2008

Worker scheduling with induced learning in a semi-on-line setting

Patrick Rodd
University of Windsor

Worker Scheduling with Induced Learning in a Semi-On-line Setting

By

Patrick Rodd

A Thesis

Submitted to the Faculty of Graduate Studies
through Industrial and Manufacturing Systems Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science at the
University of Windsor

Windsor, Ontario, Canada

2008

Your file Votre référence
ISBN: 978-0-494-42283-0
Our file Notre référence
ISBN: 978-0-494-42283-0

Abstract

Scheduling is a widely researched area with many interesting fields. The presented research deals with a maintenance area in which preventative maintenance and emergency jobs enter the system. Each job has varying processing time and must be scheduled. Through learning the operators are able to expand their knowledge which enables them to accomplish more tasks in a limited time. Two MINLP models have been presented, one for preventative maintenance jobs alone, and another including emergency jobs. The emergency model is semi-on-line as the arrival time is unknown. A corresponding heuristic method has also been developed to decrease the computational time of the MINLP models. The models and heuristic were tested in several areas to determine their flexibility. It has been demonstrated that the inclusion of learning has greatly improved the efficiency of the workers and of the system.

Dedication

To my wonderful family

Thank you for all of your support

Acknowledgements

I would like to extend my warmest thanks to my supervisor, Dr. Guoqing Zhang, whom without his encouragement and guidance this thesis would have never come to fruition. Thank you for the wonderful experience and endless knowledge you have shared with me. A special thanks to my committee members, Dr. Reza Lashkari and Dr. Mohammed Baki, for your input and valuable advice given. Thank you for your time and effort to make me achieve my full potential. Thanks to Dr. Chunhong Chen, for being able to sit as the chair for my defense. A warm thanks to Ms. Jacquie Mummery and Ms. Brenda Schreiber for all of your help and advice.

Thank you to all of my family for your continuous support and encouragement, for without you, this would never have been possible. Very special thanks to Meiling Chen, who encouraged me to enter the Masters program and helped me through hard times. I wish to thank my brother Brendan for taking time to help edit my thesis. Thanks to Sicheng Chen, Pedram Sahba, Sanjida Rouf, and Ranjeev Basur, for your continued help and support, and to all of my friends who continue to help and motivate me through life.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

The first section of this chapter is used to provide a brief discussion of scheduling and its applications to maintenance areas, and to also discuss the impacts of the learning effects experienced by the operators. The second section is an introduction to the research project within this thesis.

1.1 General Overview

The notion of scheduling is not new and has actually been around for thousands of years. Without scheduling, some of the world's greatest achievements may have never come to fruition. It took explicit planning to achieve the pyramids, which were created more than three thousand years ago. More recently, the transcontinental railway spanning America was achieved through meticulous preparation. However, it wasn't until the early twentieth century, when Henry Gantt, influenced by Karol Adamiecki, came along, that scheduling was put into a recognized format. Henry Gantt, most commonly recognized for his work with the *Gantt chart,* was a mechanical engineer. The Gantt chart was an invaluable tool for Gantt's foremen, or other supervisors, as they were now able to determine if they were either on, ahead, or behind schedule. Even today, this method is built into software to help facilitate the same goal. One drawback however was that a certain level of expertise was required to complete a Gantt chart. Knowledge of each process was required to effectively designate time durations for each task.

In the 1950's a new scheduling method arrived, which is known as the Critical Path Method. This algorithm looks at all of the tasks required to meet a specific goal and the interdependencies of tasks. The critical path method, developed in a joint venture, was initially used for plant maintenance projects. The advantage of this algorithm was that the tasks could either be deemed as critical or not, allowing schedulers to prioritize tasks needing immediate processing to remain on the given time horizon. If the task was on the critical path, there was no leeway for the task's start and finish, however, those not on the critical path were more flexible with. Many more algorithms have been developed throughout the years, but all strive to achieve a similar goal: Organization.

Today, the term scheduling is synonymous in most industries, as it provides a wide array of purposes. It can be thought of as a decision making tool, assigning resources to tasks over a given time horizon. By definition alone, scheduling does not seem complex, but industrial sized problems require much attention as they tend to be large in scale. With the advent of computer technology, complex problems, thought previously to be unsolvable, are now able to be approximated or globally solved.

To narrow the discussion a little, we will now focus on scheduling within the manufacturing sector. Scheduling is a very important tool used in manufacturing, as it can have significant impacts on production and process organization. The main goals of scheduling within this industry are to keep costs low and productivity high, two very conflicting goals. There must be some common ground with which these two goals can compromise. It is therefore the scheduler's duty to allocate what jobs to produce, who will produce the jobs, what time and where will those jobs be produced. This can be a fairly daunting task if one were to tackle these problems alone, but with new technology

comes more user friendliness. Newer production planning tools far outperform manual methods. There are many new software programs available to be customized to meet the criteria for the particular scheduling task.

One area of interest in manufacturing is maintenance. If maintenance is neglected, costs can skyrocket as tasks are processed later and later because of machine failure. Due to the nature of maintenance areas, time is a major issue. When a task enters the maintenance area it typically needs expedient processing, as it will need to be reinserted immediately. Preventative maintenance can alleviate some of these time constraints as it will be known when a machine or task will need to be completed by. But there is also the case where a machine break down occurs and will need immediate attention. These jobs are particularly hard to schedule as it is a random occurrence. To accommodate these particular jobs, a newer area of scheduling research, on-line scheduling, may be applicable.

On-line versus off-line scheduling is a much more complex task, as on-line scheduling is performed in real time, whereas off-line scheduling has all the information provided before hand, allowing for a full schedule to be made. For this reason, on-line schedules are able to mimic real life scenarios more effectively.

As stated previously, on-line schedules are performed in real time, which ultimately raises the new issue of determining if the placement of the new job within the set schedule is optimal. For this task, competitive ratio analysis is performed. Competitive analysis checks the ratio of the on-line algorithm final solution to the off-line algorithm final solution to determine how close the on-line schedule is to the off-line. Because the information arrives one bit at a time, it is impossible to always achieve an

optimal solution, hence competitive analysis tries to find the worst case scenarios and work from there. By lowering the worst case ratio, we can at least guarantee that the schedule will fall within this range.

To further mimic reality, we can introduce the concept of worker learning. Workers are continually evolving as they progress through their careers. Schedulers should take full advantage of this. Traditional concepts of learning in a mathematical sense were introduced by Theodore Paul Wright in 1936 while working in the aircraft industry. It was he that first noticed that with each successive job, there was a limited time reduction. Wright went on to develop a mathematical formula that was able to depict such phenomena. It is a much newer concept than scheduling but is applicable nevertheless.

Traditional learning curve theory represents learning as only applicable to repetitive tasks, but cognitive skills do not cease for all other tasks. The mind simply retains the information in a different manner. Rather than learning the task, one may learn certain skills which are applicable to many different jobs as found by Allwood and Lee (2004). Another common name for the learning curve is the experience curve. There are several reasons for experience to accumulate, such as labor efficiency (workers become defter in performing tasks), better use of equipment, and shared experience effects. Experience curve effects are present when products exhibit common activities or resources (i.e. machines, workers). Efficiency learned from one product is able to be transferred to others.

Regardless of where scheduling is applied, it plays an important role. It has been used throughout the ages, and is continuously evolving. New scheduling programs are

being developed each day, and with each new idea, a new one is sparked. By simulating reality, a schedule can be extremely effective. Various scheduling techniques are applied within this thesis as will be discussed in the next section.

1.2 Proposed Research

1.2.1 Research Topic

Scheduling is a very broad topic, with many different facets of interest. The problem at hand is based off of a real problem, which was encountered in the summer of 2006. The problem takes place within a maintenance area, where workers and tasks need to be scheduled. There are two main classifications of jobs that may enter the maintenance area, which include (i) Preventative maintenance jobs, and (ii) Emergency jobs. Jobs consist of either molds or dies. Typical problems arise from simple wear and tear, however when a mold or die is completely out of its tolerable range, it is classified as an emergency job and brought to the maintenance area. When a job falls into the preventative maintenance category, it is already scheduled out of the processing rotation, but will still need to be scheduled into the maintenance area.

Because of these two classifications of jobs, we will mainly focus on the scheduling of preventative maintenance jobs due to the consistency of their arrivals. As was stated, preventative maintenance jobs are scheduled out of processing rotation, and arrive at the maintenance area at the beginning of the week. Once the total number of jobs to be processed is known, it is time to schedule the operators. The schedule aims to

simulate reality as best it can, we therefore introduce worker learning. As it was discussed briefly in section 1.1 and in more detail in chapters 2 and 3, workers are able to retain a certain amount of knowledge from previous jobs processed. Therefore the schedule utilizes workload to accommodate the growing knowledge base of the workers. As time progresses, and prior task knowledge accumulates, the operators should be able to progressively increase their workload. As workloads increase, the need for copious amounts of workers decreases, and shift sizes tend to decrease. The tasks that are being processed do not fit the typical learning curve theory, but it has been demonstrated that experience, rather than learning, can be applied.

In summary, a schedule must encompass worker allocation (i.e. shift and day), assignment to jobs (i.e. what worker will process what jobs). Experience plays a key role within a schedule, as it determines how many tasks a worker can be assigned. The two classifications of jobs fall under preventative maintenance and emergency. Preventative maintenance jobs contain certain information, such as approximate processing time and due dates. Emergency jobs can enter the system at any point in time during the existing schedule and must be processed immediately.

1.2.2 Research Methodology

To accommodate the given situations presented in the previous section (1.2.1), two mathematical models have been developed. The first model aims at achieving a schedule to accommodate both workers and preventative maintenance tasks. This model is discussed in chapter 3, section 3.2. The second model takes into consideration the

emergency jobs that enter the system. This model and its steps required are discussed in Chapter 3, section 3.3. The models were solved with LINGO 9.0 solver. As the problems deal with very complex topics, and due to the complexity of the models, a heuristic method is developed and tested in conjunction with the optimization models. Numerical results are provided to justify models.

1.2.3 Organization of Thesis

Chapter two is the literature review, where certain important topics are discussed concerning basic learning theories and its applications, learning applied to scheduling problems, scheduling in on-line atmospheres, and knapsack applications. The third chapter provides the development of the mathematical models used to solve the given problems followed by detailed explanations. Chapter four presents the heuristic models which were developed to approximate the optimization models. Chapter five outlines the testing and validation of the models through numerical tests. The thesis concludes in Chapter six with final thoughts on the results of the thesis and an outline of significant contributions developed within this thesis.

Chapter 2: Literature Review

The following chapter is comprised of four main areas of discussion, which include: Learning concepts, learning as it is applied to scheduling, learn-forget models reviewed, machine maintenance scheduling, and on-line scheduling. These topics are included in this chapter due to the validity of their conceptual ties with this thesis.

2.1 General Learning Concepts

In current literature, numerous studies have been devoted to human knowledge acquisition and retention. Several studies have been aimed at the manufacturing sector, which is the main area of this thesis. There are two main areas of learning in this sector, which include autonomous and induced learning. The former is simply learning by doing, while the latter is learning through training, reading, and any other external means.

Serel et al. (2003) tried to determine the optimal mix for investing in induced learning, rather than simply relying on autonomous learning, for process improvement. The authors assume the induced learning gained will jump them further down the same autonomous learning curve. However, Biskup and Simons (2004), create a cost function associated with induced learning. "A decrease in the learning rate down to the new one is possible by increasing learning costs" (Biskup and Simons, 2004). Both authors are striving to reach a common goal, to decrease the time it takes to produce a single unit.

Both methods prove to be efficient, but can produce different results. The former, assumes that a single learning curve, developed at the inception of the project, is to be

used. When learning is induced, the workers knowledge is increased, causing a leap down the same learning curve, (i.e. a worker had knowledge of x units, now after the induced learning, the worker has $x + y$ units, where y is the increase in knowledge). The latter however, assumes a cost increase when induced learning is chosen. Biskup and Simons (2004) proposed that as a company invests in knowledge, a cost, K(x), is incurred, and percentage reduction in the standard learning curve, s, is achieved. K(x) is a non-decreasing convex function, increased through the constraint $u = (1-x)s$, where u is the new learning rate achieved through induced learning. When there is no new learning, K(x) = 0, u = s, which is the standard learning rate. However, when induced learning is present, the standard learning curve will decrease, and K(x) will increase.

As can be seen, both provide similar aspects, but will produce different results. Serel et al. (2003) will reach the steady state of the learning curve more rapidly, as each learning instance will cause a jump up the same learning curve. Biskup and Simons (2004) will cause more rapid learning through the investment in knowledge, but achieve the steady state at the appropriate time. Both Serel et al. (2003) and Biskup and Simons (2004) models are valid, as learning is an unclear area.

In both papers, discussion is raised about optimal investments between autonomous and induced learning, but no discussion is given towards the actual methods of knowledge acquisition. There are many aspects affecting ones learning abilities such as age, educational background, the types of jobs being performed, and the methods of induced learning. One method of induced learning is through worker training which Tyler (2000) discussed in a magazine article titled "Focus on Training". The purpose of this article was to help employees take what they have learned (induced learning) and apply it

to their jobs. Some of the author's suggestions included proper scheduling (i.e. keeping the training closer to the date it will be needed and training in sessions to let the information sink in), use of technology (i.e. computers at workstations to give on-site information), and approximate real life (i.e. keep examples that will mimic the actual job that is being trained). The author states, "To foster training, create an environment that encourages use. First, ensure that trainees will have the necessary tools and equipment to use new skills upon returning to the workplace. Then, enlist managers' help in providing opportunities and a pace of workflow that allows for new skills to be applied" (Tyler, 2000). Another important issue is that the worker should have "the opportunity to use what they've learned within the first two or three days" (Tyler, 2000). The author notes that if this time period is not met, then some of the knowledge will begin to decay and be lost.

Training is a valuable asset, but must be performed with care. Each person will respond to training differently, such as the elderly. "Traditional training programs designed for the younger worker may need to be reevaluated to meet the current corporate change in manpower demographics" (Ford and Orel, 2005). As Ford and Orel (2005) discuss, "job site training and a high school diploma may no longer be sufficient to learn and operate technology with extensive computer dependency". The authors note that when training elderly workers (over the age of 55), training will tend to be slower, and retention tends to decrease. "Working memory or the ability to preserve information while processing similar or different information at the same time may be limited in older adult workers because of delayed processing speed" (Ford and Orel, 2005). However, there are advantages to older workers such as worker expertise within their field.

"Generally, many of the current jobs require expertise that comes with experience rather than just intelligence or skill training. Older workers in the workplace can provide the needed expertise" (Ford and Orel, 2005). Adler and Clark (1991) noted that compared to a simple learning curve methodology, experience turns out to have an extremely powerful effect on productivity once engineering and training are accounted for.

Learning is an important part of industry. It not only affects the worker but can spark changes in products and create new understandings of the process itself. Learning typically follows a cyclical pattern. New product innovations arise from engineering changes, and the ever increasing demand causes the rise in cumulative output. "Training should be directly related to cumulative output: as the plant accumulates experience and learns about the process, it is likely that the things they learn involve readjustments of processes and procedures that will call for new training" (Adler and Clark, 1991). An example of an engineering change that could arise is when a customer requires new specifications for an existing product, which in turn will require a refinement to the process, ultimately sparking the need for more training.

