on a single machine, *Annals of Operations Research* (1999) 87–105.
- [48] P. Gilmore, R. Gomory, A linear programming approach to the cutting stock problem, *Operations Research* 9 (1961) 849–859.
- [49] C. Glassey, Minimum change-over scheduling of several products on one machine, *Operations Research* (1968) 342–352.
- [50] J. E. Gomar, C. T. Haas, D. P. Morton, Assignment and allocation optimization of partially multiskilled workforce, *Journal of Construction Engineering and Management* 128 (2) (2002) 103–109.
- [51] S. Graves, A review of production scheduling, *Operations Research* (1981) 646–675.
- [52] O. Gusikhin, G. Rossi, Improving automotive supplier operations through information logistics, *The Knowledge Gap in Enterprise Information Flow* (2004) 81–94.
- [53] O. Gusikhin, G. Rossi, Well-connected: Mes data integration in the automotive supply chain, *APICS The Performance Advantage* (2005) 32–35.
- [54] J. Ho, Recent advances in the decomposition approach to linear programming, *Mathematical Programming Study* (1987) 119–128.
- [55] J. Hooker, G. Ottosson, Logic-based benders decomposition, *Mathematical Programming* 96 (1) (2003) 33–60.

- [56] T. Hu, Y. Kuo, F. Ruskey, Some optimum algorithms for scheduling problems with changeover costs, *Operations Research* (1987) 94–99.
- [57] B. Jaumard, F. Semet, T. Vovor, A generalized linear programming model for nurse scheduling, *European Journal of Operational Research* 107 (1) (1998) 1 – 18.
- [58] K. Jones, I. Lustig, J. Farvolden, W. Powell, Multicommodity network flows: The impact of formulation on decomposition, *Mathematical Programming* (1993) 95–117.
- [59] J. Kanet, V. Sridharan, Scheduling with inserted idle time: Problem taxonomy and literature review, *Operations Research* 48 (1) (2000) 99–110.
- [60] A. Klose, An lp-based heuristic for two-stage capacitated facility location problems, *The Journal of the Operational Research Society* 50 (2) (1999) 157–166.
- [61] M. Laguna, A heuristic for production scheduling and inventory control in the presence of sequence-dependent setup times, *IIE Transactions* (1999) 125–134.
- [62] A. H. Land, A. G. Doig, An automatic method for solving discrete programming problems, *Econometrica* 28 (3) (1960) 497–520.
- [63] R. Leachman, A. Gascon, A heuristic scheduling policy for multi-item single-machine production systems with time-varying, stochastic demands, *Management Science* (1988) 377–390.
- [64] C.-Y. Lee, G. Vairaktarakis, Workforce planning in mixed model assembly systems, *Operations Research* 45 (4) (1997) 553–567.
- [65] T. Magnanti, R. Vachani, A strong algorithm for production scheduling with changeover costs, *Operations Research* (1990) 456–473.
- [66] R. Martin, Large Scale Linear and Integer Optimization: A Unified Approach, Kluwer Academic Publishers, 1999.
- [67] W. Maxwell, On sequencing n jobs on one machine to minimize the number of late jobs, *Management Science* (1970) 295–297.
- [68] A. Mingozzi, V. Maniezzo, S. Ricciardelli, L. Bianco, An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation, *Management Science* 44 (5) (1998) 714–729.
- [69] C. Monma, Sequencing to minimize the maximum job cost, *Operations Research* (1980) 942–951.
- [70] C. Monma, C. Potts, On the complexity of scheduling with batch setup times, *Operations Research* 37 (5) (1989) 798–804.
- [71] E. Nowicki, S. Zdrzalka, Single machine scheduling with major and minor setup times: A tabu search approach, *The Journal of the Operational Research Society* (1996) 1054–1064.
- [72] S. Panwalkar, W. Iskander, A survey of scheduling rules, *Operations Research* (1977) 45–61.
- [73] M. Pinedo, Scheduling theory, algorithms, and systems second edition, 2nd ed., Prentice Hall, 2002.
- [74] C. Potts, M. Kovalyov, Scheduling with batching: A review, *European Journal of Operational Research* 120 (2) (2000) 228–249.
- [75] C. Potts, L. Van Wassenhove, Integrating scheduling with batching and lot-sizing: A review of algorithms and complexity, *The Journal of the Operational Research Society* 43 (5) (1992) 395–406.

- [76] R. S. Rasmussen, M. Trick, A benders approach for the constrained minimum break problem, *European Journal of Operational Research* 177 (1) (2007) 198–213.
- [77] M. Salomon, L. Kroon, R. Kuik, L. vanWassenhove, Some extensions of the discrete lotsizing and scheduling problem, *Management Science* (1991) 801–812.
- [78] J. Schutten, S. van de Velde, W. Zijm, Single-machine scheduling with release dates, due dates and family setup times, *Management Science* (1996) 1165–1174.
- [79] H. Sherali, W. Adams, A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems, *Discrete Applied Mathematics* (1994) 83–106.
- [80] United States of America Department of the Interior, 1849 C. Street N.W., Washington, DC, USA 20240, Workforce Planning Instruction Manual (2001).
- [81] F. Vanderbeck, L. Wolsey, An exact algorithm for ip column generation, *Operations Research Letters* (1996) 151–159.
- [82] R. B. Wallace, W. Whitt, A staffing algorithm for call centers with skill-based routing, *Manufacturing & Service Operations Management* 7 (4) (2005) 276–294.
- [83] S. Webster, K. Baker, Scheduling groups of jobs on a single machine, *Operations Research* (1995) 692–703.
- [84] O. Wight, Manufacturing resource planning: MRP II, Oliver Wight Essex Junction, VT, 1984.
- [85] D. Williams, A. Wirth, A new heuristic for a single machine scheduling problem with set-up times, *The Journal of the Operational Research Society* (1996) 175–180.
- [86] W. Yang, C. Liao, Survey of scheduling research involving setup times, *International Journal of Systems Science* (1999) 143–155.