Optimization Models and Algorithms for Workforce Scheduling with Uncertain Demand

by

Gurjot Dhaliwal

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Management Sciences

Waterloo, Ontario, Canada, 2011

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

A workforce plan states the number of workers required at any point in time. Efficient workforce plans can help companies achieve their organizational goals while keeping costs low. In ever increasing globalized work market, companies need a competitive edge over their competitors. A competitive edge can be achieved by lowering costs. Labour costs can be one of the significant costs faced by the companies. Efficient workforce plans can provide companies with a competitive edge by finding low cost options to meet customer demand.

This thesis studies the problem of determining the required number of workers when there are two categories of workers. Workers belonging to the first category are trained to work on one type of task (called *Specialized Workers*); whereas, workers in the second category are trained to work in all the tasks (called *Flexible Workers*). This thesis makes the following three main contributions.

First, it addresses this problem when the demand is deterministic and stochastic. Two different models for deterministic demand cases have been proposed. To study the effects of uncertain demand, techniques of Robust Optimization and Robust Mathematical Programming were used.

The thesis also investigates methods to solve large instances of this problem; some of the instances we considered have more than 600,000 variables and constraints. As most of the variables are integer, and objective function is nonlinear, a commercial solver was not able to solve the problem in one day. Initially, we tried to solve the problem by using Lagrangian relaxation and Outer approximation techniques but these approaches were not successful. Although effective in solving small problems, these tools were not able to generate a bound within run time limit for the large data set. A number of heuristics were proposed using projection techniques.

Finally this thesis develops a genetic algorithm to solve large instances of this problem. For the tested population, the genetic algorithm delivered results within 2-3% of optimal solution.

Acknowledgements

I will be always indebted to my family for their love and support. Without their wishes, it would not be possible for me to reach this far.

I would like to thank my supervisor, Dr. Ada Barlatt, for her instructions and invaluable assistance. Without her encouragement the work would not have begun nor would it have been completed. I will always appreciate her patience, motivation, support and advice towards shaping my career. Whatever worth this work may have, it owes much to her.

I am thankful to Dr. Amer Obeidi and Dr. Stanko Dimitrov for reading my thesis and pointing some important corrections. I am grateful to Mr. Rakesh Arora of Thunder Tools for providing the data for this study. I am indebted to Department of Management Sciences and University of Waterloo for the financial support, which makes it possible for me to be here and to carry out the research.

I would like to pay my special thanks to Dr. Rakesh Sagar, Dr. Makarand Kulkarni and Mr. Mumtaz Ahmed for guiding me through various phases of my academic life. I would like to mention the patient ear of my friend Mr. Apaar Sadhwani and his motivational talks that helped me to put my best effort in this thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Workforce Planning Problems

As mentioned in the abstract, labour costs can be one of the significant costs for an organization. Lowering these costs can help companies achieve competitive edge over their competitors. A workforce plan states the number of workers required at any point in time. Adopting efficient workforce plans can help organizations lower their labour costs and achieving a competitive edge.

1.1.1 Mathematical Programming

In the literature, mathematical programming has been used to solve workforce planning problems [1, 2, 3, 4]. In addition, these techniques form a base for many sophisticated workforce scheduling systems used in industry.

Mathematical programming translates the strategic objectives and constraints of an organization into equations. Examples of strategic objective function include:

- Minimization of the total number of workers;

- Minimization of the time when workers are idle;
- Maximization of the machine or resource utilization;
- Minimization of related costs such as hiring and laying off costs, regular wage or salary of a person, insurance and benefit related expenses.

Workforce planning problems could be constrained by the production time, due dates, demand quantity, union regulations and/or spatial constraints of a facility. In addition, industry specific complexities can be added resulting in new problems. Below, we present several examples of workforce planning problems found in the literature that address industry specific complexities using math programming.

Cyclic demand is an example of an industry specific complexity. Cyclic demand is when the schedule repeats but the demand varies from shift to shift. Baker et al. [1] studied the staff scheduling problem in a cyclical demand situation. They discuss a number of practical settings where these models are applied. One of the prominent applications is scheduling of telephone operators when demand for staff is not constant throughout the day. Burns and Narasimhan [2] considered the cyclic staff scheduling problem by examining a problem similar to Baker [1] but they consider limitations on the maximum number of consecutive working days and include meal breaks.

Brusco and Jacobs [3] discussed another form of industry specific staff scheduling problem, called the tour scheduling problem. The tour scheduling problem is the assignment of employees to particular tours, where a tour is a schedule having specified work start time, meal break time and days off work. In [3], Brusco and Jacobs presented two integer programming models for the continuous tour scheduling problem containing meal break windows, overlapping start time bands and start time intervals.

Capturing employee preferences for a particular schedule is another industry specific complexity. Arthur and Ravindaran [4] considered worker preferences in their goal

programming based model for the nurse scheduling problem. The goals of the model included meeting minimum staff requirement and nurses' preference towards a particular schedule.

1.1.2 Heuristics

There are some situations when it is impossible to find an exact solution using mathematical programming. Chapter 5 of this thesis discusses this issue. For situations where mathematical programming can not find solutions, various heuristics and algorithms are used.

Morris and Showalter [6] presented a set covering formulation for scheduling the work force. The objective is to minimize the number of workers for all shifts, while enforcing the staff requirements. They use a Linear Programming (LP) relaxation based heuristic to solve the model. Bartholdi [8] solves a cyclical work force scheduling problem using a heuristic based on LP-relaxation techniques.

The case study described in this thesis was intractable due to the large number of constraints and variables of the model. Martello and Toth [5] overcome a similar issue in a bus driver scheduling problem by using a greedy algorithm. Lau [7] discussed a shift assignment problem in which large instances were hard to solve. He reduced an equivalent problem from a boolean satisfiability problem (3SAT) and presented a greedy algorithm for solving some restricted version of the problem.

1.2 Problem Statement

This thesis considers a manufacturing setting with uncertain demand across several product types. The manufacturing plant is composed of several workstations each dedicated to a specific task. For completion, every product must follow a predetermined sequence

of the workstations, where a set of workers are required to complete the tasks. We consider a setting with two types of workers: *specialized workers* who are only capable of working at a single workstation and *flexible workers* who are capable of working at any workstation. Workers are hired for shift types (i.e. the first, second or third shift of the day). Due to union regulations, workers cannot be hired and fired, thus, each worker assigned to each shift type is available for that shift type every day in the planning horizon.

The goal is to find a schedule that minimizes the workforce costs. Three models are described for the deterministic demand situations. The strengths and weaknesses of each model are highlighted. In order to cater to uncertain demand situations, approaches of Robust Optimization and Robust Mathematical Programming have been used. The next section details the both of the methodologies. Various models, reflecting the industry specific needs have been presented in Chapter 4.

For large problem instances, the model became intractable. A number of heuristics and approaches were developed in an attempt to solve the large instance. In addition, the classical approaches Lagrangian Relaxation and Outer Approximation were applied to the model. In the end, a Genetic algorithm based approach was successful in solving the large instance.

1.3 Related Works

In this section, the literature is explored for related work. The section is further divided into two subsections. The first section describes the related problems discussed by other researchers and emphasizes the importance of cross-training of workers. Cross-training ensures that workers are trained to work for different tasks. Therefore, cross-trained workers are similar to the flexible workers discussed earlier. The second section is a comprehensive list of techniques to deal with uncertainty in the parameters.

1.3.1 Workforce Planning Models

Ernst et al. [9] presented a detailed list of personnel scheduling and rostering problems in industry. They identified workforce planning as one of the classifications in rostering problems. One of the aspects of their work is a discussion on task based demand modeling, in which the number of workers required and the duration for the particular task is used to determine the number of staff members required. A similar philosophy has been applied in the deterministic models of this thesis.

Tien and Kamiyama [10] took a different approach by deconstructing a generalized manpower scheduling problem into five different but interrelated problems. A manager can either choose the whole framework of five problems or the specific ones serving the needs of an organization. This problem considered in this thesis can be categorized in Temporal Manpower requirement stage as described in [10].

Glover and Mcmillian [11] presented the general employee scheduling problems. Similar to this thesis, they have also considered two types of workers but divided them as *Full-time* and *Part-time* workers. All of the workers considered in this thesis have a full-time schedule, thus differentiating this problem from the one discussed in [9]. Apart from full-time/part-time employee constraints, Glover and Mcmillian [11] have also considered the difference in performance of the workers working on the same task and constraining the different time blocks. They implemented an integer programming approach and solutions were applied in a real world restaurant setting.

The paper that is most similar to this thesis is by Cai and Li [12]. They considered three types of workers in their model, where *type 1* and *type 2* workers have *skill one* and *skill two*, respectively, and *type 3* workers can perform both of the tasks. The cost for assigning one worker to a particular schedule is independent of the skill type. The model in [12] considers three different functions as objective, whereas our work considers only a single objective of minimizing the cost. The constraints in the model in [12] refer to

manpower demands over time, whereas the demand of product is an input to our model. Another important difference is the decision variables. Our model outputs *production schedule* (schedule determining the number of units to produce at each time point) and the number of workers required to finish that production, whereas the model in [12] considers feasible *work schedule* set (schedule telling the working days and time to start each shift) and considers the number of workers working on that schedule. Further, Cai and Li consider a limit on the number of workers to be hired. Our model does not limit the number of workers that can be hired.

By considering flexible workers, this thesis highlights cross-training as an important ideology. According to Hopp et al. [15], cross-training offers advantages along competitive dimensions of cost, time, quality and variety. Iravani, Van Oyen and Sims [13] discuss how an organization can be more flexible by implementing cross-training of the workers. Brusco and Johns [14] presented a mathematical model for evaluating cross-training configurations at the policy level. Wirojanagud et al. [16] presents a mathematical model considering inherent differences in learning new skills and the ability to perform tasks. They considered a different set of workers in a job shop environment. The decisions to be taken are related to hiring, layoff, cross-training, the worker assignment, and the amount of demand to be satisfied. They considered only deterministic demand situations, whereas our work focuses on uncertain demand situations. They considered different levels for different workers for each skill set and termed these levels as General Cognitive Ability sets. Instead of having an aggregate demand, they considered demand for a particular skill in a period.

1.3.2 Uncertain Demand

This thesis studies workforce planning problems having uncertain demand. According to Jeang [18], one way to deal with uncertainty is to hire part-time workers and to pro-

vide overtime to the existing employees. He used the above technique for scheduling nurses in a hospital. In addition to hiring part-time and over-time workers, researchers have mentioned mathematical techniques to deal with the uncertainty in demand. Nikolas [19] presented various approaches to deal with uncertainty in the parameters. The modern philosophies to handle the precarious nature of data include decision making by minimizing the expected value of an objective function (*Stochastic Programming*), by minimizing the maximum costs (*Robust Optimization*)[19], or by allowing infeasibility in some constraints (*Robust Mathematical Programming*) [27].

Stochastic Programming

Stochastic Programming is based on minimizing the expected value of an objective function over an uncertain problem data. The standard two-stage stochastic model divides the decision variables in two sets. The first set of the variables are independent of the uncertain nature of the parameters, and have to be decided before random fluctuation occurs. Once a particular realization of the parameters occur, the second set of variables, recourse variables, are decided with an additional cost to the objective function value of the first set of variables.

These techniques have been used in workforce planning problems. Campbell [20] used two-stage stochastic model to schedule and allocate the cross-trained workers in a multi-department service environment. He considered uncertainty in the demand parameter. The first decision stage is to determine the off days for a fixed set of cross trained workers. The recourse action in their model is to allocate the available workers to the demand realizations. Bard, Morton and Wang [21] presented a two-stage stochastic model with recourse to schedule and allocation of the workforce at a US Postal Service facility. The first stage model determines the number of permanent and temporary workers required in a regular work schedule. The second stage allocates the workers to specific shifts and uses overtime or casual labour to satisfy excess demand.

Robust Optimization

Robust Optimization is similar to stochastic optimization. In stochastic optimization, we input the probability distribution and try to minimize the expected value. Instead of the initial probability distribution, Robust Optimization considers an uncertainty set which contains all the instances that an uncertain parameter can take and the model is solved for the worst data case. The worst data case is the instance incurring maximum penalty or minimum performance. The technique of Robust Optimization guarantees a feasible solution for all data instances and chooses the solution corresponding to the worst data case. This technique is similar to Stochastic Programming but does not require probability distribution of the instances.

The first work on Robust Optimization is done by Soyster [22], who named the formulation as *inexact formulation*. In inexact formulation, the uncertainty lies in the constraint matrix and the instances belong to a convex set. Instead of solving for a single instance, the convex set of all the instances is solved. The solution obtained is feasible for all the data instances.

Ben and Nemirovsky [23] coined the term "Robust Optimization" in their paper. They considered uncertainty sets for the data sets to be ellipsoid and involve conic quadratic problems. As such, those methods cannot be directly applied to problems involving integers. Kouvelis et al. [24] used Robust Optimization techniques for solving problems considering two machine flow shop. Their model accounts for uncertainty in the processing times and minimizes the maximum regret associated with a schedule. Dimitrov [25], in Chapter 2 of his dissertation, applied Robust Optimization to find an effective routing policy for a network having active congestion control policies, in an uncertain demand environment. Their model maximizes the minimum performance. The approach of minimizing the worst performance can convert some polynomially solvable problems NP-Hard [26]. Berstimas [26] proposed robust formulation for integer variables

which are flexible in terms of conservatism and computationally tractable in practical and theoretical terms. The techniques of Robust Optimization have been applied in portfolio optimization, supply chain management and inventory management.

Robust Mathematical Programming

Mulvay [27] presented a Robust Mathematical Programming approach for dealing with the uncertain parameters. Instead of modeling for the worst data instance, Robust Mathematical Programming approach allows for infeasibility in some constraints. A penalty term for each of the violated constraints is added to the objective function and an optimal solution is found minimizing the penalty function. Similar to stochastic optimization, the variables are divided into two sets, *design variables* and *control variables*. Design variables are the variables whose optimal values are not affected by particular realization of data. Control variables are the ones adjusted when uncertain parameters are realized. Instead of considering a single realization, an expected value of the objective function is calculated. Higher moment terms and penalty terms for violation of the constraints are added to the objective function. The higher moment terms can lead to a nonlinear objective function. In order to address the computational difficulty related to the nonlinear objective function, Yu and Li [28] proposed using absolute value function in place of having quadratic function for variance reduction. List et al. [29] applied Robust Mathematical Programming techniques for fleet planning under uncertainty. Instead of using variance, they minimized the upper partial moment of order 1. For workforce scheduling problems, this thesis is one of the first works to apply the Robust Mathematical Programming approach.

1.4 Structure of the Thesis

The thesis is divided into a number of chapters. For easy readability, this section discusses the structure of the chapters and how the models are presented. Chapter 1 introduces the problem and presents several real world applications related to workforce scheduling. The chapter ends with a detailed literature review related to the problem.