One conclusion that Adler and Clark (1991) reached was that learning can vary very substantially across different processes and jobs. Job and process complexity can have large implications on time taken to complete a task. Bailey (1989) performed a test to see the retention of two tasks; one was a complex procedural task, while the other was a control task, with little attention needed. The participants were trained for either four or eight hours, and worked on the tasks for another four hours uninterrupted. After a time duration, the participants were called back to resume the tasks for another four hours, after which the author made some conclusions. What Bailey (1989) found was that after

the time duration there was no real significant change in the control task, but for the procedural task, considerable time fluctuations occurred. The author was able to conclude that "forgetting of the assembly task is a function of (1) the amount of learning prior to interruption; (2) the elapsed time of interruption; but not of (3) the learning rate or (4) the initial assembly time, or (5) demographic and other variables" (Bailey, 1989). This conclusion demonstrates the need to consider the task at hand when trying to develop a training program, and the results that are to be expected. A worker cannot be expected to demonstrate a high level of learning for such complex tasks, as they require time to procure the needed skills for each complex task. Only after a worker gains experience can the learning be reflected. The worker will produce fewer errors, overall task time will reduce, and the quality of the product tends to increase.

Once the worker has attained a certain level of experience and is more confident, a sense of need may become present. The worker will need motivation to continue learning, as intrinsic motivators may have been the root cause for the initial learning, extrinsic motivators may need to surface to continue the want and desire for further learning. Extrinsic motivators may include bonuses, or other monetary means, benefits, or simply paid vacation time. In a study by Gow and Kember (1990), relating to a group of students choice of study, "several of the students commented that their motives for doing the course were to obtain a qualification rather than because of an interest in the subject" (Gow and Kember, 1990). Extrinsic motivators can be powerful, but special care needs to be taken when choosing and implementing them.

Although motivation can play a part in driving a worker to learn, a worker can only handle so much. So does education play any role in further increasing one's ability

to learn? In the same study mentioned above by Gow and Kember (1990), it was determined that there is no real evidence that workers with higher education have a more adept ability to be an independent learner. Those that attend higher level institutions may gain insight into problem formation and attain a certain mind set, but there is no real evidence that people with lower education have less likely learning abilities.

Many people discount learning as a simple byproduct of time, but it can be a powerful ally. There are many ways to harness this ally, whether it is through worker education, training, or simply learning by doing (autonomous), it is the manager's decision how to best implement and manipulate the situation. Workers are all different, and in that respect, learning programs must be tailored to suit the workers. Learning can also be applied to product and process knowledge. It can spark new and innovative designs. In short, learning should not stop at the job, but should be a way of life.

2.2 Learning as it is Applied to Scheduling

There are several papers dealing with various topics of learning as it applies to scheduling. Several of these papers utilize learning effects to reduce task time for repetitive jobs. One of the first people to note this effect was Theodore Paul Wright in 1936. While working at the Wright-Patterson Air Force Base, Wright noticed that every time the production quantity of an aircraft doubled, the time taken to produce them was reduced. Wright was one of the first people to quantify this relationship. Since then, many researchers have been applying this effect to the manufacturing industry.

Learning is not limited to only the manufacturing sector, as was shown by Amor and Teplitz (1998). The authors extended the traditional learning curve methodology to "describe an efficient method for approximating a project's composite learning curve" (Amor and Teplitz, 1998). Working in industry can be very competitive, especially when bidding for jobs, as it is with construction. Amor and Teplitz (1998), worked to produce an improved model to offer managers a far more reliable method of estimating composite project duration. In previous designs, the time duration for each task of the project would be calculated through copious amounts of calculations. The authors were able to change this technique, and eliminate many calculations, saving time and reducing the possibility of error. This elimination however proved to produce significant error of up to 290%; hence the authors augmented the integral, which calculates the total project completion time, for the project's first and n^{th} duration by 0.5, drastically reducing the error to a minimal 3%. The former method would have ultimately shown no error in estimation, but was highly inefficient. The authors made an interesting note that when contracts are bid in Japan, if the contract due date is not met, due to improper scheduling, the general contractor is liable and can end up owing money to the investors, demonstrating the need for accurate project estimation.

Amor (2002) continued research in this area, trying to apply a tangent method for a faster approximation of the total project completion time. What the author found is that although the tangent method could provide a somewhat faster result, it was still not as accurate as Amor and Teplitz (1998). It was therefore Amor's conclusion that the tangent method could be used for a quick initial estimation, but the secant method of Amor and

Teplitz (1998) should still be used over the traditional full blown method, as the error produced was relatively small and could produce useable results quickly.

Arditi et al. (2001) also worked within the construction industry, utilizing fuzzy logic to decipher ambiguous language. The author's stated that the "learning rate of each construction activity should be estimated and then each activity's worker-hour estimates should be adjusted for each unit of production, based on these learning rates" (Arditi et al., 2001). Within this context, the workers are divided into crews, and once enough learning has been accomplished, these crews are released or laid off, as they were still able to maintain the level of building with fewer workers. This is possible because the learning effects reduce the time to produce a single unit, or in this case, a single task, relinquishing the need for many workers. Arditi et al. (2001) developed a model that could still maintain the level of output required, while minimizing the necessary workforce.

The previous three papers were dealing with the construction industry, but this thesis' main area of focus is in manufacturing. Mosheiov and Sidney (2004) focused on this area, trying to minimize the tardiness of jobs with a common due date. The model was formulated to create induced learning, hence it was the managers responsibility to instigate the knowledge (or training) when it would be necessary, rather than simple autonomous learning. The author's findings were to invest heavily in knowledge creation early on and invest more resources later in the production output.

Mosheiov and Sidney (2004) believed knowledge creation early on proved best. However, these learning estimates are still sensitive to differing environments. Finch and Luebbe (1991) determined two key points. The first was that slower learning rates

inherently have larger variances, resulting in larger errors. Secondly, the larger the production runs are the more sensitive the project is to learning rate variability, i.e., the learning rate can grossly over or under estimate the time needed for the production completion time. It is therefore the authors opinion that risk should be assumed in the model, and that learning effects present should be thought of in a probabilistic manner.

Prior to Mosheiov and Sidney's (2004) paper, they worked in a single machine atmosphere with job dependant learning curves. Mosheiov and Sidney (2003) focused on classical single-machine objectives such as makespan and total flow time, on a due-date assignment problem, and on minimizing total flow time on unrelated parallel machines. The author's approach to solving these problems involved many tedious calculations. For each job, the processing time would be calculated with respect to its position in the schedule for its specific learning rate. So for ten jobs there would be one hundred calculations. Once these values were found, a simple program would be invoked to find the desired results. This method however seems flawed in that it can greatly overestimate the effects on learning for longer jobs, i.e. schedule a job with a shorter estimated processing time first, and reap the benefits of scheduling the job with a larger processing time second. The learning curve theory utilizes a doubling effect, where the time decreases when production doubles. This thesis addresses this issue, to encompass a better usage of the learning curve approach.

Biskup and Simons (2004) worked on a similar topic as Mosheiov and Sidney (2004). Biskup and Simons (2004), however, worked with several due date objectives in a single machine environment. The major objectives were to find the optimal scheduling policy with the correct amount of induced learning. The appropriate measures for induced

learning stemmed from material concerns. An order for material may be due at a certain time. If the material is done too early, capital is tied up; if it is too late, the customer may be dissatisfied and other costs may be incurred. The authors solved their problem through assigning positional weights to each pending job, then through a simple matching of longest jobs, to smallest positional weighting, a schedule is made (i.e. the farther down the job is scheduled, the more it is affected by learning). Because simple autonomous learning may not provide the means to meet such demanding due dates, induced learning is included in the mix. Biskup and Simons (2004) solved many types of scheduling rules as stated above, including: Earliness and tardiness penalties; Penalty for assigning a late common due date; and a Penalty on the completion time of the jobs. All are simple extensions of the original problem, with slight variations. As with Mosheiov and Sidney (2004), Biskup and Simons (2004) address the combination of induced and autonomous learning in scheduling. The latter however, seems to address similar jobs with different processing times in a more proper fashion, and seems to estimate the learning effects more appropriately towards longer jobs, where the former may overestimate these benefits.

Stratman et al. (2004) took a different approach to learning effects, focusing on the comparison of temporary versus fulltime skilled workers, to determine how learning was implemented in a manufacturing atmosphere for each type of worker. The authors' study was aimed at determining how a mixture of permanent and temporary workers affected the manufacturing performance, the impact of these workers in different areas of processing (i.e., quality assurance or building work stations), and how the performance varied with different lot sizes and product complexity. Pro-Model 3.01 discrete-event

simulation software package was used to model the assembly process. The authors' main findings were that more temporary workers will detract costs from worker compensations but more hidden costs tend to arise. Temporary workers tend to have a lower level of learning ability thus accomplish less in a given shift. Worker assignment also impacts cost reduction as "companies employ a mixed temporary and permanent workforce with heterogeneous skill levels, workers' placement in the production systems may affect variable manufacturing costs, due to differences among the workstations such as utilization rates and task complexity" (Stratman et al., 2004). The authors' found that the largest reductions in manufacturing costs were created with skilled workers being placed in an upstream job. A final conclusion reached by the authors was "the costs of learning and forgetting incurred by temporary workers must thus be taken into account when determining the variable manufacturing costs of a particular workforce deployment policy" (Stratman et al., 2004).

Another crucial area where learning is applicable in the manufacturing area is found within setup times. Pratsini (2000) addressed this issue by extending "the lot size model with setup learning to incorporate capacity restrictions for the single level, multi-item case" (Pratsini, 2000). The author notes that most of the learning reductions will take place in the early stages of production; hence, the author deduces learning is most beneficial for short term problems. Pratsini (2000) found that when there is a low setup/holding ratio, lot for lot production is achieved, whereas when the ratio is high, setups are no longer cost effective until a higher rate of learning is achieved. The author notes that "considerable learning is required to bring down the high costs and achieve a lot for lot production" (Pratsini, 2000). Lastly, as setups increase, batch size typically

decreases, resulting in lower holding costs. But if the time to perform the setup also decreases, as learning would dictate, the setup/holding ratio should stay relatively the same. So in the beginning of production, larger batch sizes may be profitable, but as time progresses, assembly will slowly lead to a lot for lot production.

Although much improvement can be seen within the setup time of a single job, the savings can sometimes be limited. Another area to look into for time savings is the skills of the workers involved in the processing of the job. Allwood and Lee (2004) researched this area, delving in to the method of job rotation and its effects on problem solving skills. The authors' main intentions were to create a model to show how problem solving skills can be learned and applied to other similar jobs. What they created was a simulation model to carry out different scenarios. Allwood and Lee (2004) found that the learning rate and forgetting time have the biggest effects on the solution. As forgetting time increases, problem solving skills start to diminish; therefore large periods of time where no learning is present can sometimes cause total forgetting. Although some studies have stated that relearning rates tend to demonstrate faster learning, this would present a whole new issue than the one at hand. Another issue is that as problem types increase, learning is much slower as the operators must learn more skills to cope with the wider variety of problems. The authors' simple goal of trying to attain a more equal workforce is negated by the fact that worker learning is too slow, and the diminishing effects are too great for the problem instance they are working towards. Allwood and Lee (2004) concluded that job rotation does not improve overall problem solving skills or productivity. It is actually more productive to have an operator specialize in one specific problem area than to expose them to multiple problem types, without any rotation. The

author made reference to job rotation having more to do with motivation than with a method to improve processing speed. Lastly the author notes that it is possible to transfer problem solving skills gained from one job to be applied to others.

The current literature in this area is fairly widespread and somewhat limited in its approaches. It has been shown by many that learning can be applied to the area in which this thesis is proposing. Most authors have focused learning effects as it applies to a single product being produced and learning from each subsequent unit. The proposed research applies learning as a method of overall production output. As the operator acquires valuable experience from working each day, the workload of that operator will slowly increase. This new approach should provide a more accurate depiction of how learning can be applied to the manufacturing sector with somewhat related tasks.

2.3 Learn-Forget Models Reviewed

There are several papers which discuss learning, but the counterpart to learning is forgetting, which is a very common practice in most individuals. Many authors discuss forgetting as a decay of learning; ultimately leading to an individual having to re-learn what it was that was forgotten. This area however is a very fuzzy topic, as there is no real way to measure one's forgetting. Several models have been proposed to depict learn-forget cycles, but they are still only estimates and must be viewed as such.

Jaber and Bonney (1996) proposed that the forgetting slope is mathematically dependant on the learning slope and the quantity of product produced to date. The paper investigates the effects of learning and forgetting on both the optimum production

quantity and the minimum total inventory system cost. The authors assumed that when learning is stopped, forgetting begins, however this is only an assumption. Once learning has stopped, the model determines how much experience the worker has gained, which will slowly begin to decay the longer the worker is away from the working environment. When the worker is to begin production again, the model will determine how much experience has been lost, and a new value of jobs completed will be assigned to that worker.

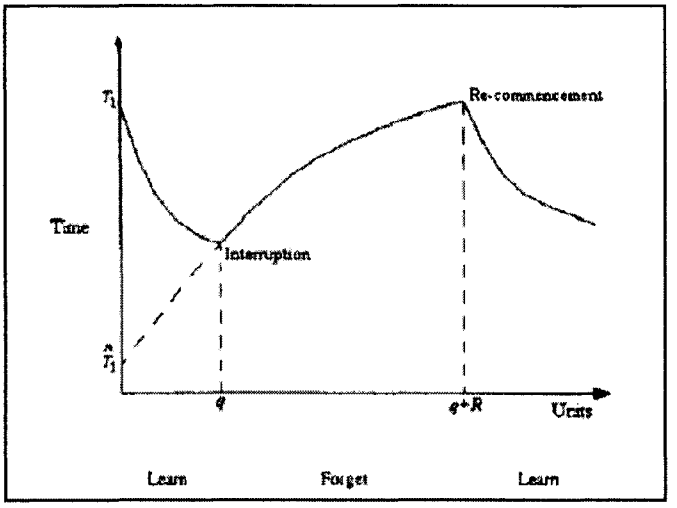


Figure 1: "The decrease and increase in labor hours due to the learning forgetting effects" Jaber and Bonney (1996)

Figure 1, shows a cycle where q units are produced, at which time an interruption transpires. At q + R units total forgetting occurs, indicating that a significantly long period, which is user defined as per the authors' model, has taken place. As with typical learning curve theory, each unit of production produces a small reduction in time, visible on the graph between 0 and q and after q + R on the x-axis. The period between q and q + R indicates the period of forgetting, which produces a regression in units produced, as per the model. Jaber and Bonney (1996) found that the forgetting slope was dependant on the length of the interruption, the amount of accumulated output at the time of

interruption, the time duration of total forgetting, and the learning rate itself. Possible error lies within the author's original assumption that forgetting occurs as soon as production stops. Some brain activity will still be evident, as some people may still think about work at home. In the authors' example, many days pass before resuming operations, hence forgetting in this case will be evident.

In 2004, Jaber and Sikstrom worked on a comparative paper. The authors compared the previous work of Jaber and Bonney (1996) (Learn-Forget Curve Model, LFCM) and Nembhard and Uzumeri (2000) (Recency Model, RC), with a new model presented by Jaber and Sikstrom (Power Integration Diffusion model, PID). Within the comparative paper, the authors note some distinct characteristics about each model. The model presented by Jaber and Bonney (1996) provided accurate results when working with a single production break. Jaber and Bonney's paper is described above in more detail. The second paper by Nembhard and Uzumeri (2000) presents a model based on how recently the last job has been performed. Jaber and Sikstrom (2004) suggest that this model is lacking as it ignores a fundamental law known as jost's law, which suggests that newer memory decays faster than older memory. Jaber and Sikstrom (2004) developed a newer model to address the issue of multiple breaks by introducing a memory trace. Through this the authors were able to more accurately depict the learn-forget cycle. By comparing all three models the authors were able to reach some conclusions. First, the authors note that the PID model is most applicable for more cognitive tasks whereas the other two models (LFCM and RC) are more related to motor skill tasks, however, when learning is a mixture of the two, the models produced similar results. Secondly, the PID and LFCM models both demonstrate that when learning is slow, forgetting is fast,

whereas the RC model does just the opposite, which contradicts Jost's law. Lastly the author's make some general rules regarding learn-forget models: (i) the experience gained before production stops, influences the amount of forgetting; (ii) the duration of the interruption influences the amount of forgetting; (iii) relearning rates tend to increase after breaks; (iv) a power function is suitable for modeling forgetting; (v) learning and forgetting are opposites, i.e. learning is increasing with time, forgetting decays with time; (vi) the amount of forgetting is positively related to the learning rate; (vii) there are only two types of tasks, either manual or cognitive. These rules provide a base for learn-forget models to be applied.

Jaber and Kher (2004) worked on a model to improve on Jaber and Bonney's (1996) work by assuming that the time for total forgetting was a function of the experience the worker has accumulated at the point of interruption. Through least-squares regression, the authors showed that the forgetting curve could be approximated to a linear function. The authors were able to improve the paper by Jaber and Bonney (1996), as the previous model tended to overestimate the performance of the relearning rates. Through the modified learn-forget curve model, the authors were able to provide a more accurate cost estimate to the relearning rates of workers. The authors note that more field research is necessary to determine the validity of the research.

Most of the current literature on learn-forget models tends to dictate that the duration of the break is one of the most crucial links in the forgetting phenomenon. The length between jobs for the current thesis is negligible; hence the only forgetting could occur when the operator finishes their respective shift and will resume learning upon arrival the next day. The duration of time where learning is negated is very short and

forgetting should be brought to a minimum. It is therefore assumed in this thesis that forgetting is negligible and will henceforth be omitted.

2.4 Scheduling Machine Maintenance

Machine maintenance is very important as it can prolong the life of the machine. Simply replacing the machine can be very costly and an unnecessary cost if proper maintenance policies exist. The topic of scheduling machine maintenance is still a widely studied subject. The necessity simply exists due to the ever increasing demands placed on manufacturing facilities. With increased production comes an increase on the strains of the machines. A machine can only function so long before it begins to degrade. Hence, there are many papers which discuss various methods to include maintenance into the flow of production.

Ruiz et al (2007) presented a paper discussing certain aspects of machine maintenance. The authors noted that there are two main types of maintenance, (i) preventative maintenance, and (ii) corrective maintenance. The difference between the two is that preventative maintenance is known and scheduled maintenance, whereas corrective maintenance refers to a failed machine which needs immediate attention. Hence the authors state that "in order to keep a minimum level of efficiency it is necessary to carry out maintenance operations during the operational life of the system or machine" (Ruiz et al, 2007). It is the goal of preventative maintenance to keep the machines in good condition so as to limit the possibility of an unexpected failure. The authors strive to incorporate a maintenance schedule that allows for the machines to

operate at a high level of reliability. This entails the inclusion of maintenance within the set production schedule to be carried out at periodic intervals. Two approaches are taken, (i) maximization of machine availability, and (ii) maintaining a minimum level of machine reliability after the production period. Due to the complexity of the schedules in questions, a series of heuristics were introduced to approximate the optimal solutions. The heuristics included: Ant Colony, Tabu Search, Simulated Annealing, and Genetic Algorithm. The Ant Colony heuristic prevailed with the closest results to the optimum for both approaches. Ruiz et al (2007) also wanted to stress the importance of the inclusion of maintenance while the schedule is being created. Therefore a further comparison was embarked upon to determine if maintenance could be merely thought of as an after thought, or if it should be included within the schedule as it is being created. The authors found that there was a 30% decrease in time, if the schedule included maintenance as it was being created. Therefore the authors concluded that maintenance should be included within the scheduling criteria, rather than as a simple after thought.

Similar to the previous paper, Kubzin and Strusevich (2006) worked with maintenance scheduling in both a flow shop and an open shop. One key difference was that the authors utilized a decay function based on time. Simply put, as machine life increases, the more maintenance it will require. Therefore as time increases, maintenance time increases. The authors were looking at the optimal time to implement maintenance within the outlined systems. To simplify their model however, they only assumed that one maintenance period needs to be implemented over the scheduled period. This assumption, as Kubzin and Strusevich state, does not necessarily parallel reality as many maintenance periods may be required over the life of the machine. The overall results of

the models indicate that maintenance should be implemented earlier on in the schedule as maintenance times were at their shortest in the beginning of the period.

These two papers share similar outlooks on maintenance as they both regard maintenance as a crucial step in the scheduling process. Rabinowitz et al (2000) however, dealt with multiple machines, each with their own respective maintenance period. The authors' main goal was to provide a continuous job with a continuous feed if possible. Within the scope of the problem, two machines were utilized for the continuous job. It was of no use to have both machines running at the same time, so a maximum of one machine was to be running at any given time. The machines fit into three main categories which were working, maintenance, and ready. Ready denotes a machine that has undergone its maintenance and is waiting for a job to process. The authors were unable to find a suitable heuristic to provide optimal solutions. Instead, they developed a pseudo heuristic which created multiple feasible solutions, where the best solution could be picked. This pseudo heuristic provides very close to optimal solutions with an operational rate of 99% on average. The authors note that a stochastic version of this problem would better simulate reality, but due to complexity, they worked with a deterministic version. Rabinowitz et al (2000) provide good insight into the problems of continuous jobs. The authors regard maintenance as a crucial part of production. The degradation of the machine is imminent and actions must be taken to prevent unexpected breakdowns. By incorporating maintenance, one can lower the probabilities of these unexpected occurrences, but they can never totally disappear.

Similar to the work of Rabinowitz et al (2000), Chen and Liao (2005) worked within a textile manufacturing facility with a continuous process. Chen and Liao looked

at the impact of several maintenance policies while minimizing job tardiness. Scenarios included both scheduled and unscheduled maintenance and varied from simple tasks such as changing small parts to the extreme of machine overhaul. Although there are many machines, the authors utilize a single machine model, as it is assumed that the other machines act as peripheral machines, feeding the main machine. The authors note that although these machines can break down and cause delay to the main machine, it is infrequent that this happens and typical problems are menial with little maintenance time required. The gist of the problem lies in the main machine. The new results provided an improvement in the tardiness of jobs of approximately 30%. When implemented in the real world industry the results were not as prevalent but there was an improvement. The reason for the discrepancy was due to job cancelations and changes. Overall, Chen and Liao provide a good base for maintenance situations faced by several types of industries.

Maintenance is ever prevalent in industry and it always will be. The research in this area is wide and covers many different aspects. As this thesis works in a manufacturing setting, the papers reviewed have followed that theme. The papers work with several aspects of maintenance scheduling, but all aim at a common goal, which is the importance and inclusion of maintenance in a formal scheduling procedure. Although this thesis does not implicitly schedule the tasks to go into maintenance, it relies on these maintenance schedules to produce the tasks assumed in the maintenance area. The necessity for maintenance scheduling is imperative to the successfulness of a working schedule.

2.5 On-line Scheduling

The prospect of an on-line scheduling program is that it allows a schedule to be formed without having all of the input available at inception. As new data, such as jobs, become available, the schedule is able to adapt to the new changes without affecting the optimal solution too greatly. There are many advantages to having an on-line system; one major component is that it can mimic real life applications more closely. In real life, you may not know how long a job will take to completion, or you may not know when the next job will arrive, hence the relevance of the on-line algorithm. To evaluate an on-line algorithm, competitive analysis is used, where the on-line program is compared to the optimal off-line solution.

Li and Huang (2007) proposed an on-line scheduling program to minimize the total makespan for a list of jobs with random release times in a multi-machine setting. The authors are improving upon an earlier paper, trying to reduce the competitive ratio from 2.93920 to 2.78436. This ratio depicts the maximum ratio that can be achieved for the worst case scenario between the on-line schedule objective value and the optimal off-line schedule objective value; hence a lower bound is better. The authors' algorithm takes a new job that enters the system and uses the information given (production time, arrival time) and uses that to build it into an existing schedule. First they try to determine if there is an idle position on any machines that can accommodate the new job. If no machine can do this, the algorithm checks if the job is released before machine one, which will complete all of its jobs first, is completed. If this is the case, the new job is assigned to machine 1 and will resume at the end of machine one's original sequence. Next it will

check if the job is released before any other machine completion time, in this case it will assign the job to the end sequence of machine k. No idle times will appear for any of these new jobs in the last two steps as they will be processed immediately after the machine finishes its last job. The last step is if the release occurs after the last job completes on the last machine. The job is assigned to the last machine and a new idle gap will appear. The authors prove the model and the competitive ratio to be true and make a final remark that it would be very hard to achieve a competitive ratio less than two as they previously proposed in the earlier paper. This on-line algorithm was designed to minimize the idle times between jobs, ultimately minimizing the total makespan.

Adzakpa et al. (2004) worked on a method to deal with preventative maintenance jobs in an on-line atmosphere. Varying release times among machine resources provided the on-line source of variance. The purpose of the project was to minimize the human workforce through proper scheduling. As the problem Adzakpa et al. worked on was NP hard, a heuristic was developed to deal with the management of the preventative maintenance jobs that occur in an on-line manner. The authors proved that the heuristic has low complexity through various problem instances tested. The algorithm works by first checking the theoretical time between jobs and each job's theoretical starting time. Next the algorithm checks for the processor's availability by first checking how many jobs are available at a given time, then checking how many fall in the urgent category, and finally assigning the remaining jobs to all available processors. If two or more processors are available at a given time, a job is assigned to the processor with the earliest availability time. This algorithm is capable of including a large number of

machines and processors, while still maintaining a fairly close approximation to the offline schedule for the large problems.

Gambosi and Nicosia (2000) looked into modeling various jobs with setup costs. The authors developed a heuristic to assign jobs to various machines, each with a corresponding setup and execution time. "If a task of a certain type is assigned to a machine that has just completed the execution of a task of the same type, then it can be processed immediately. Otherwise, there is a setup time associated with switching the machine to a different task type" (Gambosi and Nicosia, 2000). The algorithm works by first assigning different jobs to the machines throughout the system. All the jobs will be processed until a new job, not currently setup on a machine, needs to be accommodated. Once this critical juncture occurs, the algorithm looks at all the machines and checks all the current tasks to make sure they are being completed. If there is a machine without a job being processed on it and all other tasks are being fulfilled, this empty machine is deemed non-critical. The machine incurs a setup time and the job begins processing. If two or more machines are assigned to process the same job, the algorithm decides whether it is more advantageous to assign the job to the least loaded machine or to another, based on the current load of the machine and the execution time of the individual job. Lastly, if only one machine is used to process a job, the algorithm again checks the advantages to having the new job setup on the current machine or to the least loaded non-critical machine. The main focus of this algorithm is to minimize the maximum completion time among all machines in the system. The competitiveness of the algorithm is defined by the number of task types, machines and the ratio between, setup and

execution times. This model has shown how operations, such as flexible manufacturing systems, can be modeled in competitive manner to the offline schedule.

On-line scheduling is a small topic within on-line algorithms as a whole. The papers listed above, along with many others have provided deep insight into on-line approaches to the proposed scheduling topic. The proposed thesis utilizes a semi on-line procedure with respect to emergency jobs, which will be discussed later within this thesis.

2.6 Knapsack Problems

There is extensive research on knapsack problems related to scheduling. Such research can be applied to this thesis. A portion of the problem at hand deals with the assignment of jobs to workers. Each worker is able to accept so many job units in a given day; hence each worker has a capacity which can not be broken. Assigning a worker to many job units will result in a job not being completed. What this gets down to is the worker acting as a knapsack or container for jobs. Each job takes a specified amount of space within each container. The optimal solution will contain the optimal assignment of jobs to the workers, i.e. the optimal space utilization of each container.

Knapsack problems in scheduling can have various objectives. Most research stems from the assignment of jobs to machines, where machines are the limiting factor in the model. The machines in essence act as the containers to be filled. When jobs are processed on these machines, they can only accept a certain number of jobs, i.e. typical research dictates one job per machine. Croce et al. (2000), worked in a two-machine

environment, with the ultimate goal of minimizing the number of tardy jobs with a common due date. The classification of jobs for this paper is either tardy or non-tardy. The problem in which they are discussing has a classification of NP-hard, thus the author's present a modified branch and bound approach. Through linear relaxation of the problem, the author's determine the upper and lower bounds of the problem, thus minimizing the duality gap of the problem. They note that in several instances they were able to constrain the upper and lower bounds to equal each other, thus reducing the computational time of the problem to mere seconds.

Although Croce et al. (2000) present a somewhat simplistic case, they are able to provide valuable insight into the NP-hard class. It should be noted that the formulation found within this thesis shares common constraints and a similar objective function as seen within this paper. Thus further research must be done into the complexity of the problem, as will be discussed in chapter 3.

A similar paper to the above described was written by M'Hallah and Bulfin (2005). The paper aims to minimize tardy jobs on parallel machines. Again, this paper resembles the atmosphere being presented within this paper, as the workers can be thought of as parallel unrelated machines. As the workers progress through their learning careers, the amount that each worker, i.e. machine, can handle is increased. Again, M'Hallah and Bulfin note that the problems discussed are known to be NP-hard problems. The main model discussed in this paper, which concerns weighted jobs on unrelated machines, can be found within the models presented in this thesis. This thesis however provides models which are much more in-depth towards the real case scenario; hence many more constraints are invoked.

M'Hallah and Bulfin create an exact algorithm to optimally solve a specialized case of the unrelated machine weighted job scenario. By initiating a surrogate constraint, the author's are able to create a special case of a multi choice knapsack problem. The author's implemented their algorithm through a battery of tests, and was able to convey the speed an efficiency of their algorithm. However, the author's were unable to provide explanation for several of their results; hence some questions arise to the validity of their claims that the heuristic is as robust as they claim.

Another paper, more recently published, by M'Hallah and Bulfin (2007) deals with minimizing the weighted number of tardy jobs on a single machine with release dates. They propose a heuristic method and an exact method to improve upon several papers. The author's extensively test both heuristic and algorithm against several papers solutions. They determined that although their heuristic was exceptionally fast, it was unable to produce significantly good results for larger, harder problems. i.e. weighted problems. The problem discussed within this thesis deals also with weighted jobs, i.e. the priority value of each job, which is discussed in more detail in Chapter 3. The authors again note the hardness of the problems to be NP-hard for even the simplest of these problems.