Chapter 2 describe the first model to solve the problem. The chapter starts with a deterministic model followed by two robust models. The two robust models correspond to Robust Optimization and Robust Mathematical Programming approaches. Each of the three models has its own set of assumptions, notation and results. In Chapter 2, we assumed that all of the tasks required for a product can be completed simultaneously. In the remaining Chapters we relax this assumption as it is not valid for many practical settings.

Chapter 3 presents a deterministic model that can capture precedence relationships between tasks. The notation and results for this model are presented in this chapter.

Chapter 4 extends the deterministic model in Chapter 3 to consider uncertainty in demand. The chapter includes new notation for this model. In addition, this chapter presents several cases to reflect different industry specific needs. Computational results are presented for each case.

Chapter 5 opens with a discussion of a real world large problem instance. Classical heuristic methods as well as an independent heuristic were tested for this instance but were not successful. The main focus of this chapter is Genetic Algorithm based heuristic. In order to calculate the number of workers, required for a given production schedule, a network model is presented. This network model serve as the final step of Genetic algorithm. The Genetic Algorithm was successful in solving the real world instance and the results are at the end of the chapter.

Chapter 6 concludes the findings and suggest future extensions of the work.

Chapter 2

Minimizing the worker costs model

This chapter starts with the discussion of the first mathematical model to address the problem mentioned in section 1.2. The deterministic model determines the number of worker hours to meet demand and demand constraints. To consider uncertainty in the demand, two different models based on Robust Mathematical Programming and Robust Optimization techniques are presented. In Robust Mathematical Programming model, one of the constraints from deterministic model is allowed to be violated. The violation is penalized by a linear penalty function and is quadratic in nature. The chapter ends with the Minimize Maximum Penalty formulation.

2.1 Simultaneous Task Production Model

As mentioned in Chapter 1, a manufacturing setting is considered. A series of work-stations/tasks must be visited to produce products to meet the demand. The demand is assumed to be deterministic. We need to determine the number of *specialized* and *flexible* workers required to make the quantity demanded of each product.

We have decided to model this problem using a *task based modeling* approach. For every task, we will consider the total time required to meet the demand of all the products

requiring that particular task. We refer to this as the *task-demand hour*. To calculate this value we multiply the task duration (the processing time for one unit at the workstation) and the total demand for the products that require that task. In this model, we assume that the processing time at each workstation is the same for all products.

The total available hours of *specialized* and *flexible* workers assigned to each task, should be greater than or equal to the *task-demand hour* of that task. This ensures that the workers will be available to complete the production. The number of workers required for each type can be calculated using the total available worker hours.

The model is expressed as a Mixed Integer Linear Program (MILP) and the objective is to minimize the total worker cost. It is assumed that the cost per worker includes hiring, firing and all other related costs. Cost associated with each type of the worker is a parameter.

The model presented in this chapter is a simplified version of the general problem presented in Chapter 1; we acknowledge that this model has several unrealistic assumptions and we will relax several of these assumptions later in the Thesis.

2.1.1 Assumptions for Simultaneous Task Production Model

1. The planning horizon is one time period.
2. Each product has a fixed set of tasks required to complete production.
3. Each task/workstation is visited at most once to produce each product.
4. The tasks can be completed simultaneously.
5. All workers complete tasks at the same pace.

2.1.2 Notation: Simultaneous Task Production Model

Sets:

- Tasks: $\mathbf{A}$
- Products : $\mathbf{P}$
- Type of workers: $\mathbf{Y}$
- Set of products that require task a : $\mathbf{B_a}$.

Parameters:

- Demand of product p : d_p
- Number of hours required to finish one unit of task a : h_a
- Cost of one specialized worker: c^s
- Cost of one flexible worker: c^f
- Duration of a shift: u

Decision Variables:

- Number of specialized workers for task a: w_a^s
- Number of flexible workers: w^f
- Number of specialized worker hours for task a: w_a^{hs}
- Number of flexible worker hours for task a: w_a^{hf}

Objective Function:

$$\min \quad c^s \sum_{a \in \mathbf{A}} w_a^s + c^f w^f \tag{2.1}$$

Constraints

$$w_a^{hs} + w_a^{hf} \geq h_a \sum_{j \in \mathbf{B_a}} d_j \qquad \forall a \in \mathbf{A} \tag{2.2}$$

$$u w_a^s \geq w_a^{hs} \qquad \forall a \in \mathbf{A} \tag{2.3}$$

$$u w^f \geq \sum_{a \in \mathbf{A}} w_a^{hf} \tag{2.4}$$

$$w_a^s, w^f \in \mathbf{Z}^+ \tag{2.5}$$

The objective function (2.1), is the minimization of total worker costs, which is the sum of the total number of workers multiplied by the corresponding cost parameter. As w^f denotes the number of flexible workers for all the tasks, summation in the objective function is only for w_a^s variables. Constraint (2.2) refers to the demand satisfaction. For a particular task a, the *task-demand hour* is the sum of the number of units demanded over all the products p, that require task a, multiplied by the time required to finish task a. In order to meet demand, the total number of available worker hours should be at least equal to the *task-demand hour*. Constraints (2.3) and (2.4) translate required worker hours into number of workers required. The left hand side of constraint (2.3) converts the number of specialized workers required for task a into hours those workers can work in a shift u hours long. Constraint (2.4) completes this calculation for flexible workers. Constraints (2.3) and (2.4) require that the number of workers of each type should be greater than or equal to the number of hours required for that type. Constraint (2.5) requires the number of workers to be integer.

2.1.3 Implementation of Simultaneous Task Production Model

The model has been implemented in two scenarios. In Case 1, only one product and three tasks are considered. In Case 2, three products and ten tasks are considered. The model is solved using Excel solver for a certain set of parameters[1]. In both cases, the cost of a specialized worker is less than that of the flexible worker.

2.1.4 Discussion

As the cost of a specialized worker is less than that of a flexible worker, it is logical to have more specialized workers. The results are consistent with this statement; in Case 1 and Case 2, where the number of flexible workers is less than the specialized workers.

The model's biggest shortcoming is that it does not capture the situations when there are precedence relationships between tasks to produce a product. In other words, if product production follows a particular sequence of tasks and requires certain set of tasks to be finished before moving to the next task, this model will not produce useful results.

2.2 Robust Mathematical Programming Model

We consider the same manufacturing setting as in deterministic model. In this section, we extend the deterministic model and present a Robust Mathematical Programming model to cater to the uncertainty in quantity demanded for a product. At each point of time in the planning horizon, the demand can have any value from a set of discrete numbers. The proposed model hedges the uncertainty in the quantity demanded and gives a single number for the number of workers that should be hired, irrespective of the realized demand. The hired number of workers will not be able to meet demand for some

[1]See "(Appendix A)" for the data sets

instances and will be idle in other instances. To minimize the loss because of these two factors, we propose a model based on Robust Mathematical Programming. A certain set of constraints are allowed to be violated and corresponding penalty terms, proportional to the amount of constraint violations, are added in the objective function. The detailed model is explained below.

We start by defining the uncertainty set. For a particular product, the uncertainty set contains all of its demand instances. We refer to a single possible demand instance of a product as a situation. So the cardinality of the uncertainty set is the number of possible demand situations of a particular product. As demand is situation based, the appropriate parameters have to be changed in deterministic model. Constraint (2.2) of deterministic model translates the *task-demand hour* into available worker hours. In this model more than one demand situation are considered, so constraint (2.2) has to be modified accordingly. The set of variables ψ_{as} for every task a and situation s have been introduced for calculating the violation of demand constraint, and their squared value is minimized in the objective function.

2.2.1 Notation: Robust Mathematical Programming Model

Sets:

All of the sets as mentioned in the deterministic model and

- Cardinality of uncertainty set of a product p: **S**

Parameters:

All the parameters mentioned in the deterministic model. The following parameter is removed in this model

- Demand of product $p : d_p$

Instead, we add the following parameters

- Demand of product p in situation s : d_{ps}
- Probability of having a situation s : p_s
- Penalty cost for constraint violation: γ

Decision Variables:

In addition to the decision variables in the deterministic model, we add the following variables.

- Difference between the amount of worker hours for task a and the *task-demand hours* for all the products requiring task a, in situation s: ψ_{as}

Objective Function:

$$\min \quad c^s \sum_{a \in \mathbf{A}} w_a^s + c^f w^f + \sum_{p \in \mathbf{P}, a \in \mathbf{A}, s \in \mathbf{S}} \gamma p_s \psi_{as}^2 \tag{2.6}$$

Constraints:

The constraint (2.2) is replaced by the following constraint, keeping all the other constraints the same.

$$w_a^{hs} + w_a^{hf} - \psi_{as} = h_a \sum_{j \in G_a} d_{js} \qquad \forall a \in \mathbf{A}, s \in \mathbf{S} \tag{2.7}$$

In this model, we allow deviations in demand satisfaction constraint (2.2). In the deterministic model, the total number of worker hours has to be at least equal to the *task-demand hours*. In the Robust Mathematical Programming model the total number of worker hours can be less than the *task-demand hours*. The difference between the *task-demand hours* and the worker hours is captured by the variables ψ_{as}, which is minimized in objective function (2.6). The rest of the constraints are same as deterministic model.

2.2.2 Implementation of Robust Mathematical Programming Model

We implemented the model in Python and solved it using CPLEX solver. The results are presented for the one product and the three task data [2]. Two separate cases are considered. Case 1 has very low variability in the demand for different situations relative to Case 2.

2.2.3 Results and Conclusion

	Probability function	$w^s_{Task_1}$	$w^s_{Task_2}$	$w^s_{Task_3}$	w^f	Worker Costs	Penalty
γ =3	$p_1 = 0.9, p_2 = 0.1$	0	0	0	0	0	16.2
	$p_1 = 0.1, p_2 = 0.9$	0	0	0	1	15	3.4
	$p_1 = p_2 = 0.5$	0	0	0	1	15	9
γ =13	$p_1 = 0.9, p_2 = 0.1$	0	0	0	1	15	14.04
	$p_1 = 0.1, p_2 = 0.9$	0	0	0	1	15	14.733
	$p_1 = p_2 = 0.5$	0	0	0	1	15	39
γ =100	$p_1 = 0.9, p_2 = 0.1$	0	0	0	1	15	108
	$p_1 = 0.1, p_2 = 0.9$	0	0	0	1	15	113.13
	$p_1 = p_2 = 0.5$	0	0	0	1	15	300

Table 2.1: Robust Mathematical Programming Model : Low Demand Variation Case

Table 2.1 displays the computational results for the case having low variability in the demand values. The first column lists the different γ values considered. Different probability functions are considered for each value of γ, as detailed in the second column. In the second column, p_s is the probability of occurrence for situation s. As only two situations are considered, the sum of p_1 and p_2 should be equal to 1. The third, fourth and fifth columns lists the number of specialized workers working on each of the three tasks. The numbers of flexible workers hired for different values of γ and p_s are listed in the sixth column. The last columns display the total cost of the workers and the penalty for each values of γ and p_s.

[2] See "(Appendix B)" for the data sets

Results in Table 2.1 show that for $\gamma = 13$, and 100, the number of workers hired remains the same but with a different total penalty value. And the maximum value of this penalty occurs when the probability of occurrence is equal for every situation i.e variance is highest. This is expected as both demand situations have equal weights and thus each constraint has an equal absolute amount of violation. But when the probability of both situations is not equal, the constraints corresponding to the higher probable situation have lower absolute amount of violations. For $\gamma = 3$, the model returns *Zero Production* as one of the solutions. The reason for this is the low demand variability, high probability for demand of one unit, and low penalty for constraint violation in this instatnce.

Table 2.2, displays the computational results for the case having high variability in demand values. Each column lists the same parameters and variables as Table 2.1, but values correspond to Case 2.

It can be easily inferred from Table 2.2 that solution for Case 2 has a similar pattern as Case 1, except at $\gamma = 3$. As demand variation is high, instead of having *Zero Production* or hiring zero number of workers, the model chooses to hire one flexible worker. It is also observed for this instance that when the probability function is the same the same solution is obtained, independent of the γ values.

	Probability function	$w^s_{Task_1}$	$w^s_{Task_2}$	$w^s_{Task_3}$	w^f	Worker Costs	Penalty
γ =3	$p_1 = 0.9, p_2 = 0.1$	0	0	0	1	15	158.76
	$p_1 = 0.1, p_2 = 0.9$	2	2	2	0	60	158.76
	$p_1 = p_2 = 0.5$	1	1	1	0	30	441
γ =13	$p_1 = 0.9, p_2 = 0.1$	0	0	0	1	15	687.96
	$p_1 = 0.1, p_2 = 0.9$	2	2	2	0	60	687.96
	$p_1 = p_2 = 0.5$	1	1	1	0	30	1911
γ =100	$p_1 = 0.9, p_2 = 0.1$	0	0	0	1	15	5292
	$p_1 = 0.1, p_2 = 0.9$	2	2	2	0	60	5292
	$p_1 = p_2 = 0.5$	1	1	1	0	30	14700

Table 2.2: Robust Mathematical Programming Model : High Demand Variation Case

2.3 Minimize Maximum Penalty Model

For the same manufacturing setting discussed at the beginning of this chapter, we present another model based on Robust Optimization technique. As mentioned in Chapter 1, Robust Optimization is another way to consider uncertainty in the parameters. Similar to Robust Mathematical Programming, Robust Optimization considers an uncertainty set. Robust Optimization models for the worst case and one way to do it is by minimizing the maximum penalty. In this model, we also use a minimize maximum penalty approach, a penalty value is attached to each of them for underage and overage. The objective is to find the number of workers hired that minimizes the sum of worker costs and penalties.

2.3.1 Assumptions for Minimize Maximum Penalty Model

For having a simple problem structure, the additional assumptions for this model are:

1. For the situations when demand hours are not equal to the worker hours, we attach penalty costs.

2. Penalties for overage and underage do not change over time.

The goal of this model is still to determine the number of specialized workers and the number of flexible workers required. We still utilize the *task-based modeling* approach. And since, demand is situation based, the *task-demand hour* has to be calculated for each demand situation. The goal is to determine a single value for the number of workers to hire, irrespective of the realized value of demand. Thus, for some demand situations the corresponding *task-demand hour* will not be equal to the available worker hours. For these situations, we need to calculate overage (number of hours in excess) and underage (number of hours less than that of required). Both overage and underage will be penalized with the linear functions and summed together to get total penalty. This penalty is added

to the worker costs in the objective. In this model, the planning horizon has multiple time periods.

In mathematical terms, the above discussion will imply the violation of constraint (2.2). Constraint (2.2) of deterministic model translates the *task-demand hour* into available worker hours. The total number of hours for all the workers can be different than the required demand hours for every situation. This difference is penalized by linear weight function and maximum of this penalty is minimized in the model. This is how our approach differs from Robust Optimization, in our Minimize Maximum Penalty approach a constraint can be violated, while with Robust Optimization guarantees a feasible solution for all data instances.