The papers described all deal with similar cases to the thesis at hand. The machines being scheduled can easily be equated to the workers within the given problem. As all of the problems mentioned are NP-hard, and can be found within the formulation of this thesis's problem, we can assume that the given problem is thus NP-hard. The knapsack problems discussed by the above author's provide insight into the classification of this thesis's scheduling problem.

The preceding literature review covered such topics as general learning and how it can be applied to scheduling, various factors affecting one's learning and how to promote learning amongst workers. Learning benefits have been discussed as applied to scheduling and also the inverse of learning, forgetting, has been touched upon. On-line algorithms have been presented as they apply to scheduling. Lastly, knapsack problems have been reviewed due to their similarity to this thesis's main topic. All the material outlined in this literature is applicable to the research presented within this thesis.

Chapter 3: Mathematical Programming Models

The maintenance scheduling problem discussed in this thesis and any other relevant research are provided within the first section of this chapter. The problem is then formulated in two separate and distinct models to accommodate the situations discussed in the following section.

3.1 Background

In chapter 2 it was demonstrated how learning can be applied to scheduling problems. There are many areas for such applications, but this thesis tends towards the manufacturing sector. Lam et al. (2001) discussed how learning a set of skills can be cross-dimensional among many jobs, whether they are repetitive jobs or not. This opens a wide array of possibilities for learning to be applied, and more specifically, provides a basis for this research. Traditional learning curve theory has been aimed towards the benefits associated with repetitive tasks, while this thesis aims its effects towards similar but not exactly repetitive tasks.

The environment this thesis works within is a maintenance shop inside a manufacturing plant. The atmosphere most applicable to this research is where workers are starting fresh, such as apprentices, within the maintenance shop. The apprentice will have no previous experience, thus as the worker progresses throughout their career their processing abilities will increase. Of course there will be permanent workers who have

prior experience in the system as well, as no apprentice is without a teacher. Hence there will be a mixture of workers throughout the system.

In the system, workers are assigned specific jobs. A job is the repairing of a die or mold, which falls in one of the two main categories: (i) Preventative maintenance, and (ii) Emergency. Each job has a specific set of tasks that needs to be done to it, but the tasks are similar among all of the different jobs. Thus, traditional learning curve theory is applicable as an operator is now learning a specific set of skills to be applied, rather than learning a specific set of procedures to follow to complete a repetitive task.

Each set of jobs has various details that are available before the schedule is made. The preventative maintenance job information includes the due date, approximate processing time, and task importance (i.e. priority level). It is assumed that at the beginning of the scheduling period all of the jobs are available (i.e. all jobs arrive at time zero). These jobs are known to be going in for maintenance, hence, are removed from the system and brought into the maintenance area. The emergency jobs however provide a different set of problems. One major problem being the entry of the tasks into the system is unknown. An emergency job typically occurs when there is an in-process problem, a mold or die is damaged and needs immediate attention. When an emergency job enters the system the current job schedule needs to be reformulated, which is discussed in section 3.3.

The problem extends however to the extent of not only job scheduling but also operator scheduling. There are a pool of operators that need to be scheduled to specific days and shifts. Once this schedule is set, it can not change, even if an emergency job enters the system. If an emergency job does enter the system, then the only schedule that

should change is the job schedule and task assignment, which is discussed in section 3.8. Like tasks, operators also carry a set of information, which includes previous experience (i.e. cumulative days of work), wage level, etc. From this information, it can be determined how many tasks the operator can handle in a given shift. Given the knowledge of worker learning and experience, it can be assumed that over time an operator can handle more or larger tasks. The scheduling programs developed in this thesis take all of these factors into consideration. The following diagram, Figure 2, shows a typical path for a single job entering the system. Note, a partial job is considered as a job that has not been fully processed. Emergency jobs follow the same path, and will be discussed further in section 3.3.

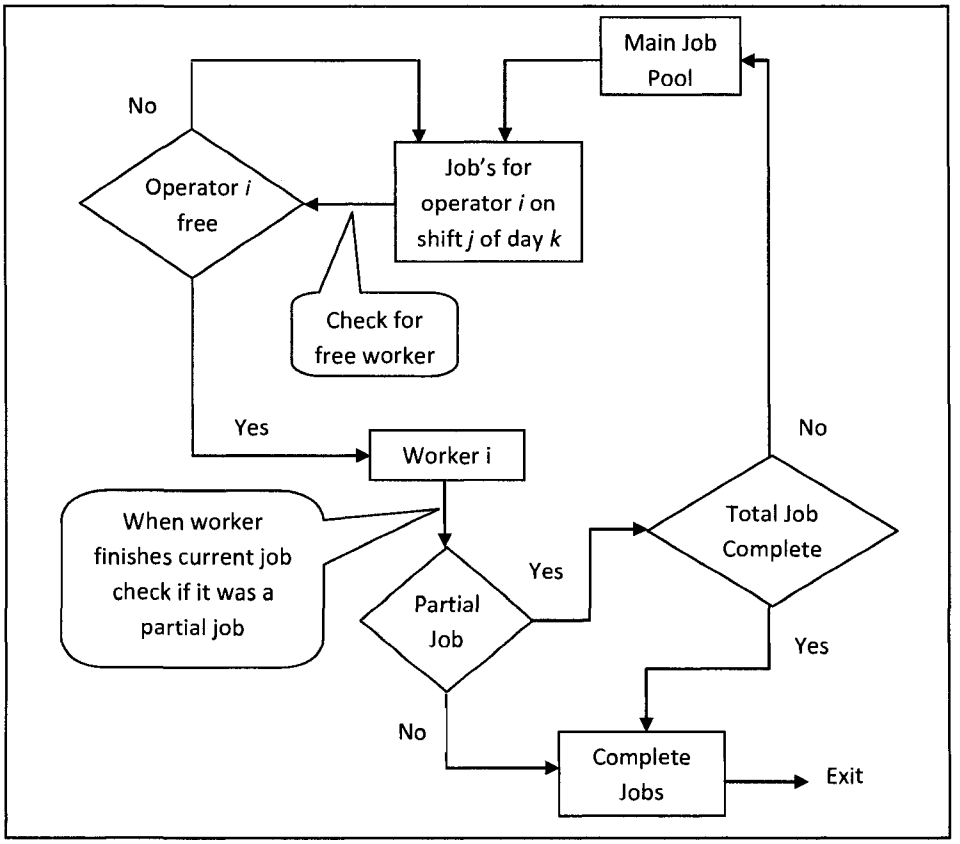


Figure 2: Typical Job flow through system

3.2 Preventative Maintenance Model

This model, as described in the previous section, deals with the preventative maintenance jobs encountered within the context of the problem. The following section is broken into subsections, containing information pertinent to the model. As was discussed in section 3.1, this model provides the appropriate schedule for operators and task assignment.

3.2.1 Index Descriptions

i : Denotes an operator number. Within the program, each operator is assigned a number to keep track of who is and is not working (i.e. worker 1, worker 2, etc.)

j: Denotes a particular shift that either a job or operator is scheduled. The program allows for multiple shifts to operate in a single day, hence this index determines the shift ID (i.e. shift 1, shift 2 etc.)

k: Denotes the day that either the worker or job is to be scheduled. As this program allows for multiple days of processing, k will be used to indicate the particular day of specific tasks or the days the operator will be brought in (i.e. Day 1, Day 2, etc.)

r: The program handles many different jobs, which is denoted by the index r (i.e. Job1, Job 2, etc.)

q: This index is related to wage incentives built into the program. The program is capable of handling many different cost benefits related to time and operator experience (i.e. Incentive 1, Incentive 2, etc.). Each progressive step yields larger wages for the operators, as is seen in real world cases

3.2.2 Variable Descriptions

3.2.2.1 Binary Variables:

x_{ijk}: This binary variable depicts the worker's availability. It reads as follows: If worker i is to be scheduled on shift j of day k, then the value assigned to this variable is one, otherwise the value is equal to zero. This variable is highly instrumental to this program as it will determine how many jobs can be processed within certain shifts, depending on which operators are scheduled, and what the operator's experience is at that given time.

Q_{ijkr} : This variable works in conjunction with the previous variable and reads as follows: If job r is assigned to operator i on shift j of day k, then this variable equals one, otherwise it equals zero. This variable only has a chance to be initiated if the operator in questions is assigned to that specific shift, otherwise it will automatically equal zero. This ensures that no job is assigned to an operator who is not present on a given shift. More discussion of this property is in the constraint section.

$y_{q\,i}$: This binary variable is used as an if-else operator. It is seen within constraints 13 and 14 and is used in conjunction with cost incentives, which are periodically awarded to operators with an accumulated amount of experience (time spent with company). More discussion of this variable's application will be brought up in section 3.7.

$PQ_{k\,r}$: The purpose of this variable is to determine if a job r has been processed on day k or not. If this job has been broken into different sections, then it is possible to work on the same job over multiple shifts. If this certain job r is late, then the job could be penalized more than once, hence this variable groups the multiple shifts into one, so as the job is only penalized once.

3.2.2.2 General Integer Variables:

$T_{i\,k}$: This is the cumulative experience in days of worker i at the current day k. The variable will be discussed more in detail in section 3.7

3.2.2.3 General Variables:

$PC_{j\,k\,r}$: This denotes the partial completion of job r processing time on shift j day k. Although not all jobs will be divided into smaller sub sets of a single job, this variable can still hold true for a full job to be completed. It will simply equal the full value of the processing time.

$CI_{q\,i\,k}$: The cost incentive variable works in conjunction with $y_{q\,i}$ discussed previously. Throughout the program wage incentives are offered to the operators and are awarded after various periods of time have passed. The variable is read as follows: wage incentive q for worker i available on day k.

S_r : Denotes the number of days of delay that a job r experiences past its due date.

$W_{i\,j\,k}$: This is a key variable, as it relates to the amount of learning that an operator has experienced. The variable works by systematically increasing the operator's workload over time, hence the operator is able to complete more tasks the longer he/she stays with the company. It is read as follows: Operator i working on shift j of day k is capable of handling a workload of "X" units, where "X" is the value of the variable. More discussion of this variable is presented in section 3.7.

3.2.3 Parameters:

C_i : Operator i has a base wage for working one shift. This parameter denotes the specific base wage of each worker, which may vary from operator to operator. Note that the base wage of a worker never changes; only the incentives for that worker may be increased over time above and beyond the base wage.

PRr_r : Each job is assigned a priority level depending on the value of the specific job. This parameter varies between one (lowest priority) and ten (highest priority). Note that

all emergency jobs would be denoted with a level ten priority as the job needs to be processed immediately as a delay will result in a large penalty.

PC : This is a scalar value for the penalty of job delays, as this promotes more emphasis towards the removal of job delay for the high cost associated with it. This value is constant for every job.

α : A simple parameter dictating how many workers are allowed per shift. This parameter is needed as different facilities have varying resource capabilities, i.e., limited workstations etc.

γ : This parameter relates to a worker with no learning experience, as it would be the total amount of work units to be processed in a single shift if no learning was present.

l_i : Each operator i is assigned a learning index for a specific learning curve. To obtain these values, management would need to assess the workforce, through various testing. Within the context of this thesis, theoretical values are assigned.

δ : A scalar denoting the maximum number of jobs that should be assigned a single worker within a given shift. The worker should not be assigned more jobs, hence adding more balance to the overall schedule. Instead of being assigned smaller tasks, over time the operator will be assigned larger tasks.

P_r : Each job r is assigned an approximate processing time. Not every job is the same, so processing times will vary.

D_r : Each job r is assigned a due date. The due date corresponds to the day only, not a specific shift. The reason being is that each day is the beginning of a new production run and it is assumed that the dies or molds are required for the beginning shift.

TI_q : Each wage incentive corresponds to a specific time period. If the operator works a certain amount of cumulative days, they will receive certain benefits, mainly pay raises. This parameter denotes the cumulative days required to initialize wage incentive q.

β: The value of the actual wage increase, seen in constraint 13.

3.2.4 Objective Function

$$MIN\ Z = \sum_{i}\sum_{jJ}\sum_{k}(C_i * x_{i\,j\,k}) + \sum_{i}\sum_{j}\sum_{k}\sum_{q}(CI_{q\,i\,k} * x_{i\,j\,k}) + \sum_{k}\sum_{r}(PQ_{k\,r} * PRr_r * PC * S_r)$$

The objective function is divided into two main parts: (i) base wage and wage incentives of operator, and (ii) job delay penalty. As all of these relate to cost, the aim is to minimize. First we aim to minimize the total workforce, which is accomplished by minimizing the total wages paid out through the scheduling period. Each time an operator is brought in, a cost of C_i is incurred and over time many cost incentives, $CI_{q\,i\,k}$, amass.

Hence the function aims to minimize the number of operators to bring in and also to vary the operators so as not to accrue too many cost incentives for a single worker as the objective is to minimize cost. The second half aims to minimize job delays. If a job is processed past its due date then certain penalties will apply. The binary variable PQ_{kr}, discussed in section 3.4.1, works in conjunction with S_r, and determines if the job is processed late or not. PQ_{kr} first determines if job r has been processed in at least one shift on day k. (i.e. yes if $PQ_{kr} = 1$, no if $PQ_{kr} = 0$). If it has, then the function looks to see if that is past the due date ($S_r > 0$). If job r has been processed past its respective due date, then the corresponding penalty will be assumed. A single job may be assigned more than one penalty depending on how many days it has been processed past its respective due date (i.e., if job r is two days past due, then the penalty will be doubled, etc.)

3.2.5 Constraints

$$\sum_{j} x_{i\,j\,k} \leq 1 \qquad \forall\, i,k \tag{1}$$

The purpose of constraint one is to simply limit the number of shifts that an operator may obtain. It is decided that a worker may only work a maximum of one shift per day; hence the summation of all shifts present per day must be no more than one. The inequality is required because an operator may not be scheduled for any shifts on day k.

$$\sum_{i} x_{i\,j\,k} \leq \propto \qquad \forall\, j,k \tag{2}$$

As was mentioned in section 3.5, α is introduced to limit the operator resources for any single shift j. To reiterate, a manufacturing facility may only have the capabilities

to schedule so many operators, as space or workstations may be limited. Constraint two also dictates a more balanced work schedule, as the number of operators will be divided more evenly across all shifts.

$$T_{i\,k} = \sum_{j} \sum_{m=1}^{k} x_{i\,j\,m} \qquad \forall\, i,k \tag{3}$$

This is a dynamic constraint as the value of day k is constantly increasing. With each passing day k, an operator has the opportunity to gain more experience if they are scheduled to a given shift. This constraint acts as a counter, determining how many days prior to the current day k an operator has accumulated. The variable $T_{i\,k}$ is then passed into constraint four to determine an appropriate workload to be assigned to operator i.

$$W_{i\,j\,k} = \gamma * T_{i\,k}^{\wedge l_i} * x_{i\,j\,k} \qquad \forall\, i,j,k \tag{4}$$

This is one of the crucial constraints of the program as it dictates how many task units can be assigned to an operator for a given shift j on day k. This constraint ties in the previous constraint value of T_{ik}, and utilizes the operator's learning abilities to determine the exact workload that should be assigned to the operator for a given day k. To allow this constraint to work, the binary variable $x_{i\,j\,k}$ must be activated to ensure that the operator is actually brought into the shift, otherwise zero job units will be assigned to the operator for that particular shift. This key constraint is what gradually increases the workload of the operator.

$$\sum_{r} Q_{ijkr} \leq \delta * x_{ijk} \qquad \forall i,j,k \tag{5}$$

Constraint five depicts the maximum number of jobs that should be assigned to an operator on any given shift. This constraint acts as a safeguard so that one operator is not being overloaded, while other operators are not being assigned enough jobs.

$$\sum_{r} Q_{ijkr} * PC_{jkr} \leq W_{ijk} \qquad \forall i,j,k \tag{6}$$

Constraint six acts as a check, ensuring that the job units assigned to operator *i* do not exceed the maximum job units that can be handled by that operator. Hence the summation of all of the jobs *r* assigned to worker *i* on shift *j* of day *k* must be less than or equal to the total amount of units worker *i* on shift *j* of day *k* can accept.

$$\sum_{i} Q_{ijkr} \leq 1 \qquad \forall j,k,r \tag{7}$$

A single job can only be processed by one worker on any given shift. Many instances will dictate more than a single operator for a given shift *j*, therefore to ensure job *r* is only processed by one operator *i*, the sum of all operators *i* for every shift *j* on day *k* for every job *r* must be strictly less than or equal to one. The inequality serves for the option of choosing to process job *r* in that particular time slot or not.

$$\sum_{j}\sum_{k} PC_{jkr} = P_r \qquad \forall r \tag{8}$$

Constraint eight is designed as a check to ensure all partial jobs *r* equal the total job processing units for the complete job *r*.

$$\sum_i \sum_j \sum_k \frac{Q_{ijkr} * PC_{jkr}}{P_r} = 1 \qquad \forall r \tag{9}$$

To ensure that jobs are going to be assigned, constraint nine must be introduced. Without this constraint the program simply sets Q_{ijkr} to zero, therefore no workers or job penalties will be assigned and the objective function will equal zero. The aim of this constraint is that the partial jobs are fractions of the total job. Since the goal is to completely finish the job, the fractions of each partial job must equal the whole job. Although similar to constraint eight, constraint nine serves a different purpose, which is to ensure that the total job is complete, whereas constraint eight simply dictates that the summation of the partial jobs must equal the total job units assigned to that job.

$$Q_{ijkr} \leq PC_{jkr} \qquad \forall i,j,k,r \tag{10}$$

To ensure that no jobs are assigned with zero hours, this constraint is introduced. If the variable PC_{jkr} is equal to zero, then it is guaranteed that there will be no assignment as the binary variable Q_{ijkr} dictates. There can only be a variable assignment when PC_{jkr} is greater than zero.

$$\sum_i \sum_j Q_{ijkr} \leq J * PQ_{kr} \qquad \forall k,r \tag{11}$$

To ensure that only a single penalty is incurred for job r being late by one day and being processed on multiple shifts, a simple function must be met. First, calculate how many shifts job r has been processed on by summing every operator i and shift j for every day k. If this value is zero, then no PQ_{kr} will equal zero as the goal of the program is to

limit the total penalty. If there is at least one assignment within a given shift j on day k then the binary value of PQ_{kr} will be initiated and then multiplied by the total number of shifts J. As the jobs can not possibly be processed more than the total shifts of one day, this constraint will never be violated.

$$PQ_{k\,r} * k \leq D_r + S_r \qquad \forall\, k, r \tag{12}$$

Constraint twelve is where penalties are calculated. Each day k is checked to see if job r has been processed or not. If it has been processed, then the program checks job r completion time with respect to its due date. If it is completed before the due date, then no penalty will be incurred and S_r will simply equal zero. If however, job r is completed past its respective due date, then S_r will increase, and a respective penalty is incurred within the objective function.

$$CI_{q\,i\,k} = \beta * y_{qi} \qquad \forall\, q, i \tag{13}$$

$$TI_q \geq T_{i\,k} * (1 - y_{qi}) \qquad \forall\, q, i, k \tag{14}$$

The final two constraints, 13 and 14, work in unison with one another. These two constraints dictate the wage incentives for each of the operators. Constraint 13 increases from zero to an arbitrary value (here shown as 0.5) depending on the value of y_{qi}, which comes from constraint 14. Constraint 14 makes use of an inequality to determine when the binary variable, y_{qi}, initiates. Each day k brings the opportunity for a wage increase, and once the value of T_{ik} surpasses the value TI_q required to achieve wage incentive q, then y_{qi} must initiate itself from zero to one, thus developing the wage incentive for operator i after day k.

3.2.6 Model Assumptions

A few assumptions for the model must be acknowledged. A main assumption regards jobs which have been divided into multiple jobs. There must be some method of tracking what has been completed for a specific job. Take for example; a job has been broken into two separate jobs, i.e. job 1.1 and 1.2. Assuming job 1.1 consists of cleaning and job 1.2 consists of the repair, it would not make sense to have job 1.2 be processed before 1.1, as we must clean the job before we can see what is to be repaired. Thus it is imperative that job 1.1 be processed before job 1.2. It will be assumed that the model makes this decision within the given subsets of jobs, hence when a job has been processed twice; it is assumed that it will be done in sequential order. In the simplified models, a simple inequality constraint can be invoked to handle this case which reads as: Job 1.2 – Job 1.1 <= 0. In this case it states that job 1.2 can not be invoked before job 1.1. Another rule must state that Jobs 1.1 and 1.2 cannot be processed on the same shift, hence additional work constraints need to be invoked as follows:

$$\sum_{i} \left(Q_{ijk1.1} + Q_{ijk1.2}\right) \leq 1 \; \forall \, j, k, 1.1, 1.2$$

The first additional constraint dictates that job 1.2 can not be processed before job 1.1; the second dictates that jobs 1.1 and 1.2 can not be processed on the same shift. With these two additional constraints, the model should be able to handle any size job.

3.2.7 Complete model

$$MIN\ Z = \sum_i \sum_j \sum_k (C_i * x_{ijk}) + \sum_i \sum_j \sum_k \sum_q (CI_{qik} * x_{ijk}) + \sum_k \sum_r (PQ_{kr} * PRr_r * PC * S_r)$$

s.t.

$$\sum_j x_{ijk} \leq 1 \qquad \forall i,k \tag{1}$$

$$\sum_i x_{ijk} \leq \propto \qquad \forall j,k \tag{2}$$

$$T_{ik} = \sum_j \sum_{m=1}^{k} x_{ijm} \qquad \forall i,k \tag{3}$$

$$W_{ijk} = \gamma * T_{ik}^{\wedge l_i} * x_{ijk} \qquad \forall i,j,k \tag{4}$$

$$\sum_r Q_{ijkr} \leq \delta * x_{ijk} \qquad \forall i,j,k \tag{5}$$

$$\sum_r Q_{ijkr} * PC_{jkr} \leq W_{ijk} \qquad \forall i,j,k \tag{6}$$

$$\sum_i Q_{ijkr} \leq 1 \qquad \forall j,k,r \tag{7}$$

$$\sum_j \sum_k PC_{jkr} = P_r \qquad \forall r \tag{8}$$

$$\sum_i \sum_j \sum_k \frac{Q_{ijkr} * PC_{jkr}}{P_r} = 1 \qquad \forall r \tag{9}$$

$$Q_{ijkr} \leq PC_{jkr} \qquad \forall i,j,k,r \tag{10}$$

$$\sum_i \sum_j Q_{ijkr} \leq J * PQ_{kr} \qquad \forall k,r \tag{11}$$

$$PQ_{kr} * k \leq D_r + S_r \qquad \forall k,r \tag{12}$$

$$CI_{qik} = \beta * y_{qi} \qquad \forall q,i \tag{13}$$

$$TI_q \geq T_{ik} * (1 - y_{qi}) \qquad \forall q,i,k \tag{14}$$

$x_{ijk}, Q_{ijkr}, y_{qi}, PQ_{kr} = 0,1$
$PC_{jkr}, T_{ik} \in Z^+$
$CI_{qik}, S_r, W_{ijk}, \quad \geq 0$

3.3 Semi On-line Off-line model

3.3.1 Model description

The problem dictates that some emergency jobs will enter the system at any given point. All of the previous jobs that have been scheduled were known ahead of time with job details containing processing time, release time, due dates etc. The problem now is that an emergency job may enter the system and need immediate processing, which may inadvertently cause disruption to the existing schedule. To include this new job into the schedule a few alterations need to be made to the existing formulation.

3.3.2 Model adjustments

The schedule for the operators has already been determined and will henceforth be the schedule that will be followed. The current job schedule however will need to be recalculated. All of the information determined from the first schedule with respect to the operators will now become parameter values, which include the variables x_{ijk}, T_{ik}, and W_{ijk}. Since these are now constant, constraints 1, 2, 3, and 4 are eliminated from the new formulation. Also, since the operators schedule is fixed, the costs for these operators are fixed, hence reducing the objective function's complexity and eliminating constraints 13 and 14.

The new formulation alters a few parameters and variables and also alters a few constraints, which will now be discussed. The objective function is as follows:

$$MIN\ Z = \sum_i \sum_j \sum_k \sum_r (Q_{i\,j\,k\,r} * PRr_r * PC * S_j)$$

It shares similar characteristics as its predecessor with one small change. Instead of looking solely at days, it must now be brought down to the level of a single shift. To accomplish this, the variable $PQ_{k\,r}$ is replaced with $Q_{i\,j\,k\,r}$ as we are now concerned with every shift instead of only the day. The costs for the operators, as mentioned above, have been negated due to their constant nature in this new formulation.

Constraints 5, 6, 7, 8, 9, and 10 remain unchanged and are now referred to as constraints 1, 2, 3, 4, 5, and 6 respectively. The previous formulation required a variable $PQ_{k\,r}$, which was determined in constraint 11. The new formulation does not require this variable as it dealt with consolidating all of the shifts a single job was processed on into a binary value, determining if it was or was not processed on day k regardless of how many shifts it was processed on. The new formulation looks at each individual shift a job is due by, henceforth, $PQ_{k\,r}$ is no longer required, eliminating constraint 11 from the new formulation.

The last constraint to be dealt with from the previous formulation is 12, which will now be referred to as constraint 7 as is depicted within the new formulation. This deals with the lateness of job r being processed. In the previous formulation, shifts were not an issue, but due to the extreme nature of the emergency job, time is of the essence. To accommodate this urgency, due dates are now referenced with shifts rather than days (i.e. a job with a due date of day 1 will now have a due date of shift 3, due date = day 2 translates to shift 6, etc.). The corresponding shift for the new emergency job will be the shift it arrives on. To simplify the formulation it is assumed that the emergency job arrives at the beginning of the shift in question (i.e. a job breaks down and can not be

removed from process until the end of shift j-1, hence arriving on shift j). Since due dates are dealing with shifts, it is natural to change the lateness variable S_r to S_j, which is now also dealing on a shift basis. The left hand side of constraint 7 has changed to accommodate shift level lateness. As was discussed with the objective function, constraint 7 also contains the binary variable Q_{ijkr}. This again takes the place of variable PQ_{kr}, and denotes if a job has been processed on shift j of day k. A key difference to be noted is the scalar $(j + (k\text{-}1) * J)$ found in the new formulation, rather than k found in the prior formulation. Each shift that jobs are processed must be accounted for with respect to the new due date system mentioned above. The value of J denotes the total number of shifts j that is found in a single day k, which is user defined. The function of $(k - 1) * J$ is to increase shifts sequentially for each day as the due dates have been changed from days to shifts (i.e. assuming 3 shifts / day, on day 1 shift 3, $(j + (k - 1) * J) = (3 + (1 - 1) * 3) = 3$; day 2 shift 2 = (2 + (2-1)*3) = 5, etc.)

3.3.3 Semi On-line Off-line procedure

A few simple steps need to be addressed before the new formulation can be run and are as follows:

1. Check day k' and shift j' that job $r + 1$ enters system. Eliminate all jobs which have had processing on day $k < k'$ and $j < j'$.
2. Check all jobs for partial completion. If job r has been semi-processed on previous shift(s) j of day(s) k,

$$P_r' = P_r - \sum_{j}^{j<j'} \sum_{k}^{k<k'} PC_{j\,k\,r}$$

Update $P_r = P_r$'

3. Set new $j = j'$ and $k = k'$
4. Set parameters $x_{i\,j\,k}$ and $W_{i\,j\,k}$ determined from previous formulation
5. Change values for due dates from days k to shift j of day k.

   $D_j = k$ * (# of shifts j)
6. Set emergency job D_j = current j, PRr_{r+1} = Max Priority Level

Once these steps are followed, the new formulation should be run to achieve the new schedule of jobs to be assigned to the operators who are currently scheduled. The new formulation follows in section 3.3.4. The variables and parameters have not changed in description and can be found in sections 3.2.2 and 3.2.3. Constraints 1 through 6 have not changed, and are as described in section 3.2.5.

3.3.4. Complete Semi On-line Off-line Mathematical Model

$$MIN\ Z = \sum_i \sum_j \sum_k \sum_r \left(Q_{ijkr} * PRr_r * Pc * S_j\right)$$

s.t.

$$\sum_r Q_{ijkr} \leq \delta * x_{ijk} \qquad \forall\, i,j,k \tag{1}$$

$$\sum_r Q_{ijkr} * PC_{jkr} \leq W_{ijk} \qquad \forall\, i,j,k \tag{2}$$

$$\sum_i Q_{ijkr} \leq 1 \qquad \forall\, i,k,r \tag{3}$$

$$\sum_j \sum_k PC_{jkr} = P_r \qquad \forall\, r \tag{4}$$

$$\sum_i \sum_j \sum_k \frac{Q_{ijkr} * PC_{jkr}}{P_r} = 1 \qquad \forall\, r \tag{5}$$

$$Q_{ijkr} \leq PC_{jkr} \qquad \forall\, i,j,k,r \tag{6}$$

$$Q_{ijkr} * (j + (k-1) * J) \leq D_j + S_j \qquad \forall i,j,k,r \tag{7}$$

$Q_{ijkr} = 0,1$
$PC_{jkr} \in Z^+$
$S_j \geq 0$

Chapter 4: Heuristic Approach

The following chapter presents a heuristic method to approximate the solution obtained by the optimization models. The heuristic was programmed and tested in MATLAB 7.0. This chapter is broken up into the following sections: Introduction, Heuristic Procedure, Heuristic Example, and a brief Discussion.

4.1 Introduction

A common practice to solve hard problems is the application of a heuristic. It is most common within online scheduling. The proposed heuristic shares characteristics of the Genetic Algorithm, where parents and children are developed constantly by searching through the priorities of each job. As the priority of the job dictates its importance, the heuristic strives to achieve the largest priority of all pairings of jobs, while still checking the feasibility of the pairing with respect to processing time. The main focus places much onus on the priority of the job, but other steps are integral to the creation of the best possible schedule. Because priority plays such an integral step in the selection process, as will be discussed in Chapter 5, the heuristic can also be equated to a greedy heuristic, in that it tries to determine the maximum possible priority combination amongst all jobs. A flow chart can be found in Appendix IV which outlines the full heuristic. In the following sections, the heuristic procedure will be presented, along with a detailed explanation of the procedure in whole. A small demonstration will also be presented.

4.2 Heuristic Procedure

Step 0: Data Initialization

Initialize values for priority (PRr), due date (Dr), and processing time (Pr) for each job n (i.e. PRr{J_1, J_2, J_3,..., J_n}, Dr{J_1, J_2, J_3,..., J_n}, and Pr{J_1, J_2, J_3,..., J_n}, where J_j is the job in position j).