2.3.2 Notation: Minimize Maximum Penalty Model

Sets:

- Time : **T**
- Tasks: **A**
- Products : **P**
- Type of workers: **Y**
- Number of days: **E**
- Number of situations on a particular day e: $\mathbf{S_e}$

Parameters:

- Demand value on observing situation s, for product p, on day e: d_{eps}
- Number of hours required to finish one unit of task a : h_a

- Cost of one specialized worker for task a : c_a^s
- Cost of one flexible worker: c^f
- Duration of a shift: u
- Penalty when production of product p is in excess of its demand: η_p
- Penalty when production of product p is less than the demand: β_p

Decision Variables:

- Number of specialized workers required for task a, on day e, for product p : w_{epa}^s
- Number of flexible workers required on day e, for product p: w_{ep}^f
- Number of specialized worker hours required for task a, on day e, for product p : w_{epa}^{hs}
- Number of flexible worker hours, required on day e for task a and product p : w_{epa}^{hf}
- Penalty cost for demand situation $s, s \in S_e$, on day e and product p : pen_{eps}
- Maximum penalty for product p: θ_p
- Number of demand hours of task a for product p less than the optimal solution on day e, for situation s : q_{epas}^-
- Number of demand hours of task a for product p more than the optimal solution on day e, for situation s : q_{eas}^+
- Number of flexible workers required to make product p for whole of the planning horizon of $|\mathbf{E}|$ days: w_p^f
- Number of specialized workers for task a required to make product p,for the whole planning horizon of $|\mathbf{E}|$ days: w_{pa}^s

Objective Function:

$$\min \sum_{a \in \mathbf{A}, p \in \mathbf{P}} c_a^s w_{pa}^s + \sum_{p \in \mathbf{P}} c^f w_p^f + \sum_{p \in \mathbf{P}} \theta_p \tag{2.8}$$

Constraints:

$$\theta_p \geq pen_{eps} \qquad \forall e \in \mathbf{E}, p \in \mathbf{P}, s \in \mathbf{S_{es}} \tag{2.9}$$

$$w_{epa}^{hs} + w_{epa}^{hf} - d_{eps} h_{pa} = q_{epas}^{+} - q_{epas}^{-} \qquad \forall a \in \mathbf{A}, e \in \mathbf{E}, p \in \mathbf{P}, s \in \mathbf{S_e} \tag{2.10}$$

$$pen_{eps} = \sum_{a \in \mathbf{A}} \eta_p q_{epas}^{+} + \beta_p q_{epas}^{-} \qquad \forall e \in \mathbf{E}, p \in \mathbf{P}, s \in \mathbf{S_e} \tag{2.11}$$

$$uw_{epa}^{s} \geq w_{epa}^{hs} \qquad \forall a \in \mathbf{A}, p \in \mathbf{P}, e \in \mathbf{E} \tag{2.12}$$

$$uw_{ep}^{f} \geq \sum_{a \in \mathbf{A}} w_{epa}^{hf} \qquad \forall p \in \mathbf{P}, e \in \mathbf{E} \tag{2.13}$$

$$w_{pa}^{s} \geq w_{epa}^{s} \qquad \forall e \in \mathbf{E}, p \in \mathbf{P}, a \in \mathbf{A} \tag{2.14}$$

$$w_{p}^{f} \geq w_{ep}^{f} \qquad \forall e \in \mathbf{E}, p \in \mathbf{P} \tag{2.15}$$

$$w_{epa}^{s}, w_{ep}^{f} \in \mathbb{Z}^{+} \tag{2.16}$$

$$q_{epas}^{+}, q_{epas}^{-} \geq 0 \tag{2.17}$$

The objective function minimizes the total worker costs and the total penalty of either exceeding or not meeting the demand. Constraint (2.9) states that overall penalty for product p, θ_p has to be at least equal to the maximum penalty over every day e and situation s. Constraint (2.10) calculates the underage and overage of the demand hours for every product p. Based on the constraint structure only one of q_{epas}^{-} and q_{epas}^{+} will be non-zero. The penalty for overage in demand hours of product p, for task a is a product of the number of units exceeding demand, q_{epas}^{+} on day e, and unit cost of penalty for

overage, η_p. Similarly, the penalty for not meeting the demand is a multiplicative sum of q^-_{epas} and β_p . As stated in constraint (2.11), the total penalty for product p on day e and situation s should be the sum of penalties for overage and underage, summed over all tasks a.

Constraints (2.12) and (2.13) calculate the number of specialized and flexible workers required on day e for product p. The left hand side of constraint (2.12) is the total number of specialized worker hours required for task a, which equals the multiplication of hours in a shift u and the number of specialized workers required for task a, w^s_{epa}. This multiplied term should be greater than or equal to the number of specialized worker hours required on day e at task a for product p. The left hand side of constraint (2.13) calculates the number of flexible workers required on day e for product p as the multiplication of the number of hours in a shift u and the total number of flexible workers required w^f_{ep}. This multiplied term should be at least equal to the flexible worker hours required on day e for product p. Constraint (2.14) and (2.15) calculate the number of specialized and flexible workers required for the planning horizon of E days. Constraint (2.16) states the integer requirements of the number of workers.

2.3.3 Results and Conclusions

We implemented the model in Python and solved it using CPLEX solver. The following results are presented for one product, one day planning horizon and requiring three tasks to be finished. The demand of the product is one of two numbers i.e. the cardinality of demand set is 2. We present results for two different cases, where the first case has a very low variance in the demand values compared to the second. [3]

Table 2.3 displays the computational results for the case having low variance in the demand values. The first column lists the decision variables of this model. Subsequent

[3]See "(Appendix B)"

Decision variables	$\eta = \beta = 5$	$\eta = 200, \beta = 5$	$\eta = 200, \beta = 20$	$\eta = \beta = 200$	$\eta = 5, \beta = 200$
$w^s_{1,Task1}$	0	0	0	0	0
$w^s_{1,Task2}$	0	0	0	0	0
$w^s_{1,Task3}$	0	0	0	0	0
w^f_1	1	1	1	1	1
$q^+_{1,1,Task1,1}$	1	0	0.27	0	0.93
$q^+_{1,1,Task1,2}$	0	0	0	0	0
$q^+_{1,1,Task2,1}$	0.5	0.073	0	0.5	1
$q^+_{1,1,Task2,2}$	0	0	0	0	0
$q^+_{1,1,Task3,1}$	0	0	0	1	1
$q^+_{1,1,Task3,2}$	0	0	0	0	0
$q^-_{1,1,Task1,1}$	0	0	0	0	0
$q^-_{1,1,Task1,2}$	0	1	0.73	1	0.073
$q^-_{1,1,Task2,1}$	0	0	0	0	0
$q^-_{1,1,Task2,2}$	0.5	0.927	1	0.5	0
$q^-_{1,1,Task3,1}$	0	0	0	0	0
$q^-_{1,1,Task3,2}$	1	1	1	0	0

Table 2.3: Results for Low Demand Variance Case

columns show the values taken by these decision variables for different η and β.

Table 2.4 displays the computational results for the case with high variance in the demand values. The columns list the value of variables for different values of η and β. The higher values of η indicate a large penalty for excess of demand while higher β corresponds to lower tolerance towards backorders.

Tables 2.3 and 2.4, show that when η is lower than β, the model tends to select the solutions which result in the overage of demand. When both η and β are the same, we get the same number of workers for both cases. From Tables 2.3 and 2.4, all the data sets tested had a non zero penalty. When there is a low difference in demand values, the penalty is relatively lower. Even though the cost of flexible workers is higher than the specialized workers, the model chooses more flexible workers. Since production of the product requires three tasks, it is cheaper to hire one flexible worker than three specialized workers. Even though the model is effective in capturing the uncertainty of demand, it is not useful in situations when there are precedence relationships between the tasks to produce products.

Decision variables	$\eta = \beta = 5$	$\eta = 200, \beta = 5$	$\eta = 200, \beta = 20$	$\eta = \beta = 200$	$\eta = 5, \beta = 200$
$w^s_{1,Task1}$	2	0	0	2	2
$w^s_{1,Task2}$	1	0	0	1	2
$w^s_{1,Task3}$	1	0	0	1	2
w^f_1	0	1	1	0	1
$q^+_{1,1,Task1,1}$	10	0	0	10	16
$q^+_{1,1,Task1,2}$	0	0	0	0	0
$q^+_{1,1,Task2,1}$	7	1.173	4.36	7	14.83
$q^+_{1,1,Task2,2}$	0	0	0	0	0
$q^+_{1,1,Task3,1}$	7	0	0	7	16
$q^+_{1,1,Task3,2}$	0	0	0	0	0
$q^-_{1,1,Task1,1}$	0	0	0	0	0
$q^-_{1,1,Task1,2}$	6	16	16	6	0
$q^-_{1,1,Task2,1}$	0	0	0	0	0
$q^-_{1,1,Task2,2}$	9	14.83	11.6	9	1.17
$q^-_{1,1,Task3,1}$	0	0	0	0	0
$q^-_{1,1,Task3,2}$	9	16	16	9	0

Table 2.4: Results for High Demand Variance Case

Chapter 3

Task Precedence Relationship Model

The previous models do not capture settings when there are precedence relationships between tasks to produce products. In this Chapter, we propose a different model that is capable of handling precedence relationships.

We developed a new set of constraints that create the production schedule. This set of constraints counts the number of products that are produced simultaneously *at each task* at each point in time. This set of constraints also ensures that there are enough workers available at the task to complete the scheduled production. Another set of constraints ensure that enough of each product is produced to meet demand. For example, if the demand of product p at the fifth hour is 10 units and it takes 2 hours to produce the product. It is feasible to start all the production at $t = 0$; however this will result in higher worker costs than if we spread the production for this product across the first 8 periods. If all of the products began processing at $t = 0$, then we would need 10 workers simultaneously, however if we spread out the production so that a new product begins every 2 hours we could use significantly fewer workers by reducing the number of products at the same task at the same time.

The detailed model is presented in the next section. In this chapter, we present a model for deterministic cases only. Demand uncertainty is addressed in the next chapter.

3.1 Notation: Task Precedence Relationship Model

Sets:

- Product Types: **P**
- Tasks: **A**
- Set of time blocks (currently measured in hours) in the planning horizon: **T**
- Set of shifts: **Q**
- Set of shifts in a day: **I** (i.e., First, Second, Third)

Parameters:

- Demand, measured in the number of units, of product type p at time t : d_{pt}
- Time required to complete one unit of task a, measured in hours: h_a
- Total cost of one specialized worker for task a, for the entire planning horizon: c_a^s
- Total cost of one flexible worker, for the entire planning horizon: c^f
- Duration of a shift, measured in hours: u
- Cumulative completion time for task a for product type p : α_{pa}
- Number of workers required to complete task a : n_a
- The shift that corresponds to time t: $q(t)$
- The shift type that corresponds to shift q: $i(q)$

Decision Variables:

- Number of flexible workers, for task a, at time t hours: w_{at}^f

- Number of units of product type p that start production at time t: x_{pt}
- Number of specialized workers required for task a, in shift type i: w_{ia}^{s}
- Number of flexible workers required in shift type i: w_{i}^{f}

Objective Function:

$$\min \sum_{i \in \mathbf{I}} \sum_{a \in \mathbf{A}} c_a^s w_{ia}^s + \sum_{i \in \mathbf{I}} c^f w_i^f \tag{3.1}$$

Constraints:

$$\sum_{p \in \mathbf{P}} \sum_{k \in \mathbf{T}: k+\alpha_{pa}-h_a \leq t < \alpha_{pa}+k} n_a x_{pk} \leq w_{i(t)a}^s + w_{at}^f \qquad \forall a \in \mathbf{A}, t \in \mathbf{T} \tag{3.2}$$

$$\sum_{k \in \mathbf{T}: k+\sum_a h_a \leq t} x_{pk} \geq \sum_{j \leq t} d_{pj} \qquad \forall t \in \mathbf{T}, p \in \mathbf{P} \tag{3.3}$$

$$w_{i(t)}^f \geq \sum_{a \in \mathbf{A}} w_{at}^f \qquad \forall t \in \mathbf{T} \tag{3.4}$$

$$x_{pt}, w_{at}^f \in \mathbb{Z}^+ \tag{3.5}$$

$$w_{i(t)}^f, w_{ia}^s \geq 0 \tag{3.6}$$

Objective (3.1) is to minimize the workforce costs. Constraint (3.2) calculates the total number of workers needed in each shift. This is achieved by summing the required number of units of task a, for all the products p. The precedence relationship is enforced by α_{pa} matrix. Figure 3.1 illustrates the constraint for a 2-product situation having completion time as $h = [1.5, 2, 2, 4]$, h_a is the completion time of ath task, in hours. The products have the following sequence

- Product Type 1 order: Task1 → Task2 → Task3
- Product Type 2 order: Task2 → Task3 → Task4

Time in hours is shown horizontally. At any point in time, the number of workers required for a task a, should be the sum of x_{pk}. The example in the figure shows at t= 1.75 hours, two products are being processed at Task 2, one of each type. Assuming the number of workers required to complete Task2 (n_{Task2}) in one, this schedule would require the number of total workers for that task to be greater than or equal to two, for this shift. Constraint (3.3) ensures that demand is met on time, by enforcing that the total number of demand units is less than or equal to the production (x_{pk}). Constraint (3.4) determines the number of specialized workers required for each shift type, by taking the maximum number of workers required during each shif in that type. Constraint (3.5) complete a similar calculation to determine the number of flexible workers.

3.2 Results and Conclusions

We implemented the model in Python and solved it using CPLEX solver. The results were compared for different values of cost function for the same demand set.