Set: x(i) = PRr(i) = Pr(i) = Dr(i) = JN(i) = y(i) = 0, where i ∈ {1,2,3,4,5,6}

PENALTY = WC = 0, MaxWorkers = Total workers

Step 1: Divide the jobs according to their due dates

Step 1.0: Set j = 1;

Step 1.1: If Dr{J_j} > 5, proceed to step 1.2, else proceed to step 1.3;

Step 1.2: y(6) = y(6) + 1;

Dr(6){$J_{y(6)}$} = Dr{J_j}; PRr(6){$J_{y(6)}$} = PRr{J_j}; Pr(6){$J_{y(6)}$} = Pr{J_j}; JN(6){$J_{y(6)}$} = J_j;

Check if j = n; if yes, proceed to step 2, else, j = j + 1, return j to step 1.1 and repeat steps

Step 1.3: Set i = Dr{J_j};

y(i) = y(i) + 1; where j ∈ {1,2,3,4,5}

Dr(i){$J_{y(i)}$} = Dr{J_j}; PRr(i){$J_{y(i)}$} = PRr{J_j}; Pr(i){$J_{y(i)}$} = Pr{J_j}; JN(i){$J_{y(i)}$} = J_j;

Check if j = n; if yes, proceed to step 2, else, j = j + 1, return j to step 1.1 and repeat steps

Step 2: Create comprehensive due date and job number sets

Step 2.0: Check if any Dr(i) = 0 (i.e. nothing due on day j). If yes, delete, else, proceed to step 2.1.

Check if any JN(i) = 0 (i.e. nothing due on day j). If yes, delete, else, proceed to step 2.1.

Step 2.1: For all Remaining values of Dr(i) and JN(i) create

Dr{Dr(1,), …, Dr(5)}, JN{JN(1), …, JN(5)}

*Note, Dr(6) and JN(6) are omitted as they will be processed in the following week.

Step 3: Initialize looping values for priority Search

Initialize: q = 1, r = 1

Step 4: Initialize Looping Values

Set: Pr = {Pr(1),…, Pr(q)}; *Note: if q = 1 ➔ Pr = {Pr(1)}

PRr = {PRr(1),…, PRr(q)}; *Note: if q = 1 ➔ PRr = {PRr(1)}

Step 5: Check value of x

Set: x = max(PRr);

If x > 0, Proceed to step 6, else, proceed to step 8.

Step 6: Priority Search

Step 6.1: Check processing time and priority of single jobs

If $Pr(J_j)$ <= Workers Processing Ability, and $PRr(J_j) > x$

Set $x(1) = j$, $x = PRr(J_j)$;

if all jobs in Pr have been checked, proceed to step 6.2;

Step 6.2: Check processing time and priority of paired jobs

If $Pr(J_j) + Pr(J_{j'})$ <= Workers Processing Ability, and,

$PRr(J_j) + PRr(J_{j'}) > x$, Set $x(1) = j$, $x(2) = j'$, and $x = PRr(J_j) + PRr(J_{j'})$;

if all jobs in Pr have been checked, proceed to step 6.3;

*Note: Job $J_{j'}$ is always > job J_j

Step 6.3: Three Jobs

If $Pr(J_j) + Pr(J_{j'}) + Pr(J_{j''})$ <= Workers Processing Ability, and,

$PRr(J_j) + PRr(J_{j'}) + PRr(J_{j''}) > x$, Set $x(1) = j$, $x(2) = j'$, $x(3) = j''$and

$x = PRr(J_j) + PRr(J_{j'}) + PRr(J_{j''})$; *Note: $J_{j''} > J_{j'} > J_j$

if all jobs in Pr have been checked, proceed to step 6.4;

Step 6.4: Four Jobs

If $Pr(J_j) + Pr(J_{j'}) + Pr(J_{j''}) + Pr(J_{j'''})$ <= Workers Processing Ability, and,

$PRr(J_j) + PRr(J_{j'}) + PRr(J_{j''}) + PRr(J_{j'''}) > x$, Set $x(1) = j$, $x(2) = j'$, $x(3) =$

j'', $x(4) = j'''$ and $x = PRr(J_j) + PRr(J_{j'}) + PRr(J_{j''}) + PRr(J_{j'''})$;

if all jobs in Pr have been checked, proceed to step 6.5;

*Note $J_{j'''} > J_{j''} > J_{j'} > J_j$

Step 6.5: Five Jobs

If $Pr(J_j) + Pr(J_{j'}) + Pr(J_{j''}) + Pr(J_{j'''}) + Pr(J_{j''''}) <=$ Workers Processing Ability, and, $PRr(J_j) + PRr(J_{j'}) + PRr(J_{j''}) + PRr(J_{j'''}) + PRr(J_{j''''}) > x$,

Set $x(1) = j$, $x(2) = j'$, $x(3) = j''$, $x(4) = j'''$, $x(5) = j''''$, and

$x = PRr(J_j) + PRr(J_{j'}) + PRr(J_{j''}) + PRr(J_{j'''}) + PRr(J_{j''''})$;

*Note: $J_{j''''} > J_{j'''} > J_{j''} > J_{j'} > J_j$

if all jobs in Pr have been checked, proceed to step 7;

Step 7: Calculate Penalty

Step 7.0: Check values of x(i), for any $x(i) > 0$ then proceed to step 7.1

Step 7.1: If day $q > Dr(J_{x(i)})$ proceed to step 7.2,

else PENALTY = PENALTY;

if all jobs of value $J_{x(i)}$ have been looked at, proceed to step 8

Step 7.2: $PENALTY = PENALTY + (q-Dr(J_{x(i)})*PRr(J_{x(i)})*100)$;

if all jobs of value $J_{x(i)}$ have been looked at, proceed to step 8

Step 8: Set values for Pr and PRr for next iteration

Step 8.0: Assign all jobs $J_{x(i)}$ where $x(i) > 0$ to worker *r*, which are to be completed on day *q*; Set: $Pr(J_{x(i)}) = M$; $PRr(J_{x(i)}) = 0$;

Step 8.1: check day *q*

Break up Pr and PRr according to their due dates

$Pr = \{Pr(1), \ldots, Pr(q)\}$, $PRr = \{PRr(1), \ldots, PRr(q)\}$

Step 8.2: If r = MaxWorkers,

q = q + 1; r = 1

return to step 4 and repeat until either all jobs are scheduled

or q = 5, r = MaxWorkers

else, q = q, r = r + 1;

return to step 4 and repeat until either all jobs are scheduled

or q = 5, r = MaxWorkers

END

4.3 Heuristic Explanation

To elaborate on the proposed heuristic, a detailed explanation will now be presented. In step 1 we must create sub groups for each day's activities. Therefore the heuristic works by first sorting the jobs according to their respective due dates, hence initializing the sets PRr(i) and Pr(i) where $i \in \{1,2,3,4,5,6\}$. This also gives rise to the sets Dr(i) and JN(i) which indicate the due dates of the respective jobs and a marker number for each job so we know which jobs are being processed. Because we do not want any zero values in our sets (because there should be no jobs that take zero hours), step 2 eliminates any possibility of creating this case. Step 3 is where we initialize our looping function where q indicates which day processing is taking place and r indicates which worker we are currently interested in. In step 4 we initialize the values of jobs to be processed. We are only interested in jobs that are past due and due on current day q. In step 5 we check for the current largest value of x, which unless all jobs have been

assigned will have a value greater than zero. If however all of the jobs have been assigned, the heuristic will jump directly to step 8, which will be discussed shortly.

Step 6 is where we can see the similarities between the heuristic and the genetic algorithm. At first, we are only interested in viewing the jobs which have large priority values and still fall under the processing abilities for a given worker. After finding these values we move to the next iteration which takes a combination of two jobs and checks again for its feasibility (i.e. if the processing time can be completed for both jobs, and if the value of the priority is greater than that of the previous value of x). If this is a feasible case, the values are updated. We do this for three, four, and five jobs. Five jobs seems to be the limit with the given set parameters for the optimization models, so the heuristic only allows for a maximum of five jobs to be assigned to any one worker. The similarity to the genetic algorithm is that a parent i.e. current optimal case, maybe succeeded by one of its children i.e. new optimal case. Take for instance two jobs {J1, J2}, with processing times {3,2} and priorities {7,8}. If the current optimal case was just job J2 from the first iteration, in the second iteration the parent J2 will spawn a new child which is jobs J2 and J1. Now we have the new pairing (J1,J2) with the processing time of 5 and priority of 15, and assuming that the processing time is feasible, we have a better pairing than just the single job J2 alone.

Once the jobs have been selected to be processed, we must check the amount of penalty to be incurred, if any. In step 7 we do just that, all of the jobs that have been selected are referenced against the current day q with respect to its due date $Dr(J_i)$. If the job is completed before or on the respective due date, then no penalty shall be incurred.

However, if the completion date is past due, then a respective penalty is incurred for each successive day late.

In step 8, the heuristic must establish which jobs have been processed and which jobs are remaining. Therefore, some precautionary steps are taken, which include setting the priorities and due dates of completed jobs to "0" and "M" respectively, where "M" is an arbitrarily large number. Next, the algorithm separates the joint Pr and PRr sets back into their original Pr(i) and PRr(i) sets. This is done in case the value of q is increased (day q ➔ day (q +1)). Next we check and see if all of the workers have been utilized. If not, the next worker is checked and run through the cycle. If all of the workers have been utilized (or at least checked) for day q, then day q changes from q ➔ q+1, and the workers are reset back to 1. In step 5, we can either go to step 6 or 8 (both previously described). If we proceed directly to step 8, then it indicates that all jobs have been scheduled and we no longer will need to schedule workers for that day. The next day we will continue to schedule workers will be the next day a job is due. This heuristic main processing time takes place within step 6. For an absolute worst case the heuristic will need to perform $q \cdot r \cdot [n + (n-1)! \cdot [n \cdot ((n-4)^2 + 6) - 19]]$ checks in step 6. It will not always be this case as not all of the workers will be utilized. The checks are simple and require very little processing time to compute, as it only checks the feasibility of the combination. If the combination proves feasible, the original data, pending the new combination is better, is replaced by the new combination. To demonstrate the heuristics processing methodologies, a small demonstration will now be provided.

4.4 Heuristic Example Problem

This small problem will help to see how the heuristic works. The problem data used in the example is found in the following table.

Jobs	Priority	Processing Time	Due date
Job 1	10	7	1
Job 2	3	4	1
Job 3	8	3	1
Job 4	4	2	1
Job 5	1	1	1
Job 6	6	3	1
Job 7	2	2	1
Job 8	4	4	1
Job 9	7	3	1
Job 10	6	4	1

Table 1: Heuristic Example Data – Jobs

For ease we have stated that all jobs are due on day 1, so as to limit some of the possibilities. We also have the following worker data available.

	Day 1	Day 2	Day 3	Day 4
Worker 1	8	8	8	8
Worker 2	8	8	8	8
Worker 3	8	8	8	8

Table 2: Heuristic Example Data - Workers

This table indicates for each day the abilities of that worker. That data does not provide learning within the context of the problem, as it will be shown later, learning plays more of a factor when there is a mixture of workers within a system. To begin the heuristic, the sets Pr, PRr, Dr, and JN are created as follows:

Pr = {7, 4, 3, 2, 1, 3, 2, 4, 3, 4}; PRr = {10, 3, 8, 4, 1, 6, 2, 4, 7, 6};

Dr = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1}; JN = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

As all jobs are due on day 1, we are only concerned with $i = 1$. Therefore Pr = Pr1 and PRr = PRr1, and we can skip directly to step 3. We now initialize $q = i = 1$ and proceed to step 5 (Step 4 is not needed as we previously stated Pr = Pr1 and PRr = PRr1). Calculate x = max(PRr) = 10 (i.e. job 1). This allows us to enter step 6. In step 6 we begin our search by checking each job to see if there is any better combinations than the current value x. See Appendix III for tabular representation. What we get from step 6 is the job sequence $\{J_3, J_4, J_9\}$, which has a combined priority of 19. No other combination of jobs can achieve a better priority level; hence this is the best solution. As these jobs are processed on time, no penalty is incurred and the heuristic will proceed to step 8. The heuristic assigns worker 1 on day one with a job sequence $\{J_3, J_4, J_9\}$. Now that these jobs are assigned, the heuristic must ensure that they are never re processed. To do this the $Pr\{J_3, J_4, J_9\} = M$, and $PRr\{J_3, J_4, J_9\} = 0$, giving the overall values as follows:

Pr = {7, 4, M, M, 1, 3, 2, 4, M, 4}; PRr = {10, 3, 0, 0, 1, 6, 2, 4, 0, 6};

After updating the system, the value of r = r + 1 = 2, and the processing begins again. The following job sequences are calculated for day 1 (See Appendix III for more details): $\{J_5, J_6, J_{10}\}$, which is assigned to worker 2, and $\{J_1\}$, which is assigned to worker 3. Now that we have moved into the second day, certain penalties shall be incurred for the remaining jobs. Worker 1 is assigned job sequence $\{J_2, J_8\}$, and worker 2 is assigned $\{J_7\}$. In step 7 we calculate the penalty for these sequences as follows:

PENALTY = PENALTY + $(r\text{-}Dr(J_{x(i)})*PRr(J_{x(i)})*100)$;

Currently PENALTY = 0, but since jobs $\{J_2, J_7, J_8\}$ are processed late this will change. For the sequence $\{J_2,J_8\}$, penalty is calculated as follows:

$$PENALTY = 0 + (2 - 1) * 3 * 100 + (2 - 1) * 4 * 100 = 700$$

Again, for job sequence $\{J_7\}$:

$$PENALTY = 700 + (2 - 1) * 2 * 100 = 900$$

Therefore the total penalty incurred for late jobs in the system is 900. Since the heuristic utilized 5 operators (all three operators assigned on day 1, and operators 1 and 2 on day 2), a wage payout must be incurred:

$$WC = WC + 80$$

At the beginning of the schedule there were no workers assigned, hence WC is initiated at zero. It is assumed that the wage of the operator is \$80 / day or roughly \$8/hr. Due to the small nature of this problem, no wage incentives have been implemented. The wage amount is calculated to be 400, causing a total system cost of \$1300. Hence, excluding any extra costs (i.e. costs to run tools etc.), to run this system of jobs with the given resources will cost \$1300. Due to the small nature of the example, this result is duplicated in the optimization model. Hence the heuristic was able to reach the optimal case for this problem instance.

Chapter 5: Computational Results and Analysis

The following chapter contains specific analysis pertaining to both optimization models and its corresponding heuristic. Several areas and problem instances have been looked at to see how the models react under different circumstances. The chapter is broken up into four main sections which include Tests, Optimization Models, Heuristic Algorithms, and Hybrid Optimization Model.

5.1 Test Parameters

5.1.1 Large Scale Tests

To test the validity of the models, three large scale tests were attempted. The large scale tests included 15 operators, working a total of 3 shifts over a period of 5 days. There were 100 jobs given for each test, which were randomly generated in Microsoft Office Excel 2007. For each problem, the tasks were given a due date, priority level, and processing time. The due dates fell between 1 and 5, which would correspond to weekdays. Priority levels ranged between 1 and 10, 10 being the highest priority and 1 the lowest. Lastly, processing times ranged between 1 and 8, which corresponds to the number of task units assigned to a specific job. As discussed before, processing time is discussed in units rather than simple time, as classic learning curve theory always relates to number of units produced. Each set of problem data can be found in Appendix V.

When running the scenario data in Lingo 9.0 solver, none were able to provide optimal solutions. The problem instances were too large (see Figure 3) for the program to handle. Each test was run for several days to no avail. Two of the problem instances ran out of memory space and the third was never able to establish an initial solution. The tests were then given initial feasible solutions but were still unable to obtain a solution and remained as unknown problems. To be able to solve these optimization models, the Hybrid optimization models had to be invoked, which will be discussed later within this chapter. Figure 3 demonstrates the status window for Lingo 9.0 for one of the large scale tests.

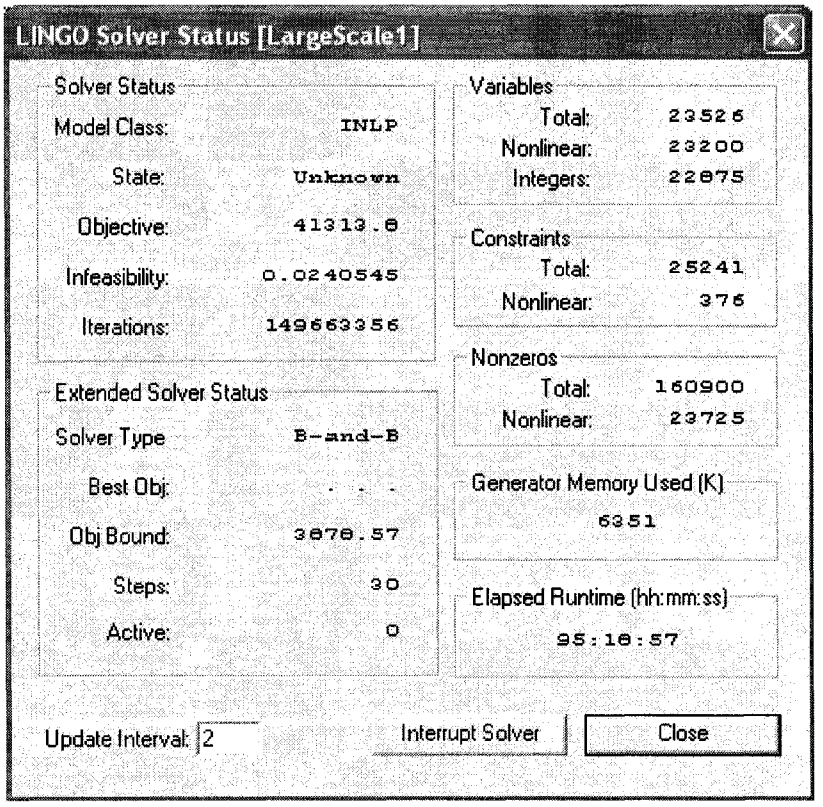


Figure 3: Large Scale Problem Solver Window in LINGO 9.0 Solver

5.1.2 Small Scale Tests

Twenty small scale tests were generated to determine relationships among various parameters. There were four main areas of interest, which are broken up as follows: tests 1 – 4 dealt with due dates, 5 – 7 dealt with learning abilities, 8 – 14 dealt with job priority level, and 15 – 20 dealt with job processing time. All of the test data were randomly generated in Microsoft Office Excel 2007. All of the tests were generated to suit four workers, with 20 jobs. To ease the computational efforts needed by the optimization solver, only one shift was invoked, but still allowing all of the workers to be scheduled if it was optimal to do so. All twenty tests parameters can be found in Appendix V. Each test was run in both Lingo 9.0 Solver and also in the proposed heuristic. A comparison of results between the optimization model and heuristic model is found in section 5.2.6.

The first set of tests dealing with due dates stems from the interest of knowing how a jobs due date affects the outcome of the system. A series of tests was developed to determine if a relationship existed or not. The tests vary the due dates of the jobs from the extreme case of having all of the jobs due on the first day, to a full variability of job due dates. A discussion of the results and any relationships found can be seen in section 5.2.1.

In the second block of tests, which dealt with learning abilities, three tests were developed. Each of these tests were run two times, once with workers with no previous learning experience, and a second time with half of the workers with a more advanced learning state. These tests were used to determine the relationships, if any, between learning abilities and job assignments. A discussion of these results can be found in section 5.2.2.

Another key area of interest is how job priority levels affect job assignments. Seven tests were developed to determine if any relationships existed between a job's priority level and its position in the job schedule. Priorities are broken into the following categories: Low = {1, 2, 3, 4}; Medium = {3, 4, 5, 6, 7}; High = {7, 8, 9, 10}. The tests were devised as follows: Test 1 = Low; Test 2 = Medium; Test 3 = High; Test 4 = Low + Medium; Test 5 = Low + High; Test 6 = Medium + High; Test 7 = Low + Medium + High. The results and discussion can be found in section 5.2.3.

The final tests deal with processing time required for each job. These six tests were used to determine if any relationship exists between a job's position in the schedule and its processing time. The tests were developed similar to the tests previously described for priority. Processing times are broken into the following unit categories: Low = {1, 2, 3}; Medium = {3, 4, 5}; High = {5, 6, 7, 8}. The tests were devised as follows: Test 1 = Low; Test 2 = Medium; Test 3 = High; Test 4 = Low + Medium; Test 5 = Medium + High; Test 6 = Low + Medium + High. The results and discussion can be found in section 5.2.4.

5.1.3 On-Line Tests

The on-line tests consist of three small scale tests, which tests 5NL, 6NL, and 7NL, described in section 5.2.2. These tests are chosen because they have no prior experience and have all free range parameters. Each test inserts a new emergency job into the system on the second day, thus eliminating any of the jobs processed on day 1, as

outlined in the heuristic procedure (refer to Chapter 4 for clarification). The results of these tests can be found in section 5.2.7, with the remaining jobs seen in Appendix V.

5.1.4 Preventative Maintenance Numerical Test

To show how the model works in total, a small example was tested. This test is used to demonstrate how the model decides which jobs to break up, based on the priority and processing time of jobs. The parameters of the test can be shown in Table 3. The workers for this test demonstrate all of the same characteristics, i.e. 95% learning curve and zero experience.

JOB	Processing Time	Priority	Due
1	12	10	1
2	4	3	1
3	6	8	1
4	2	4	1
5	1	1	1
6	3	6	1
7	2	2	1

Table 3: Preventative Maintenance Parameter Set

The model, as described previously, also aims to minimize the worker cost by trying to assign the jobs to as few workers as possible. The complexity of the model however is extremely great, and as such Lingo can not guarantee global optimal solutions. The best Lingo can provide is a local optimal feasible solution. This test was run multiple times and several different solutions were generated. The best solution that was developed is shown in Table 4.

worker	shift	day	job	Amount Processed
1	1	1	1	8
3	2	1	1	4
1	1	2	2	4
2	1	1	3	6
3	2	1	4	2
3	2	1	5	1
2	1	1	6	2
3	2	1	6	1
1	1	2	7	2

Table 4: Output from Preventative Maintenance Model

What we can see is that because job 1 is beyond the processing capabilities of any workers, it has been decomposed. Another job of interest is job 6. Worker 2 began processing this job, but was unable to complete it. A plausible reason for this was because he/she was working on job 3, which has a larger priority. It is assumed that when jobs 1 and 6 were resumed, the worker began where the last left off. The final solution for this instance was 820. This consists of four workers being scheduled at a cost of 80 dollars per day, and two jobs (jobs 2 and 7) being late for a penalty of 500 (based off of priority). This solution does not indicate that it is optimal; it is merely the best case that was obtained. To ease the problem, simplified models were tested. These models can be found in Appendix I and II.

5.2 Test Results and Discussion

In order to reduce the problem complexity, simplified models were tested, which are outlined in the Appendix I and II. A corresponding explanation for each model can also be found within these Appendices. The model complexity was too great, thus could not be handled within a reasonable time frame; thus the need for the simplified models. Although the problems are simplified, they still do not fit any general case, and are still quite complex in nature.

5.2.1 Due Dates Tests

As stated previously, the first set of tests deals with varying due dates of jobs. Each job has a specific due date, which if not met, shall incur a penalty based on its priority level. Four tests were created to validate the importance of job due dates. The first test deals with jobs all due on day 1 (i.e. the first day jobs are available). The second and third tests have a mixture of due dates ranging between the first two days of the work week. The last test has a larger variety of due dates which range the entire work week. The data, as stated previously, was generated using the rand() function in Microsoft Office Excel 2007 and can be found in Appendix V.

Test	Total Pr	Total PRr	Total Penalty
1	75	105	3900
2	105	119	5980
3	89	115	4260
4	87	88	960

Table 5: Test data for Test 1 - 4

Table 5 shows comparisons of total processing time, priority, and penalty incurred for each of the four tests. From these statistics it can be seen that the ratios between penalty and processing time, and penalty and priority are fairly evenly distributed for the first three tests. The last test however was able to achieve an optimal answer consisting only of worker costs. A relationship that can be seen is how due dates affect penalty values.

Relationship 1: *The closer the due dates are scheduled together towards the initial time of job release, causes a larger penalty value to be incurred.*

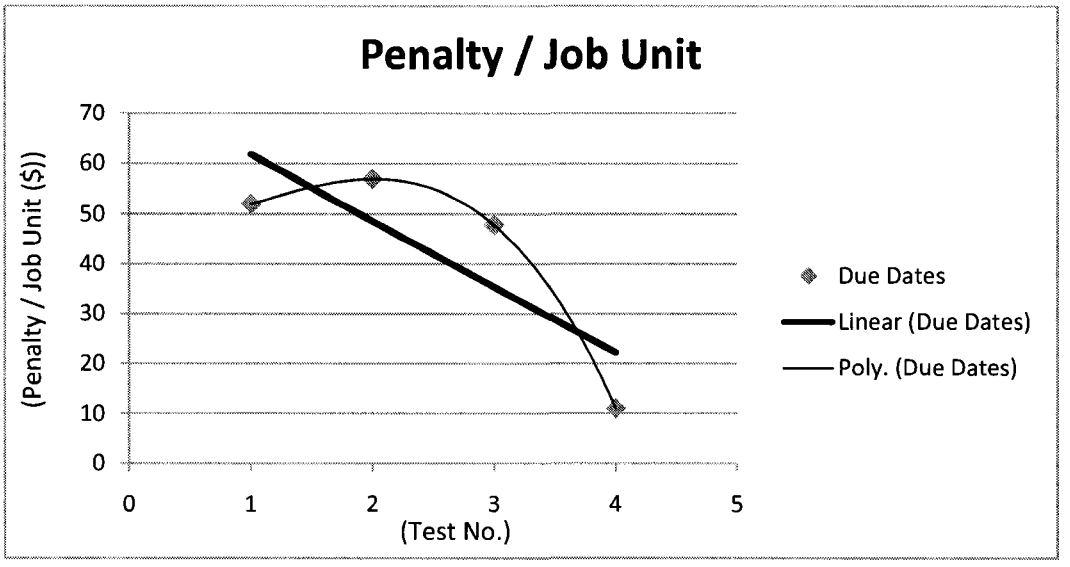


Figure 4: Due Dates - Penalty / Job Unit

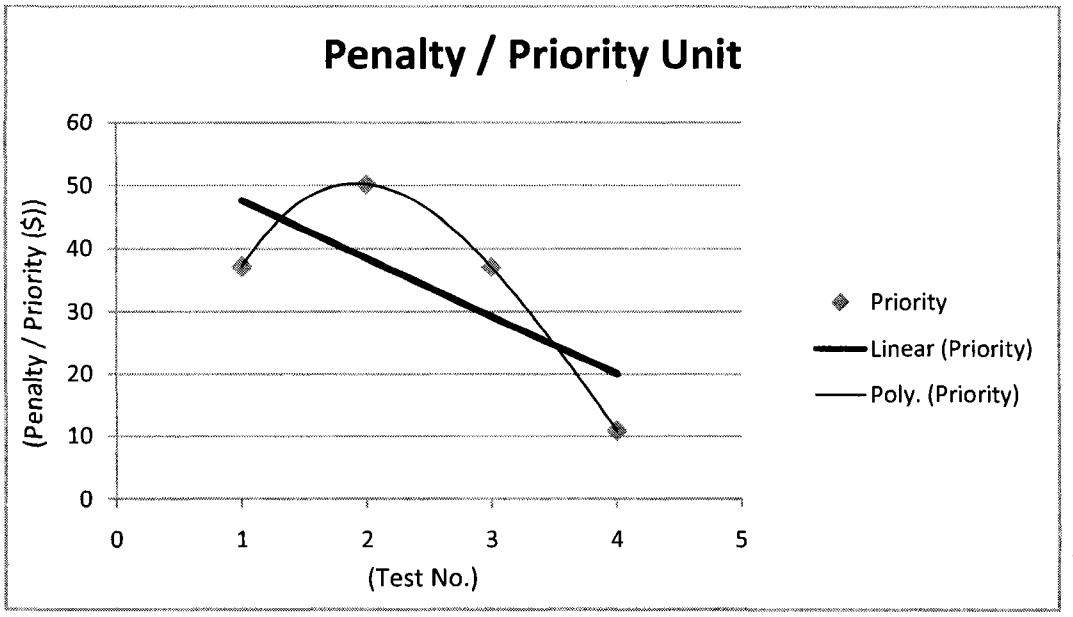


Figure 5: Due Dates - Penalty / Priority Unit

Figure 4 depicts the ratio of penalty to job units (i.e. test 1 ratio = 3900 / 75 = 52). It can be seen that Test 4, which consists of a mixture of job due dates (see Table 6) has a lower average penalty than Tests 1-3 which causes a decreasing linear trend line as jobs become more evenly dispersed. This relationship can also be seen in Figure 5. To see that the first three tests produce an almost linear relationship, Figures 6 and 7 have excluded test 4. Hence, early due dates for tests 1, 2, and 3, which are due on days 1 and 2 show no real significant deviations from each other. There is a significant difference between the first three tests and test 4 however, which demonstrates the validity of relationship 1.