Iteration	$c^s_{Task_1}$	c^f	$w^s_{1,Task_1}$	w^f_1	Optimal Solution	Time (seconds)
1	10	12	6	0	60	0.15
2	15	13	0	6	78	0.28
3	8	8	6	0	48	0.289

Table 3.1: Results for Deterministic Model with Precedence

A small case was considered to check the model implementation. The data set includes one product one task and a 4-hour long shift [1]. Table 3.1 display the computational results for this model. Each row of Table 3.1 depicts different costs for specialized

[1] See "(Appendix C)"

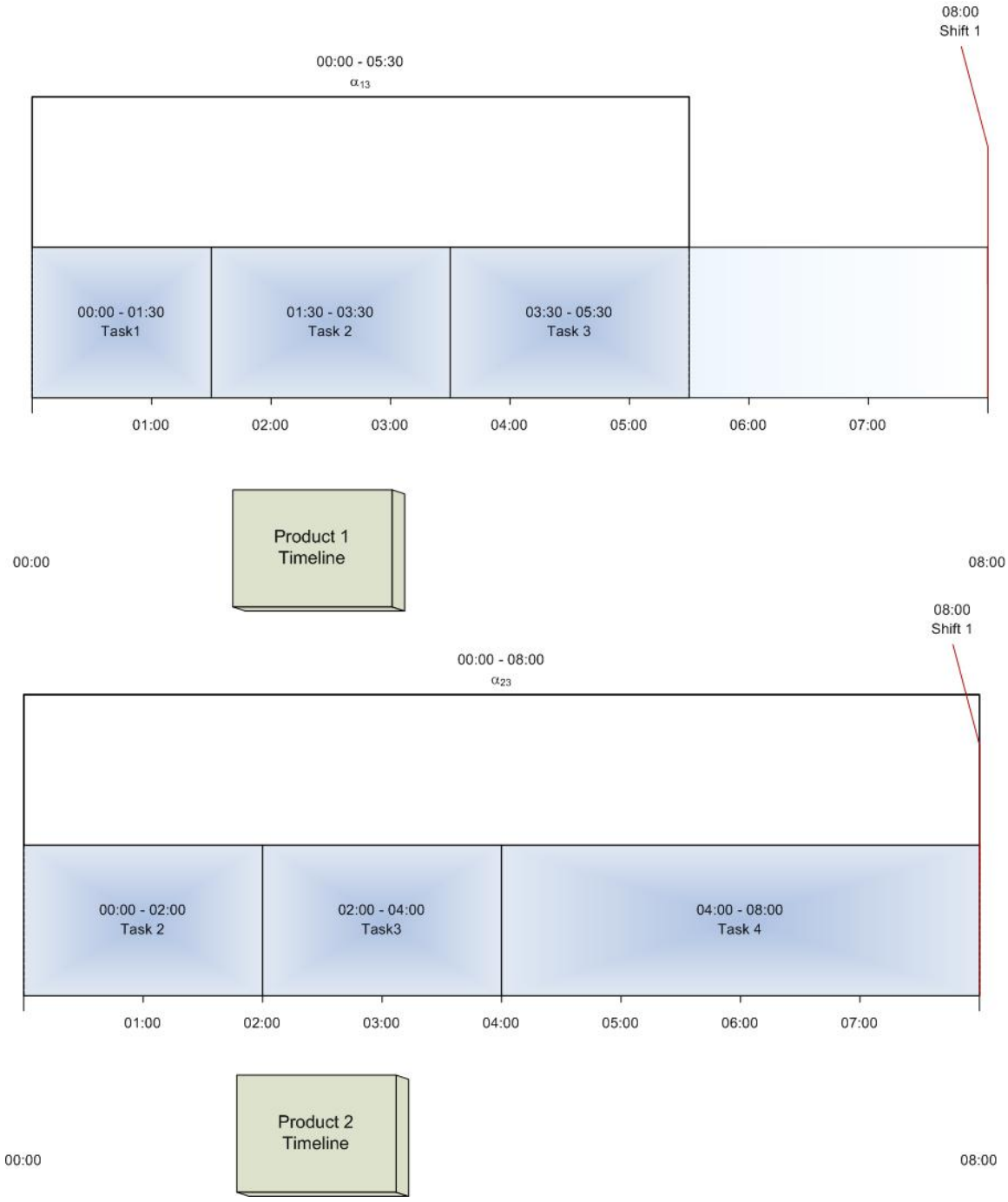

Figure 3.1: Example Timeline for Product 1 and Product 2

and flexible workers. The last column denotes the time (in seconds) taken to solve the corresponding instance.

When the cost of a specialized worker is higher than the cost of a flexible worker, the solution should have more flexible workers or vice-versa. From Table 3.1, it can be easily seen that computation results are in accordance with this intuition. When both costs are the same, the model arbitrarily chooses specialized workers over flexible workers. Another possible solution, for the same cost values, could be a mix of both types of workers. From equations (3.2) to (3.6), that solution can be easily verified to having higher costs.

Chapter 4

Robust Math Programming with Precedence

In Chapter 3, we introduced a deterministic model to schedule the workforce in a manufacturing setting having a series of tasks required to finish the production. The model is able to capture the precedence relationship in the tasks. In this chapter, we extend the same model to handle uncertainty in demand. We propose a Robust Mathematical Programming based formulation and discuss different cases related to the model. Each case represents different management objectives, which are discussed in the corresponding sections.

Uncertainty lies in every aspect of a manufacturing facility. There can be uncertainty in the delivery time, production time, arrival of orders, production yield or demand quantity. In this thesis, we considered uncertainty only in demand quantity. Specifically we assume demand of a particular product can take any value from a set of possible instances (uncertainty set) at a given point in time. The elements of the uncertainty set are referred as demand situations.

As discussed in Chapter 1, Robust Mathematical Programming finds a fixed set of

variables as a solution for different instances (uncertainty set) of an uncertain parameter. If this fixed set of variables can not satisfy all the instances, a suitable penalty is attached for violation. Implementing the similar strategy, we allowed violation of some or all of the constraints of deterministic model in Chapter 3 and are minimizing not only the worker costs, but also penalty for the violation. In this chapter, we implemented the minimization of mean variance approach of Robust Mathematical Programming. For every demand situation, the model calculates the objective function value, which is the sum of total worker costs and penalty for constraint violations. A probability is assigned to each situation. The objective function is to minimize the expected value and the variance of constraint violation. This converts the Mixed Integer Program (MIP) discussed in Chapter 3, into a Mixed Integer Non-Linear Program (MINLP). In order to develop the models, we define the following notation.

4.1 Notation: Robust Math Programming with Precedence

Sets:

- Product Types: **P**
- Tasks: **A**
- Set of time blocks (currently measured in hours) in the planning horizon: **T**
- Set of shifts: **Q**
- Set of shifts in a day: **I** (i.e., First, Second, Third)
- Set for number of situations: **S**
- Set for type of constraints: **G**

Parameters:

- Demand, measured in the number of units, of product type p, at time t, in situation s : d_{pst}
- Time to complete task a, measured in hours: h_a
- Total cost of one specialized worker for task a, throughout the planning horizon: c_a^s
- Total cost of one flexible worker, throughout the planning horizon: c^f
- Duration of a shift, measured in hours: u
- Cumulative completion time for task a for product type p : α_{pa}
- Number of workers required to complete task a : n_a
- The shift that corresponds to time t: $q(t)$
- The shift type that corresponds to shift q: $i(q)$
- Probability of having a situation s : p_s
- Tradeoff vector for model robustness to solution robustness: ω
- Penalty for violation of the constraint g : ω_g

Decision Variables:

- Number of flexible workers, for task a, at time t hours: w_{at}^f
- Number of units of product type p that start production at time t, for situation s: x_{pst}
- Number of specialized workers required for task a, in shift type i: w_{ia}^s

- Number of flexible workers required in shift type i: w_i^f
- Slack in the constraint type g, for situation s, task a at time t : z_{gats}

4.2 Case 1: Violation in Number of Workers Constraint

In this section, we present a situation when the management is only interested in the effects of not hiring the exact number of workers as required by the production schedule. Since the variables x_{pst} denote the number of units of product p to be produced at time t in a situation s, the entire set of variables x_{pst} can be referred as a production schedule. Constraint (3.2) calculates the number of required *specialized* and *flexible* workers from a production schedule. By allowing violation in constraint (3.2), we do not have to hire the required number of workers as required by the production schedule. In situations where backorder and inventory storage costs are lower than the worker costs, the penalty for violating this constraint is low. In the situations where hiring different number of workers than required by the production schedule is highly undesirable, the penalty should be high.

In representing the above idea, we define the variables z_{1ats}. The variables z_{1ats} are free and thus capture both the positive and negative violation in the constraints.

Objective Function:

$$\min \quad \sigma(w_{ia}^s, w_i^f, w_{at}^f, x_{psk}) + \omega\rho(z_{1at1}, z_{1at2}, ...z_{1ats}) \tag{4.1}$$

where

$$\sigma(w_{ia}^s, w_i^f, w_{at}^f, x_{psk}) = \sum_{i\in\mathbf{I}} \sum_{a\in\mathbf{A}} c_a^s w_{ia}^s + \sum_{i\in\mathbf{I}} c^f w_i^f$$

$$\rho(z_1, z_2, ...z_s) = \sum_{a \in \mathbf{A}} \sum_{t \in \mathbf{T}} \sum_{s \in \mathbf{S}} p_s z_{1ats}^2$$

Constraints:

$$\sum_{p \in \mathbf{P}} \sum_{k \in \mathbf{T}: k + \alpha_{pa} - h_a \leq t < \alpha_{pa} + k} n_a x_{psk} + z_{1ats} = w_{i(t)a}^s + w_{at}^f \qquad \forall a \in \mathbf{A}, t \in \mathbf{T}, s \in \mathbf{S} \quad (4.2)$$

Objective function (4.1) is composed of two parts. The first part $\sigma(w_{ia}^s, w_i^f, w_{at}^f, x_{psk})$ represents the total worker costs i.e., the sum of all specialized workers and flexible workers costs. The second part, $\rho(z_1, z_2, ...z_s)$, is the penalty for constraint violation. The total violation is the sum of squares of z_{1ats} variables. In this way, both the negative and positive violations are penalized by the same penalty. These squared terms are multiplied by the respective probability of having situation s. The term ω_1 is the unit cost of penalty for having a constraint violation. Thus the objective function as a whole minimizes the total worker costs and total penalty for constraint violations.

Constraint (4.2) is a relaxed version of constraint (3.2). This constraint is relaxed in a sense because free variables are added to accommodate the difference between the production schedule and the number of workers required to finish that schedule. Through this constraint we are allowing for having more or less number of workers than required by the schedule. The remaining constraints are the same as in the MIP model.

4.2.1 Results and Conclusions

We implemented the model in Python and solved it using CPLEX solver. The following results considered two different values of ω_1 for a single product, requiring one task to finish. Two different demand situations are considered. The first situation has high variability in the demand values across different time periods. The second situation has very low variability across different time periods. At time t, the difference in the demand values is much higher for situation 1 than situation 2. We have tested the

model for different values of probability function for a particular situation. The cost of a specialized worker is $c^s_{task1} = \$10$ and of flexible worker, $c^f = \$12$.

Table 4.1 displays the computational results. The first column lists the different weights considered in the analysis. The second column lists the probability functions considered. The number of workers corresponding to each solution is shown in the third and fourth column. The objective function value and corresponding penalty is listed in the fifth and sixth column.

Referring to Table 4.1, at low values of ω_1, there is violation in constraints. When the probability of having a higher demand value is high, there is zero penalty. While at equal probabilities, the model chooses a solution corresponding to the smallest penalty and the number of workers is closer to the number of workers required for high demand value situations. In the situation of having high penalty for constraint violation, ω_1= 70, the number of workers hired remains the same for all the probability functions. Since the cost of hiring flexible workers is higher than hiring specialized workers, no flexible worker is hired in any situation.

	Probability function	$w^s_{1,task_1}$	w^f_1	Optimal Solution	Penalty
ω_1 =10	$p_1 = 0.9, p_2 = 0.1$	20	0	200	0
	$p_1 = 0.1, p_2 = 0.9$	17	0	188	18
	$p_1 = p_2 = 0.5$	19	0	200	10
ω_1 =70	$p_1 = 0.9, p_2 = 0.1$	20	0	200	0
	$p_1 = 0.1, p_2 = 0.9$	20	0	200	0
	$p_1 = p_2 = 0.5$	20	0	200	0

Table 4.1: Robust Mathematical Programming Model : Case 1

4.3 Case 2: Violation in Production Constraints

In this section, we discuss another scenario when the effect of having a different production schedule than required by the quantity demanded is the main focus of study. Given

the demand of a product p, constraint (3.3), calculates the number of units of product p to be produced at time t. Ideally, number of units to be produced should be at least equal to the quantity demanded. In this Robust Mathematical Programming model, the production quantity is allowed to be different than the demand. Lower penalty for constraint violation represents the situation when backorder and inventory costs are low while higher penalty denotes the opposite.

In this case, we introduced uncertainty in the demand constraints, numbered as constraint (3.3) in the previous chapter. We attach the free variables, z_{2pts} in constraint (3.3), to measure the violation from demand. All remaining constraints remain the same.

Objective Function:

$$\min \quad \sigma(w^s_{ia}, w^f_i, w^f_{at}, x_{psk}) + \omega\rho(z_{2pt1}, z_{2pt2}, ...z_{2pts}) \tag{4.3}$$

where

$$\sigma(w^s_{ia}, w^f_i, w^f_{at}, x_{psk}) = \sum_{i\in\mathbf{I}}\sum_{a\in\mathbf{A}} c^s_a w^s_{ia} + \sum_{i\in\mathbf{I}} c^f w^f_i$$

$$\rho(z_1, z_2, ...z_s) = \sum_{p\in\mathbf{P}}\sum_{t\in\mathbf{T}}\sum_{s\in\mathbf{S}} p_s z^2_{2pts}$$

Constraints:

$$\sum_{k\in\mathbf{T}:k+\sum_a h_a\leq t} x_{psk} - z_{2pts} = \sum_{j\leq t} d_{psj} \qquad \forall t \in \mathbf{T}, p \in \mathbf{P}, s \in \mathbf{S} \tag{4.4}$$

Again, the objective function minimizes the total worker costs and the penalty for constraint violations. The function $\rho(z_1, z_2, ...z_s)$ measures the total penalty violation of constraint (4.4). All the other constraints remaining the same, we replaced constraint (3.3) with constraint (4.4). Constraint (4.4) allows for the total production to deviate from demand amount. We have attached penalty for over and under production.

4.3.1 Results and Conclusions

We implemented the model in Python and solved it using CPLEX solver. Table 4.2 shows the computational results for this case considering the same demand data as in Case1. For lower values of ω_2, constraint violation or non zero penalty exists at a different probability function. The highest violation is for the case when lower demand values are more probable. Even at higher ω_2 values, constraint violations do exist. Thus, the variable set z_{2pts} should have higher penalty.

	Probability function	$w^s_{1,task_1}$	w^f_1	Optimal Solution	Penalty
ω_2 =10	$p_1 = 0.9, p_2 = 0.1$	26	0	278	18
	$p_1 = 0.1, p_2 = 0.9$	18	0	230	50
	$p_1 = p_2 = 0.5$	26	0	270	10
ω_2 =70	$p_1 = 0.9, p_2 = 0.1$	28	0	280	0
	$p_1 = 0.1, p_2 = 0.9$	26	0	274	14
	$p_1 = p_2 = 0.5$	28	0	280	0

Table 4.2: Robust Mathematical Programming Model : Case 2

4.4 Case 3: Violation in Number of Flexible Workers Constraint

In the previous sections we looked into the cases having violations in production and demand constraints. This section presents a scenario regarding the hiring related decisions of flexible workers. Constraint (3.4) calculates the overall flexible workers required for all the tasks. Violation of this constraint signifies the situation when hiring of the flexible workers is not equal to the requirement.

In this case, the uncertainty in constraint (3.4) is considered. We introduced the free variables, z_{3t}, representing deviation from the required number of flexible workers.

Objective Function:

$$\min \quad \sigma(w^s_{ia}, w^f_i, w^f_{at}, x_{psk}) + \omega\rho(z_{3t}) \tag{4.5}$$

where

$$\sigma(w^s_{ia}, w^f_i, w^f_{at}, x_{psk}) = \sum_{i\in\mathbf{I}} \sum_{a\in\mathbf{A}} c^s_a w^s_{ia} + \sum_{i\in\mathbf{I}} c^f w^f_i$$

$$\rho(z_1, z_2, ...z_s) = \sum_{t\in\mathbf{T}} p_s z^2_{3t}$$

Constraints:

$$w^f_{i(t)} - z_{3t} = \sum_{a\in\mathbf{A}} w^f_{at} \qquad \forall t \in \mathbf{T} \tag{4.6}$$

The terms in the objective function are the same as in Case 1. The constraints of the model are the same as in Chapter 3, except constraint (3.4). Instead, we propose constraint (4.6), which allows the number of flexible workers for a shift i to be different than the sum of flexible workers for all the tasks.

4.4.1 Results and Conclusions

We implemented the model in Python and solved it using CPLEX solver for the same demand set. The only difference is that the cost of flexible workers is lowered to \$8. Since the variables z_{3t} are not situation specific, there are not any probability attached to them.

Table 4.3 shows the computational results for different values of ω_3. From Table 4.3, it is evident that constraint violation (non-zero penalty) exists at much lower values of ω_3. Even when the ω_3 is lower than the cost of a flexible worker, ω_3= 5, the penalty is zero. From the above observation, it seems that hiring a different number of flexible workers than the required is advantageous only when penalty is very low. For high penalty values, it is always recommended to hire flexible workers according to the requirement.

	$w^s_{1,task_1}$	w^f_1	Optimal Solution	Penalty
$\omega_3 = 3$	0	19	158	6
$\omega_3 = 5$	0	20	160	0
$\omega_3 = 70$	0	20	160	0

Table 4.3: Robust Mathematical Programming Model : Case 3

4.5 Case 4: Violation in Both Production and Number of Workers Constraints

In this case, we considered situations when demand is not met and/or the number of workers is not enough to meet the production schedule. Given the quantity demanded, constraint (3.3) calculates the number of units of product p to be produced at time t. Constraint (3.2) calculates the number of specialized and flexible workers required for task a at time t. Since demand is situation based, the effects of the violation of constraints (3.2) and (3.3) are studied. We attach the free variables z_{1ats} and z_{2pts} respectively to constraints (3.2) and (3.3). The squared value of these variables is minimized in the objective function.

Objective Function:

$$\min\sigma(w^s_{ia}, w^f_i, w^f_{at}, x_{psk}) + \omega\rho(z_{gat1}, z_{gat2}, ...z_{gats}) \tag{4.7}$$

where

$$\sigma(w^s_{ia}, w^f_i, w^f_{at}, x_{psk}) = \sum_{i\in\mathbf{I}}\sum_{a\in\mathbf{A}} c^s_a w^s_{ia} + \sum_{i\in\mathbf{I}} c^f w^f_i$$

$$\rho(z_1, z_2, ...z_s) = \sum_{g\in\mathbf{G}}\sum_{a\in\mathbf{A}}\sum_{t\in\mathbf{T}}\sum_{p\in\mathbf{P}}\sum_{s\in\mathbf{S}} p_s z^2_{gapts}$$

Constraints:

$$\sum_{p\in\mathbf{P}}\sum_{k\in\mathbf{T}:k+\alpha_{pa}-h_a\leq t<\alpha_{pa}+k} n_a x_{psk} + z_{1ats} = w^s_{i(t)a} + w^f_{at} \qquad \forall a\in\mathbf{A}, t\in\mathbf{T}, s\in\mathbf{S} \quad (4.8)$$

$$\sum_{k\in\mathbf{T}:k+\sum_a h_a\leq t} x_{psk} - z_{2pts} = \sum_{j\leq t} d_{psj} \qquad \forall t\in\mathbf{T}, p\in\mathbf{P}, s\in\mathbf{S} \quad (4.9)$$

The objective function is the same as explained in Case 1. The difference is in the penalty function, where we considered the variables z_{1ats} and z_{2pts}. The rest of the model is the same as the model in Chapter 3, except constraints (3.2) and (3.3). Instead, we are considering (4.2) and (4.4) which are reproduced in (4.8) and (4.9).

4.5.1 Results and Conclusions

We implemented the model in Python and solved it using CPLEX solver. The results are presented for different values and combinations of penalty functions for both of the variables. Table 4.4 displays the computational results for this case. The first column lists the different weights considered in the analysis. The second column lists the different probability functions for each combination of weights. Referring to Table 4.4, the lowest worker costs are when both the penalty values are lower and lower demand values are more probable. For every combination of penalty values, the lowest worker cost corresponds to the situation when lower demand values are more probable. Also the penalty values are lower for these values of probability function. When the probability of having both of the situations is equal, the penalty is highest, even though the worker costs are not highest for this case. Depending upon the values of ωs, the corresponding free variables are non-zero.

	Probability function	$w^s_{1,task_1}$	w^f_1	Optimal Solution	Penalty	Worker Costs
ω_1 =10, ω_2 = 200	$p_1 = 0.9, p_2 = 0.1$	11	15	744	454	290
	$p_1 = 0.1, p_2 = 0.9$	4	10	605	445	160
	$p_1 = p_2 = 0.5$	7	13	1476	1250	226
ω_1 =200, ω_2 = 10	$p_1 = 0.9, p_2 = 0.1$	12	13	1457	1181	276
	$p_1 = 0.1, p_2 = 0.9$	4	10	1327	1167	160
	$p_1 = p_2 = 0.5$	8	11	3502	3290	212
ω_1 =ω_2=100	$p_1 = 0.9, p_2 = 0.1$	9	17	3514	3220	294
	$p_1 = 0.1, p_2 = 0.9$	4	11	3362	3190	172
	$p_1 = p_2 = 0.5$	7	13	9026	8800	226
ω_1 =ω_2=5	$p_1 = 0.9, p_2 = 0.1$	12	11	433	181	252
	$p_1 = 0.1, p_2 = 0.9$	5	7	303	169	134
	$p_1 = p_2 = 0.5$	9	8	643.5	457.5	186

Table 4.4: Robust Mathematical Programming Model : Case 4

4.6 Case 5: Violation in Both Production and Number of Flexible Workers Constraint

In the last section, we studied the violation in production constraint and the constraint calculating total requirement of the workers at time t for task a. In this section, we study the effects of violation in production constraint and the number of flexible workers constraint. Constraint (3.3) calculates the production schedule from the quantity demanded. Violation in this constraint refers to the situation when product production is not exactly equal to the requirement. Constraint (3.4) calculates the total number of required flexible workers in a shift. The required number of flexible workers should be atleast equal to the sum of their requirement over all tasks. In order to study the combined effect of the violation in constraints (3.3) and (3.4), the free variables z_{2pts} and z_{3t} are attached to the respective constraints.

Objective Function:

$$\min \quad \sigma(w^s_{ia}, w^f_i, w^f_{at}, x_{psk}) + \omega\rho(z_{gat1}, z_{gat2}, ...z_{gats}) \tag{4.10}$$

where

$$\sigma(w_{ia}^s, w_i^f, w_{at}^f, x_{psk}) = \sum_{i\in\mathbf{I}}\sum_{a\in\mathbf{A}} c_a^s w_{ia}^s + \sum_{i\in\mathbf{I}} c^f w_i^f$$

$$\rho(z_1, z_2, ...z_s) = \sum_{g\in\mathbf{G}}\sum_{a\in\mathbf{A}}\sum_{t\in\mathbf{T}}\sum_{p\in\mathbf{P}}\sum_{s\in\mathbf{S}} p_s z_{gapts}^2$$

Constraints:

$$\sum_{k\in\mathbf{T}:k+\sum_a h_a\leq t} x_{psk} - z_{2pts} = \sum_{j\leq t} d_{psj} \qquad \forall t\in\mathbf{T}, p\in\mathbf{P}, s\in\mathbf{S} \tag{4.11}$$

$$w_{i(t)}^f - z_{3t} = \sum_{a\in\mathbf{A}} w_{at}^f \qquad \forall t\in\mathbf{T} \tag{4.12}$$

The objective function is the same as detailed in Case 1. The only difference is in the penalty function. In this case, we considered variables z_{2pts} and z_{3t}. The rest of the model is the same as in Chapter 3, except constraints (3.3) and (3.4). Instead, we are considering (4.4) and (4.6) which are reproduced in (4.11) and (4.12).

4.6.1 Results and Conclusions

We implemented the model in Python and solved it using CPLEX solver. The results are presented for different values of the penalty function. Table 4.5 displays the computational results for this case. In Table 4.5, the lowest worker costs are for the case when higher probability is for lower demand values. When ω_2 is higher than ω_3, the results are independent of the probability distribution function.

	Probability function	$w^s_{1,task_1}$	w^f_1	Optimal Solution	Penalty	Worker Costs
ω_2 =10, ω_3 = 200	$p_1 = 0.9, p_2 = 0.1$	26	0	278	18	260
	$p_1 = 0.1, p_2 = 0.9$	18	0	230	50	180
	$p_1 = p_2 = 0.5$	26	0	270	10	260
ω_2 =200, ω_3 = 10	$p_1 = 0.9, p_2 = 0.1$	27	0	280	10	270
	$p_1 = 0.1, p_2 = 0.9$	27	0	280	10	270
	$p_1 = p_2 = 0.5$	27	0	280	10	270
ω_2 =ω_3=100	$p_1 = 0.9, p_2 = 0.1$	28	0	280	0	280
	$p_1 = 0.1, p_2 = 0.9$	27	0	280	10	270
	$p_1 = p_2 = 0.5$	28	0	280	0	280
ω_2 =ω_3=5	$p_1 = 0.9, p_2 = 0.1$	25	0	264	14	250
	$p_1 = 0.1, p_2 = 0.9$	16	0	197.5	37.5	160
	$p_1 = p_2 = 0.5$	23	0	255	25	230

Table 4.5: Robust Mathematical Programming Model : Case 5

4.7 Case 6: Violation in Both Number of Workers and Number of Flexible Workers Constraints

In this section, we study a tradeoff between the requirement of the *flexible* workers versus the total number of required workers (*specialized* and *flexible*) at a single point in time. The total number of required workers are determined by the constraint (3.2). The constraint translates the production schedule into the number of workers required to finish the schedule. The number of flexible workers required is determined by the constraint (3.4). By considering different weights for violation in each of the constraint (3.2) and (3.4) provides more flexibility in management goals. For instance, the mangement goals might be strict towards zero inventory but comparitively relaxed towards hiring the required flexible workers. This instance would lead to having high penalty for the constraint (3.2) compared to the constraint (3.4). In order to study the constraint violation, we introduce the free variables z_{1ats} and z_{3t} for constraint (3.2) and (3.4) respectively.

Objective Function:

$$\min \quad \sigma(w^s_{ia}, w^f_i, w^f_{at}, x_{psk}) + \omega\rho(z_{gat1}, z_{gat2}, ...z_{gats}) \tag{4.13}$$

where

$$\sigma(w^s_{ia}, w^f_i, w^f_{at}, x_{psk}) = \sum_{i\in\mathbf{I}}\sum_{a\in\mathbf{A}} c^s_a w^s_{ia} + \sum_{i\in\mathbf{I}} c^f w^f_i$$

$$\rho(z_1, z_2, ...z_s) = \sum_{g\in\mathbf{G}}\sum_{a\in\mathbf{A}}\sum_{t\in\mathbf{T}}\sum_{p\in\mathbf{P}}\sum_{s\in\mathbf{S}} p_s z^2_{gapts}$$

Constraints:

$$\sum_{p\in\mathbf{P}}\sum_{k\in\mathbf{T}:k+\alpha_{pa}-h_a\leq t<\alpha_{pa}+k} n_a x_{psk} + z_{1ats} = w^s_{i(t)a} + w^f_{at} \qquad \forall a\in\mathbf{A}, t\in\mathbf{T}, s\in\mathbf{S} \quad (4.14)$$

$$w^f_{i(t)} - z_{3t} = \sum_{a\in\mathbf{A}} w^f_{at} \qquad \forall t\in\mathbf{T} \quad (4.15)$$

The objective function is as explained in Case 1. The difference is in the penalty function, by considering variables z_{1ats} and z_{3t}. Instead of constraints (3.2) and (3.4), we introduce constraints (4.14) and (4.15) respectively. Instead, we are considering (4.2) and (4.6) which are reproduced in (4.14) and (4.15).

4.7.1 Results and Conclusions

We implemented the model in Python and solved it using CPLEX solver. The computational results for various values of probability and penalty are presented in Table 4.6. As Table 4.6 illustrates, whenever higher weight is given to the z_{1ats} variables, the penalty value is zero. For very low values of ω_3, the z_{3t} variables are non-zero.

4.8 Case 7: Violation in All Three Constraint Sets

In this section, we present a case having more theoretical interest. This case considers the violation in all the three constraints namely the number of workers constraint (3.2), the

	Probability function	$w^s_{1,task_1}$	w^f_1	Optimal Solution	Penalty	Worker Costs
ω_2 =10, ω_3 = 200	$p_1 = 0.9, p_2 = 0.1$	20	0	200	0	200
	$p_1 = 0.1, p_2 = 0.9$	17	0	188	18	170
	$p_1 = p_2 = 0.5$	19	0	200	10	190
ω_2 =200, ω_3 = 10	$p_1 = 0.9, p_2 = 0.1$	20	0	200	0	200
	$p_1 = 0.1, p_2 = 0.9$	20	0	200	0	200
	$p_1 = p_2 = 0.5$	20	0	200	0	200
ω_2 =ω_3=100	$p_1 = 0.9, p_2 = 0.1$	20	0	200	0	200
	$p_1 = 0.1, p_2 = 0.9$	20	0	200	0	200
	$p_1 = p_2 = 0.5$	20	0	200	0	200
ω_2 =ω_3=5	$p_1 = 0.9, p_2 = 0.1$	18	0	199	19	180
	$p_1 = 0.1, p_2 = 0.9$	14	0	175	35	140
	$p_1 = p_2 = 0.5$	18	0	195	15	180

Table 4.6: Robust Mathematical Programming Model : Case 6

production constraints (3.3) and the number of flexible workers constraint (3.4). Different weights are attached to study the effects of violation in each constraint with respect to violation in other constraint. The uncertainty is studied by attaching the free variables z_{1ats}, z_{2pts} and z_{3t} respectively to the constraints (3.2) (3.3) and (3.4) respectively.

Objective Function:

$$\min \quad \sigma(w^s_{ia}, w^f_i, w^f_{at}, x_{psk}) + \omega\rho(z_{gat1}, z_{gat2}, ...z_{gats}) \tag{4.16}$$

where

$$\sigma(w^s_{ia}, w^f_i, w^f_{at}, x_{psk}) = \sum_{i\in\mathbf{I}}\sum_{a\in\mathbf{A}} c^s_a w^s_{ia} + \sum_{i\in\mathbf{I}} c^f w^f_i$$

$$\rho(z_1, z_2, ...z_s) = \sum_{g\in\mathbf{G}}\sum_{a\in\mathbf{A}}\sum_{t\in\mathbf{T}}\sum_{p\in\mathbf{P}}\sum_{s\in\mathbf{S}} p_s z^2_{gapts}$$

Constraints:

$$\sum_{p\in\mathbf{P}}\sum_{k\in\mathbf{T}:k+\alpha_{pa}-h_a\le t<\alpha_{pa}+k} n_a x_{psk} + z_{1ats} = w^s_{i(t)a} + w^f_{at} \qquad \forall a\in\mathbf{A}, t\in\mathbf{T}, s\in\mathbf{S} \tag{4.17}$$

$$\sum_{k\in\mathbf{T}:k+\sum_a h_a\le t} x_{psk} - z_{2pts} = \sum_{j\le t} d_{psj} \qquad \forall t\in\mathbf{T}, p\in\mathbf{P}, s\in\mathbf{S} \tag{4.18}$$

$$w^f_{i(t)} - z_{3t} = \sum_{a \in \mathbf{A}} w^f_{at} \qquad \forall t \in \mathbf{T} \tag{4.19}$$

$$x_{pst}, w^f_{at}, w^f_{i(t)}, w^s_{ia} \in \mathbb{Z}^+ \tag{4.20}$$

$$w^f_{i(t)}, w^s_{ia} \geq 0 \tag{4.21}$$

In this case, we introduced uncertainty in all three constraints. We allowed deviations from the demand and also allowed the possibility of having the workers not equal to the required number.

4.8.1 Results

We implemented the model in Python and solved it using CPLEX solver. The computation results are presented in Table 4.7 for different values of the penalty weights and probabilities. It can be easily verified from Table 4.7 that lowest worker cost exist when higher probability corresponds to low difference in demand values. The highest penalty occurs when both situations are equally probable, for each combination of the weights.

	Probability function	$w^s_{1,task_1}$	w^f_1	Optimal Solution	Penalty	Worker Costs
$\omega_1 = \omega_2$=200, $\omega_3 = 100$	$p_1 = 0.9, p_2 = 0.1$	13	2	13114	12960	154
	$p_1 = 0.1, p_2 = 0.9$	6	1	9032	8960	72
	$p_1 = p_2 = 0.5$	10	1	21712	21600	112
$\omega_1 = 5, \omega_2 = \omega_3$ =200	$p_1 = 0.9, p_2 = 0.1$	15	0	1230	1080	150
	$p_1 = 0.1, p_2 = 0.9$	7	0	693.5	623.5	70
	$p_1 = p_2 = 0.5$	11	0	1330	1220	110
$\omega_2 = 5, \omega_1 = \omega_3$ =200	$p_1 = 0.9, p_2 = 0.1$	14	0	1396	1256	140
	$p_1 = 0.1, p_2 = 0.9$	7	0	894	824	70
	$p_1 = p_2 = 0.5$	11	0	2322.5	2212.5	110
$\omega_3 = 5, \omega_1 = \omega_3$ =200	$p_1 = 0.9, p_2 = 0.1$	11	4	7448	7290	158
	$p_1 = 0.1, p_2 = 0.9$	4	4	6883	6795	88
	$p_1 = p_2 = 0.5$	8	3	18231	18115	116
$\omega_1 = \omega_2 = \omega_3$ =5	$p_1 = 0.9, p_2 = 0.1$	13	1	510.5	368.5	142
	$p_1 = 0.1, p_2 = 0.9$	6	1	311.5	239.5	72
	$p_1 = p_2 = 0.5$	10	0	677.5	577.5	100

Table 4.7: Robust Mathematical Programming Model : Case 7

4.8.2 Comparison of the Cases

Tables 4.8, 4.9, and 4.10 provide the comparison of all the above cases for the same demand data, same number and sequence of the products. We are considering a one product case requiring three tasks to complete. For this comparison, we are assuming that $\omega_1 = \omega_2 = \omega_3 = \omega$. Table 4.8 displays the computational results for ω= 10. The columns show variable values for the corresponding cases. Tables 4.9, and 4.10 display the computational results for ω =150 and 200 respectively.

Tables 4.8, 4.9, 4.10, reveal that Case 7 has the lowest number of total workers for different values of ωs. For lower values of ω_1, ω_2, the penalties do come into account. At very low values of ω_3, z_{3t} variables are non-zero. For higher values of ωs, the total number of workers is the same for all the cases except Case 7.

	$w^s_{1,Task_1}$	$w^s_{1,Task_2}$	$w^s_{1,Task_3}$	w^f_1
Case1	8	0	0	16
Case2	7	7	0	11
Case3	8	7	0	10
Case4	0	0	0	15
Case5	7	6	0	11
Case6	0	0	0	23
Case7	1	0	0	5

Table 4.8: Comparison of Cases for ω =10

	$w^s_{1,Task_1}$	$w^s_{1,Task_2}$	$w^s_{1,Task_3}$	w^f_1
Case1	8	0	0	18
Case2	8	8	0	10
Case3	8	8	0	10
Case4	0	0	0	18
Case5	8	8	0	10
Case6	0	0	0	26
Case7	1	0	0	6

Table 4.9: Comparison of Cases for ω =150

Another important observation is that number of each type of workers is same for Case 1, 2, and 3. As explained before, infeasibility in Case 3 occurs only at low values

	$w^s_{1,Task_1}$	$w^s_{1,Task_2}$	$w^s_{1,Task_3}$	w^f_1
Case1	8	0	0	18
Case2	8	8	0	10
Case3	8	8	0	10
Case4	0	0	0	18
Case5	8	8	0	10
Case6	0	0	0	26
Case7	1	0	0	6

Table 4.10: Comparison of Cases for ω =200

of ωs, for a single task situation.

In a practical setting, we would like to have flexibility in terms of number of workers to be hired and also to schedule the production. In other words, it is more realistic to consider violation in the constraints corresponding to flexible workers and production. Thus, we compare Case 2 and Case 5 for different values of ωs. In Case 2 only demand constraints are allowed to be violated, whereas Case 5 represents when the constraint calculating the number of flexible workers is also allowed to be violated. Referring to Tables 4.11, 4.12 and 4.13, a difference in both cases occurs for low values of ωs. For the 3-task case, at moderate values of ωs, the total number of workers is the same and penalty is zero for both cases.

	Case 2	Case5
$w^s_{1,Task_1}$	7	7
$w^s_{1,Task_2}$	6	6
$w^s_{1,Task_3}$	1	0
w^f_1	11	11
z_{33}	0	-1
z_{2182}	-1	0

Table 4.11: Comparison of Cases for ω = 10

	Case 2	Case5
$w^s_{1,Task_1}$	8	8
$w^s_{1,Task_2}$	5	6
$w^s_{1,Task_3}$	3	2
w^f_1	10	10
z_{33}	0	0
z_{2182}	0	0

Table 4.12: Comparison of Cases for $\omega = 70$

	Case 2	Case5
$w^s_{1,Task_1}$	8	8
$w^s_{1,Task_2}$	6	6
$w^s_{1,Task_3}$	2	2
w^f_1	10	10
z_{33}	0	0
z_{2182}	0	0

Table 4.13: Comparison of Cases for $\omega = 200$

Chapter 5

Techniques to Solve Large Problem Instances

In this section, we tested the models of Chapter 4 on a real job-shop setting. In the first section, we present the data involved. An overview of the different techniques used to solve the problem is given in the next section. The chapter ends with the discussion of Genetic algorithm based heuristic.

5.1 Case Study

We applied the model in Chapter 4 on the data provided by Thunder Tools. Thunder Tools is a job-shop based manufacturing firm producing approximately 200 products. The products range from automotive components to B.B.Q burners. The production facility has a number of presses, varying in size and capacity. The annual demand for each product varied between 250,000 to 500,000. These presses are in five different areas. A product has a production sequence which involves these five areas. At most a product can have three different areas in sequence. We denote the areas of Thunder Tools as tasks in our model.

Based on the above data and considering two demand situations, in our model had 4,642,578 variables and 2,363,040 constraints. CPLEX solver took more than 24 hours to solve the problem (this run time is not acceptable for Thunder Tools). In some cases, software crashed without providing a solution. The goal of this chapter is to develop a solution approach to solve this real world instance.

5.2 Overview of the Unsucessful Techniques Used

In this section, an overview is given of the methods employed to solve the case study data.

As mentioned in the previous section, CPLEX took longer than 24 hours to solve the model. In order reduce the run time, we attempted to divide the problem into two separate but related small problems. The first problem is to determine the optimal production schedule, i.e., set of the variables x_{pst} from the previous Chapter. The production schedule was then passed into the second problem, which calculates for the minimum number of the workers required. We tested and compared five different problems of finding x_{pst} variables. We selected small problem instances to test our approach on so that we could compare the solutions from this approach to the optimal solution. Unfortunately, for the tested instances, the solution provided by these techniques was only within 30 % of the optimal solution (as provided by CPLEX).

In the second attempt, we tested different heuristics based on classical approaches of Lagrangian Relaxation [30] and Outer Approximation [31, 32]. For the small problem data sets, Lagrangian Relaxation provided an optimal bound within finite number of iterations. For the large problem instances, the heuristic did not converge quickly. The heuristic based on Outer Approximation provided the similar results. Neither approach provided us with high-quality solutions within the desired run time.

Next, we implemented and compared two independent heuristics. The first heuristic

was based on the LP relaxation techniques and the second heuristic used integer cut generation methods. For small test cases, both these heuristics provided near optimal solutions. Moreover, the run time performance for the LP relaxed heuristic is better than the CPLEX solver for most of the instances considered. However, both of these heuristics were not able to solve the case study data. In the next section, we discuss a Genetic Algorithm based heuristic that successfully solved the case study data.

5.3 Genetic Algorithm

Genetic Algorithms (GA) use the concept of evolution in order to solve large scale problems. Biological theory states that strong species have more chance in giving birth to better offspring. The same principle is adopted to solve large scale problems. From a set of candidate solutions, solutions having better objective function values are selected and a series of operations are applied to obtain new solutions. In order to understand the methodology behind Genetic Algorithms, the following terminology is presented:

- Population: Set of candidate solutions from which the parents are selected.
- Chromosome: A particular member of the population.
- Gene: A particular element of a chromosome.
- Parent: A particular solution which is selected on the basis of having a better objective function value than the rest of the solutions. A series of operations are applied on the parent to obtain a new solution.
- Offspring: After applying the series of operations, a new solution is obtained which is termed as offspring.
- Crossover operations: These are the operations that are performed on the parent solution to obtain an offspring solution.

- Mutation: A series of operations applied on the new solution in order to get a more diverse population than the previous population.

Genetic Algorithms contain a set of candidate solutions. Each solution of this set is called a chromosome and a fitness value is attached to each of them. Fitness value can be calculated as the sum of objective function values and the penalty associated with constraint violations. Every element of a chromosome is termed as a gene. By using certain heuristics, two chromosomes are selected as parents. Baker and Ayechew [33] use the binary tournament process to select the chromosomes. Tanomaru [34] uses the roulette selection method for choosing parents. A series of crossover operations are applied on the parents, resulting in two new solutions which are termed as offspring. Depending on the fitness value of the offspring, they are appended into the current population set which results in a whole new set of population. This new set of population has solutions better than the previous population set. The crossover and mutation operations are applied to get the new offspring and continue to do so until termination criteria are reached. The termination criteria can be based on the upper limit of the number of iterations to perform the crossover and mutation operations or when the whole population has converged. In order to decide if a population has converged or not, we assume a number ψ. The ψ is problem specific. The population is said to be converged if the following condition is met.

$$max\{F_i\} - min\{F_i\} \leq \psi \tag{5.1}$$

where $max\{F_i\}$ is the maximum fitness value of the current population and $min\{F_i\}$ is the minimum fitness value of the current population. Equation 5.1 states that when the difference between the maximum and minimum fitness value is less than or equal to the ψ, it is not productive to perform more crossover and mutation operations.

5.3.1 Application of Genetic Algorithm to Scheduling

Cai and Lee [12] have used Genetic Algorithms for solving workforce scheduling problems in a multiple objective problem. A set of solutions representing the number of workers of different types is considered as a population. A rank based selection method is used to select parents. They select two chromosomes randomly and assign them ranks based on the weights assigned to each objective function. According to the probability distribution function, chromosomes of rank 1 are selected with the specified probability or vice-versa. The multi point crossover is applied to the selected chromosome using a crossover mask. If there is a one in the crossover mask, then the gene is copied from parent 1. If it is zero, then the gene from parent 2 is copied. Mutation is applied to ensure solutions are different from the current generation. A heuristic approach is used to ensure feasibility of the constraints.

Baker and Ayechew [33] have employed Genetic Algorithms in vehicle routing problems. In their problem, a chromosome is the set of N customers where each gene represents the vehicle number to which each customer is assigned. The initial population has been generated using different heuristics. A 2-point crossover method is applied to obtain the set of new solutions. The results include the comparison of different crossover methods for the same data set, stating the suitability of a 2-point crossover method over a single point crossover.

Tanomaru [34] applied Genetic Algorithms with heuristic operators for staff scheduling problems. In his problem, each chromosome represents the set of shifts every employee has. The roulette method is used to select the parents.

5.3.2 Application of Genetic Algorithm for the Case Study

We propose a Genetic Algorithm based heuristic for solving the large problem instances (specifically our case study). The chromosomes in our algorithm are the production

schedules or x_{pts} variables and value of each of these variables are genes. The algorithm selects the best solutions from a series of schedules. A network model, explained later in this section, is solved for establishing the number of specialized and flexible workers from these schedules. The penalty for overage and underage of demand is calculated as the absolute sum of the difference in production and demand. F_i, sum of worker costs and penalty for overage and underage, denotes the function value corresponding to schedule i. In order to select the parent solutions, two methods have been employed separately: the Binary tournament and the Roulette selection. The algorithms for both selection methods are explained in the following sections.

5.3.3 Generating Initial Population

The inputs for the Genetic algorithm are the feasible production schedules that give the candidate solutions for the number of workers. The feasible schedules is generated in a number of ways. Below is the list of heuristics used to develop feasible production schedules for the initial population.

- No production

  One input for the feasible production schedule is zero production i.e. no production for all the products. In this case, the worker costs will be zero but this schedule comes with a very high penalty for not meeting the demand.

- Random demand

  Another input for a feasible production schedule is to generate random numbers, uniformly distributed between 0 and 9, for every product p and time t. the numbers 0 and 9 have been chosen empirically.

- Alternate random numbers

  We generate a series consisting of 0 and a random number, uniformly distributed

between 0 and 4, at alternate positions. The number 4 has been selected empirically.

- Equally distributed demand

  It is assumed that demand for every situation s is satisfied in the same time bucket, i.e. if for a particular product p, the demand at $t = 3$ hours is 10 units. The production for t less than three hours will correspond to this demand only and the load for production will be equally distributed in this time bucket.

5.3.4 Methods to Choose Parents

In this section, we present two methods to select the parents, the Binary Tournament method and the Roulette selection method.

Binary Tournament

In this method, two chromosomes are selected at random. Fitness value of both is recorded and the one with a better fitness value is selected as parent 1. Another pair of chromosomes is selected and the one with a better fitness value is selected as parent 2.

Roulette Selection method

Roulette selection method uses a pie-chart based approach to select the parents. The number of divisions in the pie-chart is equal to the cardinality of the population set, where each division represents a chromosome. The area of each division is a function of the corresponding fitness value of the chromosome. Two different functions were employed in the roulette selection approach.

In the first approach, we used the following equation for determining the area of

division for the i^{th} chromosome.

$$P_i = (1 - p_i) / \sum_{j \in POPULATION} (1 - p_j) \tag{5.2}$$

where $p_i = (WorkerCosts_i + Penalty_i) / (\sum_{j \in Population} WorkerCosts_j + Penalty_j)$

Since the problem is having minimization in the objective function, the area corresponding to each chromosome is not directly proportional to the fitness value. Equation 5.2 allows the chromosome with the higher fitness value to have the lower area in the pie-chart.

In the second approach, we considered the maximum fitness value inside the population. In order to allocate area to a particular chromosome, we calculate the deviation of the fitness value of that chromosome from the maximum value. The area for each chromosome i is calculated by the following equation:

$$P_i = p_i / \sum_{j \in POPULATION} p_j \tag{5.3}$$

where $p_i = Max\{F\} - WorkerCosts_i - Penalty_i$

The idea behind equation 5.3 is that the more the difference between the fitness value of a particular chromosome and the maximum value, the greater should be its chance of getting selected as a parent.

After allocating each chromosome its corresponding area in the pie-chart, we generate a random number in the range $(0, 1)$. Starting from the first chromosome, the cumulative sum of area of the divisions is compared with the random number. When the cumulative sum exceeds the random number value, the corresponding chromosome is selected as parent. Since the chromosome having a better fitness value will have more area on the

pie-chart, there is a greater chance that it will be selected as a parent.

5.3.5 Crossover

After selecting parents by using the Binary or Roulette selection method, the crossover operations are applied to get new solutions. The crossover mixes genes of the two chromosomes at specified positions. The following are some of the variations of crossover operations:

- 1-point Crossover : In the single point crossover, the first half genes of the offspring are from parent 1 and the rest are from parent 2.
- 2-point Crossover : In the 2-point crossover, two random numbers in the range from 2 to the number of genes in a chromosome are generated. The genes that have a position between the two random numbers are copied from parent 1 and the rest are copied from parent 2 to get the offspring.
- Multi-point Crossover : In multi-point crossover operations, for every gene, a random number in the range (0,1) is generated. If the random number is less than 0.5, then the gene is copied from parent 1. Otherwise, the gene is copied from parent 2. The set of random numbers generated can be referred to as a crossover mask.

5.3.6 Mutation

Mutation is helps to provide diversity in the new generation. With the probability of mutation p_m, a randomly chosen gene of an offspring is altered. With a pre-specified fixed probability the selected gene is changed to either 0 or a random number in the range (1,9).

5.3.7 Network Model for Calculating the Number of Workers

Given a schedule, a minimum cost flow approach is used to calculate the number of specialized and flexible workers required. In this model, we introduce a source node for every task a and another source node for flexible workers. The flow in the network is the number of specialized workers and flexible workers. Apart from the source node, every other node represents the time $t \in T$. The network is for a particular shift type $i(t)$. An example of the network is shown in Figure 5.1. We assume that there are three shifts in a working day. The network in Figure 5.1 is for the first shift. This figure assumes that the shift is four hours long and that there are two possible tasks. In the Figure 5.1, the starting nodes represent the source nodes for every task and another source node for flexible workers. The subsequent nodes represent time points. Based on the assumption that we have the same number of workers every shift and that a shift is four hours long, the next node is for $t = 12$ hours. The black solid line represents $s_{1,Task1}$ workers, those workers assigned to the first task during the first shift type. The black dashed lines represents the movement of flexible workers that were working on $Task1$ at time t. The red solid line is for the second task and the first shift type ($s_{1,Task2}$). Since, the number of specialized workers is not changing for a shift $i(t)$, we have a single solid line flowing back to the source node. The flexible worker can move across the tasks during a shift and will work for one hour at that task. Thus flexible workers can flow across the tasks.

We use this network to calculate the number of flexible and specialized workers, given a production schedule. The following sections explain this model in detail.

Sets:

As defined in Chapter 3.

Parameters:

As defined in Chapter 3.

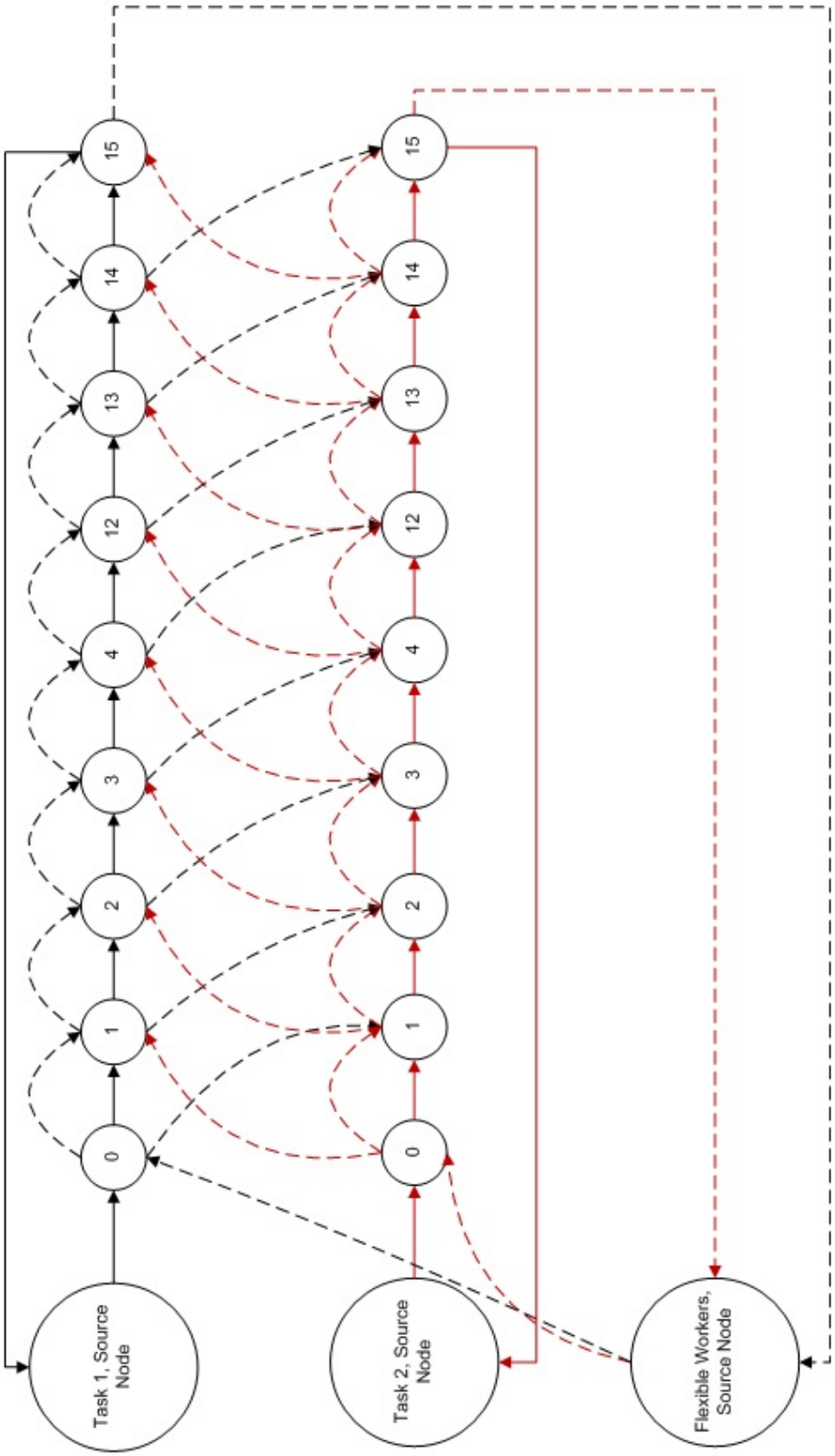

Figure 5.1: Network Model Illustration for Calculating the Number of Workers Required Given the Production Schedule

Decision Variables:

- Number of specialized workers required for task a, in shift type i: w^s_{ia}
- Number of flexible workers required in shift type i: w^f_i
- Number of flexible workers from source node to the task a, for shift i: $w^f so_{ai(t)}$
- Number of flexible workers from last node, of task a, to source node, for shift i: $w^f fo_{ai(t)}$
- For shift $i(t)$, Number of flexible workers working at task $a1$ moving to task $a2$, at time t: $w^f_{a1,a2,t,i(t)}$

Objective Function:

$$\min \sum_{i \in \mathbf{I}} \sum_{a \in \mathbf{A}} c^s_a w^s_{ia} + \sum_{i \in \mathbf{I}} c^f w^f_i \tag{5.4}$$

Constraints:

source node flow

$$w^f so_{ai(t)} = \sum_{a1 \in A} w^f_{a,a1,t,i(t)} \qquad \forall a \in A, t \in \{0, U, 2U\} \tag{5.5}$$

Last node flow

$$w^f fo_{ai(t)} = \sum_{a1 \in A} w^f_{a1,a,t-1,i(t)} \qquad \forall a \in A, t \in \{(Q-2)U-1, (Q-1)U-1, QU-1\} \tag{5.6}$$

Flow in = flow out except source and last node

$$\sum_{a1 \in A} w^f fo_{a1,a,t-1,i(t)} = \sum_{a2 \in A} w^f fo_{a,a2,t,i(t)} \tag{5.7}$$

$$\forall a \in A, t \in T/\{0, U, 2U, (Q-2)U-1, (Q-1)U-1, QU-1\}$$

Flow at flexible source node

$$\sum_{a \in A} w^f so_{ai(t)} = \sum_{a \in A} w^f fo_{ai(t)} \tag{5.8}$$

Flow greater than equal to production

$$w^s_{i(t)a} + w^f so_{ai(t)} \geq Xsum_{at} \qquad \forall t \in \{0, U, 2U\} \tag{5.9}$$

$$w^s_{i(t)a} + \sum_{a1 \in A} w^f_{a1,a,t-1,i(t)} \geq Xsum_{at} \qquad \forall a \in A, t \in T/\{0, U, 2U\} \tag{5.10}$$

$$w^f_{i(t)} \geq \sum_{a \in A} w^f so_{ai(t)} \tag{5.11}$$

Objective function (5.4) states minimization of worker costs. Constraints (5.5), (5.6), and (5.8) represent the flow balance constraints. Constraint (5.5) states that at the start of every shift i, the number of flexible workers working at task a is equal to the number of flexible workers moving at task $a1$ in the next time period, summed over all tasks $a1 \in A$. Constraint (5.6) represents the flow of flexible workers between source node of flexible workers and the last node of planning horizon. Constraint (5.6) states that the number of flexible workers returning to the source node from task a is equal to the sum of the flexible workers working at task $a1$ at time $t-1$, summed over all tasks $a1 \in A$. Constraint (5.8) represents the flow balancing in all the other nodes. The left hand side of this constraint is the sum of all the flexible workers working at task $a1$ at time $t-1$, who moved to task a at time t. The right hand side of the constraint is the sum of all the flexible workers who were working at task a at time t, and moved to all the other tasks. Constraint (5.8) states that the outward flow at the source node of flexible workers, the sum of the number of flexible workers going to all the tasks in shift $i(t)$, is equal to the inward flow, the sum of the number of flexible workers coming from all the tasks in shift

$i(t)$.

Constraints (5.9) and (5.10) represent the demand satisfaction constraints. $Xsum_{at}$ calculates the number of units required for a task a at every time t. At the starting nodes, the flow of workers constitutes only two types of variables, $w^s_{i(t)a}$ and $w^f so_{ai(t)}$. For the rest of the nodes of task a, the flow of workers is from both the preceding nodes and the nodes representing tasks other than a. Constraint (5.11) gives a final value of flexible workers needed for a shift $i(t)$.

5.3.8 Binary Tournament algorithm

This section details the algorithm, which uses the binary tournament method. In this algorithm J denotes the number of iterations done. The upper limit of the number of the iterations is assumed to be UB. Initially, POPULATION is an empty set. The algorithm generates the initial population members and append them in POPULATION set.

Set J = 0 and POPULATION = {}
Generate the initial population
while {J $\leq UB$ OR $max\{F_i\} - min\{F_i\} \leq \psi$ } **do**
    $i = 1$
    while {i$\leq$2} **do**
        Randomly select two parent solutions
        Check the fitness value of both
        Select the one with the minimum fitness value
        i = i + 1
    end while
    Apply a 2-point crossover to the parents selected in the previous step
    Apply the mutation
    Calculate the fitness value of the offsprings

14: **if** {The fitness values of the offsprings are more than the parent's} **then**

15: Discard the offspring solutions

16: **end if**

17: **if** {The fitness value of the offsprings lies between that of the parent's} **then**

18: Append them in POPULATION

19: **end if**

20: Discard the solution with the maximum fitness value from POPULATION

21: J=J+1

22: **end while**

23: Choose the solution corresponding to the minimum fitness value function

5.3.9 Roulette Selection

This section details the algorithm that uses the Roulette selection method.

1: Set J=0 and POPULATION = {}

2: Generate the initial population

3: **while** {J $\leq UB$ OR $max\{f_i\} - min\{f_i\} \leq \psi$} **do**

4: **for** { i in POPULATION} **do**

5: $P_i = (1 - p_i) / \sum_{j \in POPULATION} (1 - F_j)$

6: i = i+1

7: **end for**

8: Generate a random number r, uniformly distributed between (0,1)

9: SUM = 0

10: **while** { SUM $\geq$ r} **do**

11: **for** { i in POPULATION} **do**

12: SUM = SUM + P_i

13: **end for**

RETURN i when while condition is reached
end while
Repeat the above process to generate second parent
Apply 2-point crossover to the parents selected in the previous step
Apply mutation
Calculate the fitness value of the offsprings
if { Fitness value of offsprings more than the parents} **then**
Discard the offspring solutions
end if
if { Fitness value of offsprings lies between that of the parents} **then**
Append them in POPULATION
end if
Discard the solution having maximum fitness value from POPULATION
J=J+1
end while
Choose the solution corresponding to minimum fitness value function

5.3.10 Implementation and Comparison

The above algorithms were implemented using Python. The performance of the Genetic Algorithm is compared with the equivalent MINLP model, which is Case 5 from Chapter 4. The results below are for 25 products and cardinality of demand set for a product p at time t is 2. While generating the initial random population, we have considered the random numbers for the schedule at alternate points of time to be in the range (0, 9) and for the schedule having values greater than or equal to zero at every point in time to be in the range (0,4). The maximum number of tasks is three.

Table 5.1 displays the computational results of this algorithm for eleven instances.

The second column lists the number of iterations taken by the Genetic algorithm. As outlined in the third column, we have assumed that out of five tasks, every product needs three tasks to finish. The fourth and fifth columns list the Genetic algorithm and MINLP run time respectively. The sixth and the seventh columns list the solution obtained by the Genetic algorithm and by solving the MINLP. The last column lists the optimality gap for each instance.

Instance	No. of iterations to finish GA	Maximum sequence of the tasks	GA run time	MINLP run time	GA solution	Optimal solution	Optimality Gap
1	200	3	648.32	3377.94	2723	2748	-0.0091
2	200	3	653.12	965.67	2795	2768	-0.00975
3	200	3	649.9	1183.47	2761	2745	-0.00583
4	200	3	661.3	1668.19	2724	2841	-0.04118
5	200	3	647.17	836.54	2842	2871	-0.0101
6	200	3	677.37	1007.19	2817	2832	-0.0053
7	200	3	649.33	1853.72	2818	2801	-0.00607
8	200	3	654.85	946.15	2725	2863	-0.0482
9	200	3	649.14	1147.17	2761	2795	-0.0122
10	200	3	651.03	1019.04	2865	2805	0.01996
11	200	3	651.85	1751.11	2739	2849	-0.0391

Table 5.1: Comparison of GA with the MINLP

From Table 5.1, it can be easily verified that for all the instances considered, the Genetic Algorithm produces solutions within 4% of the optimality. The time taken by MINLP is less than that of the Genetic Algorithm for these data instances.Computation time is higher for these data instances however the Genetic algorithm outperformed MINLP for larger data instances.

While the MINLP was intractable for problems the approximate size of the Case Study, the Genetic Algorithm converged within a few hours. Table 5.2 presents different instances of problems that are the approximate size of the Case Study. For each instance, demand data and processing sequence is generated randomly. The number of iterations for each instance is set to be 200. The Genetic Algorithm was able to solve problems the size of the Case Study problem mentioned (from Section 5.1) in less than 3 hours.

Instance	No. of iterations to finish GA	Maximum sequence of the tasks	GA run time	GA solution
1	200	3	5870.3	23355
2	200	3	5796.87	23330
3	200	3	5783.05	23453
4	200	3	5865.29	22923

Table 5.2: Genetic Algorithm Solution of the Case Study Data

Chapter 6

Conclusion and Future Direction

In this thesis, we considered a manufacturing setting with uncertain demand across several product types. The manufacturing plant is composed of several workstations each dedicated to a specific task. For completion, every product must follow a predetermined sequence of the workstations, where a set of workers are required to complete the tasks. We assumed that this production environment had two types of workers: *specialized workers* who are only capable of working at a single workstation and *flexible workers* who are capable of working at any workstation. Our goal was to develop techniques to a schedules that minimized the workforce costs.

6.1 Conclusion

We began by considering the deterministic demand and developed models for two scenarios: 1) when tasks can be completed simultaneously, and 2) when there are precedence relationships between tasks. The computational results, strengths and weaknesses of each model are discussed independently.

To handle uncertainty, models based on Robust Optimization and Robust Mathematical Programming were developed and implemented. The models extend the deterministic

models and the results are presented for various values of the input parameters.

We attempted to use these models to solve a real-world Job-shop workforce scheduling problem. Unfortunately, CPLEX was not able to solve these large problem instances of our models. We implemented various approaches (Lagrangian Relaxation and Outer Approximation and relaxation based heuristics) to solve the problem. To test each of these approaches we used small test data sets and compared each of their results to the results from our math models. Often these techniques did give similar solutions to our math models for the small data sets. However, when these approaches failed to solve problems of the same size as the Job-shop Case Study described in Section 5.1.

We implemented a Genetic algorithm based approach. We conducted analysis to select the best approach to select parents. The computational results were very promising. The results indicated that the optimality gap between the Genetic Algorithm based approach and our math models was less than 4% for the tested data sets. Moreover, the Genetic Algorithm converged to solutions within hours for instances the same size as the Case Study problem.

6.2 Future Direction

This work could be extended in many ways. Possible extensions of the models discussed in this Thesis include capturing break times, tolerance for absenteeism and worker preference for particular shifts.

In addition, work could be done to extend this work to address with other types of demand uncertainty. In this thesis, we only considered uncertainty in the quantity demanded. In the future it would be important to address uncertainty in the demand arrival, i.e., when arrival time of the demand is not known. The emergency room in a hospital is one example of when this could occur in the service industry. Even though

the arrival of patients is uncertain, the nurses still need to be given a schedule. This nurse scheduling problem could relate to the work in this Thesis if one could devide the nurses in to two categories, specialized (restricted to one task) and flexible or floaters (capable of completing any task).

Lastly, this work can be extended by considering other types of uncertainty. Throughout this thesis, we have assumed that workers finish work at the same pace, and that the time to finish a task is deterministic. This is not always true in a practical setting. In several situations, the time required to finish a task as a random number. For example, in the construction industry, the time to finish a task is not always constant and is dependent on many other factors. In manufacuturing, workers can take different amounts of time to finish the same task. Perhaps the approaches described in the Thesis can also be extended to address these sources of uncertainty.

Bibliography

[1] Baker K., Workforce allocation in cyclical scheduling problems: A survey, *Operational Research Quarterly Volume 27 No.1(ii)*, 155-167, 1976

[2] Burns R., Narasimhan R., Multiple shift scheduling of workforce on four-day work weeks, *Journal of the operational research society, Volume 50*,979-981, 1999

[3] Brusco M., Jacobs L., Optimal models for meal-break and start-time flexibility in continuous tour scheduling, *Management Science, Volume 46, No. 12*, 1630-1641, 2000

[4] Arthur J., Ravindran A., A multiple objective nurse scheduling model, *AIIE Transactions, Volume 13, Issue 1*, 55-60, 1981

[5] Martello S., Toth P., A heuristic approach to the bus driver scheduling problem, *European Journal of Operational Research, 24*, 106-117, 1986

[6] Morris J., Showalter M., Simple approaches to shift, days off and tour scheduling problems, *Management Sciences, Volume 29, No. 8*, 942-950, 1983

[7] Lau H., On the complexity of manpower shift scheduling, *Computers Operations Research, Volume 23, No. 1*, 93-102, 1996

[8] Bartholdi J., A guaranteed-accuracy round off algorithm for cyclic scheduling and set covering, *Operations Research, Volume 29, No.3*, 501-510, 1981

[9] Ernst A., Jiang H., Krishnamoorthy M., Sier D., Staff scheduling and rostering: A review of applications, methods and models, *European Journal of Operational Research, Volume 153*, 3-27, 2004

[10] Tien J., Kamiyama A., On manpower scheduling algorithms, *SIAM review, Volume 24, No.3*, 275-287, 1982

[11] Glover F., McMillan C., The general employee scheduling problem: an integration of MS and AI, *Computers and Operations Research, Volume 13, No.5*, 563-573, 1986

[12] Cai X., Li K., Theory and methodology, A genetic algorithm for scheduling staff of mixed skills under multi-criteria, *European Journal of Operational Research, Volume 125*, 359-369, 2000

[13] Iravani S.M. and Van Oyen M.P. and Sims K.T., Structural flexibility: A new perspective on the design of manufacturing and service operations, *Management Science, Volume 51, No.2*, 151-166, 2005

[14] Brusco M., Johns T., Staffing a multiskilled workforce with varying levels of productivity: An analysis of cross-training policies, *Decision Sciences, Volume 29, Issue2*, 499-515, 1998

[15] Hopp W.J. and Tekin E. and Van Oyen M.P., Benefits of skill chaining in serial production lines with cross-trained workers, *Management Science, Volume 50, No.1*, 83-98, 2004

[16] Wirojanagud P., Gel E., Fowler J., Cardy R., Modelling inherent worker differences for workforce planning, *International Journal of Production Research, Volume 45, No.3*, 525-553, 2007

[17] Stewart B., Webster D., Ahmad S., Matson J., mathematical models for developing a flexible workforce, *International Journal of production Economics, Volume 36, Issue 3*, 243-254, 1994

[18] Jeang A., Flexible nursing staff planning when patient demands are uncertain, *Journal of medical systems, Volume 18, No.3* 125-138, 1994

[19] Sahinidis N., Optimization under uncertainity: state-of-the-art and opportunities, *Computers and Chemical Engineering, Volume 28* 971-983, 2004

[20] Campbell G., A two stage stochastic program for scheduling and allocating cross-trained workers, *Journal of the Operational Research Society, 62* 1038-1047, 2011

[21] Bard J., Morton D., Wang Y., Workforce planning at USPS mail processing and distribution centres using stochastic optimization, *Annals of Operations Research, 155* 51-78, 2007

[22] Soyster A., Covex Programming with set-inclusive constraints and applications to inexact linear programming, *Operations Research, Volume 21, No.5* 1154-1157, 1973

[23] Ben-Tal A., Nemirovski A., Robust Convex Optimization, *Mathematics of Operations Research, Volume 23, No.4*, 1998

[24] Kouvelis P., Daniels R., Vairaktarakis G.,Robust scheduling of a two machine flow shop with uncertain processing times, *IIE Transactions, Volume 32*, 421-432, 2000

[25] Dimitrov S., Information procurement and delivery: Robustness in prediction markets and network routing, Diss. University of Michigan, Michigan, 2010

[26] Bertsimas D., Sim M., Robust discrete optimization and Network flows, *Mathematical Programming, Ser. B 98*, 49 - 71, 2003

[27] Mulvey J., Vanderbei R., Zenios S., Robust optimization of large scale systems, *Operations Research, Volume 43, No.2* 264-281, 1995

[28] Yu C., Li H., A robust optimization model for stochastic logistic problems, *International Journal of production economics, Volume 64*, 385-397, 2000

[29] List G., Wood B., Nozick L., Turnquist M., Jones D., Kjeldgaard E., Lawton C., Robust optimization for fleet planning under uncertainity, *Transportation Research Part E, Volume 39*, 209-227, 2003

[30] Fischer M., An application oriented guide to lagrangian relaxation, *Interfaces, Volume 15, No.2* 10-21, 1985

[31] Duran M., Grossmann I., An outer approximation algorithm for a class of mixed-integer nonlinear programs, *Mathematical Programming, Volume 36*, 307-339, 1986

[32] Geoffrion A., Elements of large scale mathematical programming, part1:Concepts, *Management Science, Volume 16, No.11* 652-675, 1970

[33] Baker B., Ayechew M., A genetic algorithm for the vehicle routing problem, *Computers and Operations Research, Volume 30, Issue 5* 787-800, 2003

[34] Tanomaru J., Staff scheduling by a genetic algorithm with heuristic operators, IEEE International conference on evolutionary computation, 1995

Appendix A

Results of Deterministic Model without Precedence

Case 1: One product and three tasks.

Sets:

$T \in \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$

$A \in \{Task_1, Task_2, Task_3\}$

$P \in \{1\}$

$Y \in \{Flexible, Specialized\}$

$G_{Task_1} = \{1\}$

$G_{Task_2} = \{1\}$

$G_{Task_3} = \{1\}$

Parameters:

$d_1 : 10 units$

$h_{Task_1} : 1.5 hours$

$h_{Task_2} : 1hours$

$h_{Task_3} : 2hours$

c^s : \$10

c^f : \$20

$u : 8hours$

Results:

The optimal solution for this case is $w^s_{Task_1} = w^s_{Task_2} = 2; w^s_{Task_3} = 3; w^f = 0$. The cost will be \$ 70.

Case 2: Three products and ten tasks.

Sets:

$T \in \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$

$A \in \{Task_1, Task_2, Task_3, Task_4, Task_5, Task_6, Task_7, Task_8, Task_9, Task_{10}\}$

$P \in \{1, 2, 3\}$

$Y \in \{Flexible, Specialized\}$

$G_{Task_1} = \{1, 3\}$

$G_{Task_2} = \{2, 3\}$

$G_{Task_3} = \{1, 2, 3\}$

$G_{Task_4} = \{1, 2, 3\}$

$G_{Task_5} = \{1, 2\}$

$G_{Task_6} = \{1, 2\}$

$G_{Task_7} = \{1, 3\}$

$G_{Task_8} = \{1, 2\}$

$G_{Task_9} = \{2, 3\}$

$G_{Task_{10}} = \{1, 3\}$

Parameters:

$d_1 : 10 units$

$d_2 : 12 units$

$d_3 : 13 units$

$h_{Task_1} : 1.5 hours$

$h_{Task_2} : 1 hours$

$h_{Task_3} : 2 hours$

$h_{Task_4} : 0.5 hours$

$h_{Task_5} : 2 hours$

$h_{Task_6} : 1 hours$

$h_{Task_7} : 1.2 hours$

$h_{Task_8} : 0.8 hours$

$h_{Task_9} : 3 hours$

$h_{Task_{10}} : 1.2 hours$

c^s : \$10

c^f : \$20

$u : 8 hours$

Results:

The optimal solution for this case is $w^f = 2; w^s_{Task_1} = w^s_{Task_2} = 4; w^s_{Task_3} = w^s_{Task_9} = 9; w^s_{Task_4} = w^s_{Task_8} = 2; w^s_{Task_5} = 6; w^s_{Task_6} = w^s_{Task_7} = w^s_{Task_{10}} = 3$. The cost for this case is \$490.

Appendix B

Parameters for Minimization of Maximum Penalty Model

Case 1

Sets:

$T \in \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$

$A \in \{Task_1, Task_2, Task_3\}$

$P \in \{1\}$

$Y \in \{Flexible, Specialized\}$

$E \in \{1\}$

$S_e \in \{2\}$

Parameters:

$d_{111} : 1unit$

$d_{112} : 2units$

$h_{Task_1} : 1hour$

$h_{Task_2} : 1hour$

h_{Task_3} : $1hour$

c^s_{Task1} : \$10

c^s_{Task2} : \$11

c^s_{Task3} : \$12

c^f : \$15

u : $8hours$

Case 2

Sets:

Same as in Case 1

Parameters:

All other parameters are same, except

d_{111} : $1unit$

d_{112} : $17units$

Appendix C

Parameters for Deterministic Model with Precedence

Case 1

Sets:

$P \in \{1\}$

$A \in \{Task_1\}$

$T \in \{1, 2, 3, 4\}$

$Q \in \{1\}$

$I \in \{1\}$

Parameters:

$d_{1,3} = 12$

$d_{1,4} = 5$

$h_{Task_1} = 1$ hour

$u = 4$ hours

$\alpha_{1,Task_1} = 1$ hour

$n_{Task_1} = 1$

Appendix D

Comparing Two Different Functions in Roulette Selection

The results for comparing the two approaches used in roulette selection method

Penalty Value	Method 1	Method 2
5600600	0.0998	0.176294
3135200	0.09989	0.1771
502287200	0.0838	0.0226
7196400	0.0998	0.1758
11717200	0.0996	0.1744
365913800	0.0882	0.0648
350168800	0.0887	0.0697
350168800	0.0887	0.0697
350168800	0.0887	0.0697
575355400	0.0814	0
575355400	0.0814	0

Table D.1: Comparison of Two Methods Used in Roulette Selection