	Jobs / day				Workers / day			
Test	1	2	3	4	1	2	3	4
Day 1	20	10	12	2	4	4	4	2
Day 2	0	10	8	2	4	4	4	2
Day 3	0	0	0	6	2	4	4	2
Day 4	0	0	0	3	0	4	0	3
Day 5	0	0	0	7	0	0	0	3

Table 6: Test Parameters for Due Dates

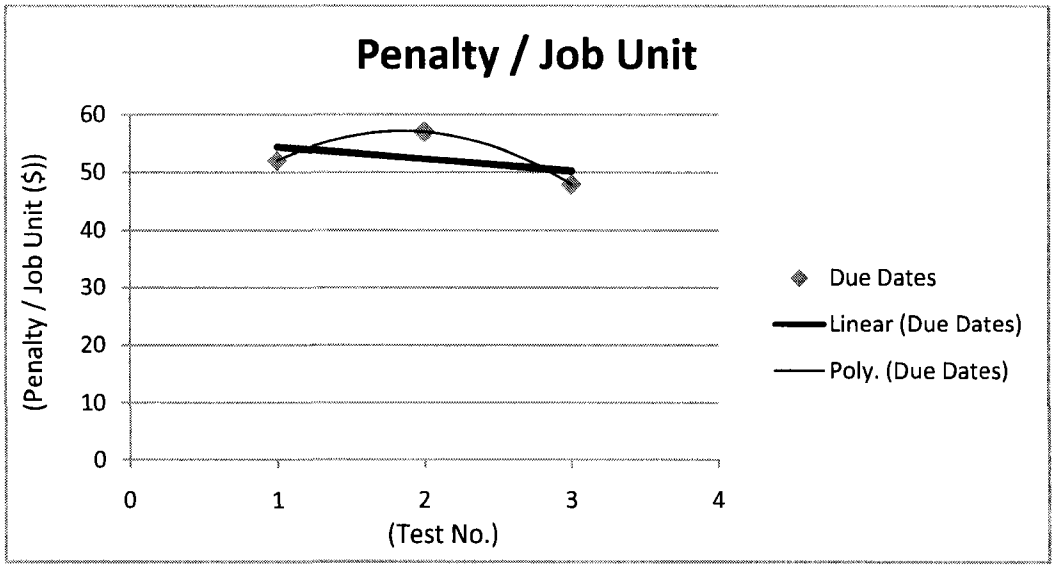


Figure 6: Due Dates - Penalty / Job Unit less Test 4

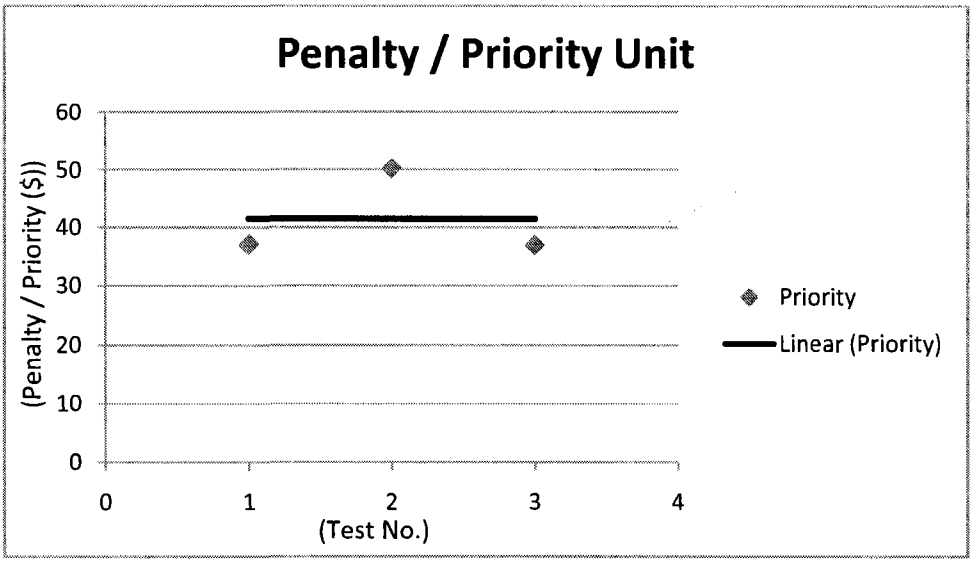


Figure 7: Due Dates - Penalty / Priority Units less Test 4

Another relationship that can be seen lies between average job lateness and due dates. Figure 8 shows the average job lateness for each job. These values are calculated as follows:

$$\% \, Lateness = \frac{(\#\ of\ jobs\ processed\ late)}{(Average\ Due\ Date)} * \frac{1}{n} * 100\%$$

(n = total number of jobs)

The first test produced the most late jobs, due to the fact that all jobs are due on day 1. Tests 2 and 3 have less late jobs than test 1, and on average it shows this. Lastly, test 4 has no late jobs, hence the average lateness for this type of due date schedule is zero. The relationship found is as follows

Relationship 2: *The closer the due dates lie, and the closer they are scheduled towards the initial time of job release, the larger the percentage of daily lateness.*

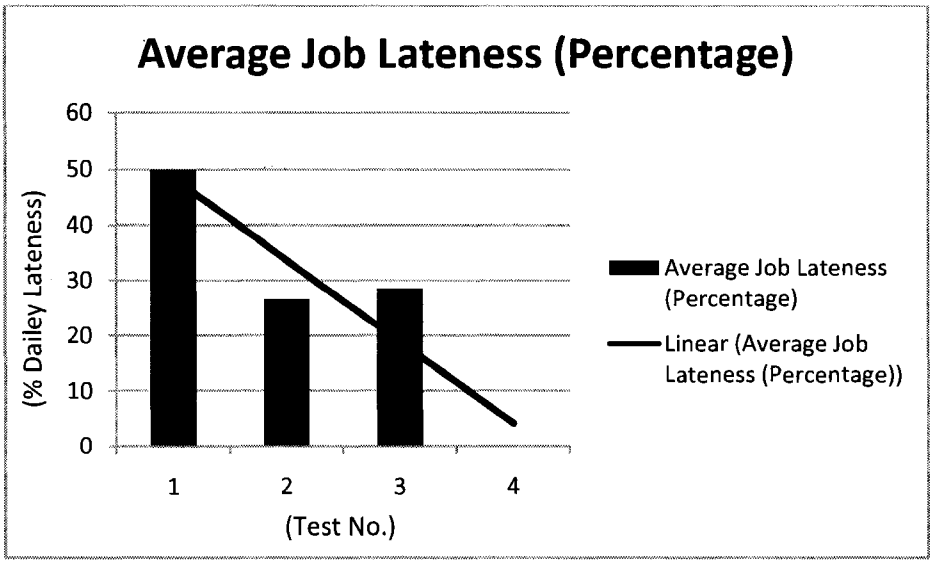


Figure 8: Due Dates - Average Job Lateness

5.2.2 Learning Abilities

The next set of tests is aimed at determining if any relationships exist between the learning abilities of workers and a job schedule. Three tests were invoked to try to determine these relationships, where each test was run twice. Each test was run with no previous worker experience and a second time with workers of varying experience. The tests performed with no previous experience will be referenced as 5NL, 6NL, and 7NL, which correspond to tests 5, 6, and 7. Tests that have workers with previous worker experience will be referred to as 5L, 6L, and 7L, which correspond to test 5, 6, and 7. Tests 5NL, 6NL, and 7NL, were run with the initial worker experience parameters of 0 (i.e. worker 1 = 0 experience). Tests 5L, 6L, and 7L, were run with worker 1 having 95 days of previous experience and worker 2 with 65 days of previous experience. This allowed for these workers to be placed up higher on the learning curve, thus allowing them to accomplish more tasks. Workers 3 and 4 were again set to 0 work experience, thus completing a mixed worker learning schedule. To allow for ease of computational results, each worker was given the same learning index. Table 7 shows a comparison between various parameters for the schedules that were tested, which include the total priority, processing units, and penalty.

Test	Total PRr	Total Pr	Total Penalty
5NL	117	83.6	4160
6NL	115	92.7	3940
7NL	114	80	2580
5L	117	83.6	2752
6L	115	92.7	1828
7L	114	80	1572

Table 7: Parameters for Learning Tests

From the statistics in Table 5, the following figures are obtained. Figures 9 and 10 depict the ratios between penalty and job units, and penalty and job priority respectively. Figures 9 and 10 have an average reduction of 42% in penalty per job unit and penalty per priority unit when learning experience is present to that of no experience. Both figures indicate a downward linear trend. The reason for this is due to the penalty value of the job schedules. Tests 5 and 7 are extremely close with respect to average priority level and processing units, yet the penalty value of test 7 is approximately 38% less than penalty value of test 5. Thus the downward trend must be related to the penalty, which corresponds only to the parameters that are selected. From these graphical results a new relationship has formed which is as follows:

Relationship 3: *The more experience a worker has, and the more experienced workers that are present, cause significant reductions in the penalty value of the system.*

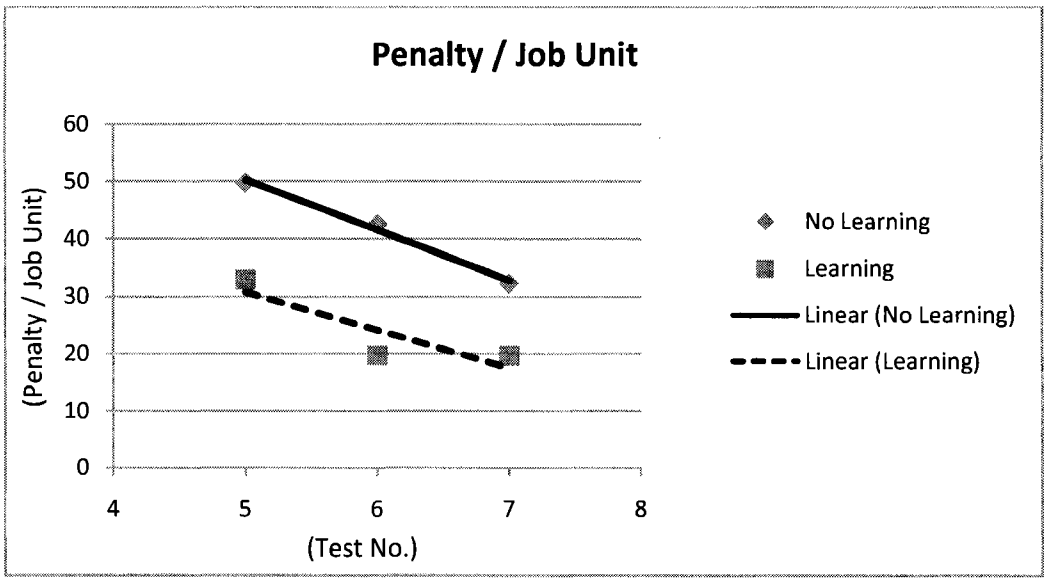


Figure 9: Learning Abilities - Penalty / Job Units

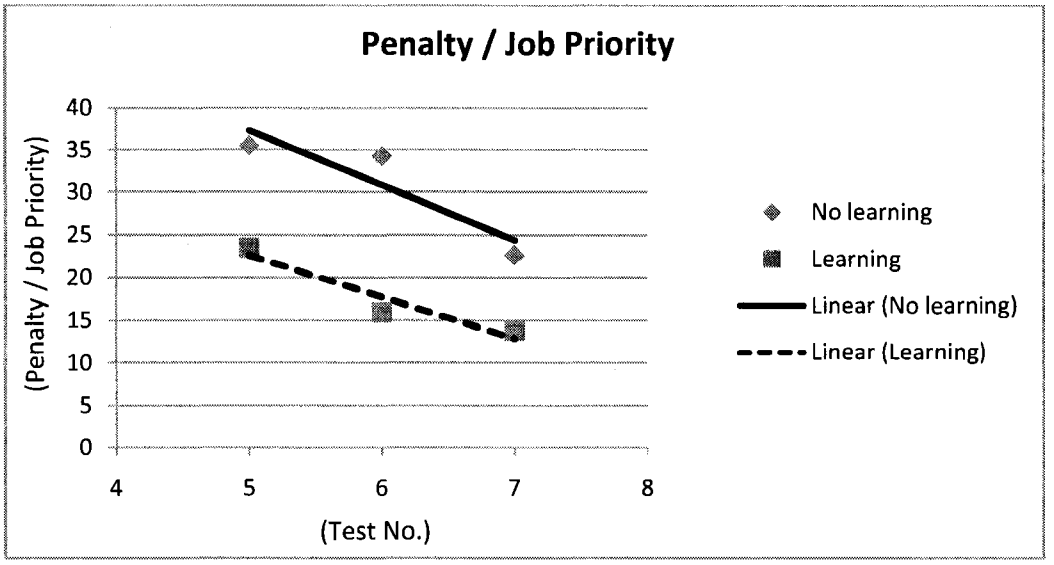


Figure 10: Learning Abilities - Penalty / Job Priority

The next area of interest lies within the workers being scheduled. Figure 11 depicts the cost of workers for both experienced and inexperienced workers. What is seen is a decrease in overall cost when experienced workers are present. The reason for this is that

that less workers are needed to fulfill the required tasks on time. What is interesting to note is the mixture of workers within the system when experience is present. Because it costs more to bring in a worker with more learning experience, the models determine the best possible mixture of workers to bring in. Take for example in test 5, all workers are scheduled on days 1 and 2, but on day 3, worker 1 and 4 are brought in. This is due to worker 1's ability to process more job units than any other worker. Since it costs only 12 more dollars to bring in worker 1 vs. worker 2 and 3, it is more cost beneficial to bring in worker 1. Thus from these results, a new relationship is formed

Relationship 4: *When experienced workers are present in a mixed experience environment, overall worker cost will decrease as experienced workers can accept more jobs.*

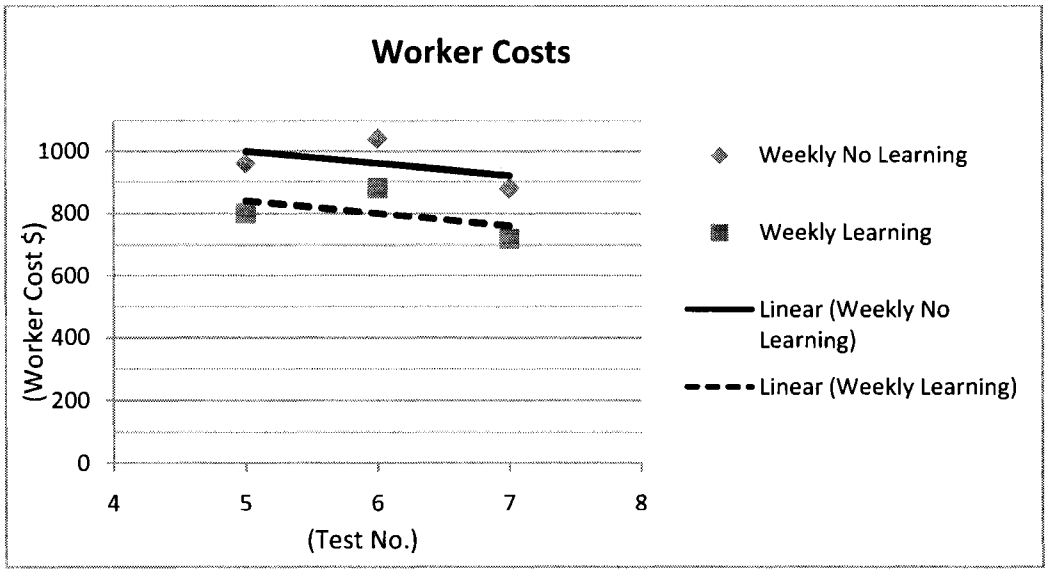


Figure 11: Learning Abilities - Worker Costs

The last area of interest is with respect to job lateness. It can be seen in Figure 12 that when learning is present, reductions in job lateness can be seen within the range of 20 – 60 %. Referring to relationship 4, the reduction in average job lateness stems from workers being able to handle more jobs at any given time, thus reducing the amount of jobs that are late. Some values used in Figures 11 and 12 can be seen in Table 8. Table 8 shows how many jobs are due per day and how many workers were scheduled per day. The values to be noted in this table are the reduction in workers from the tests with no experience to the tests with experience present.

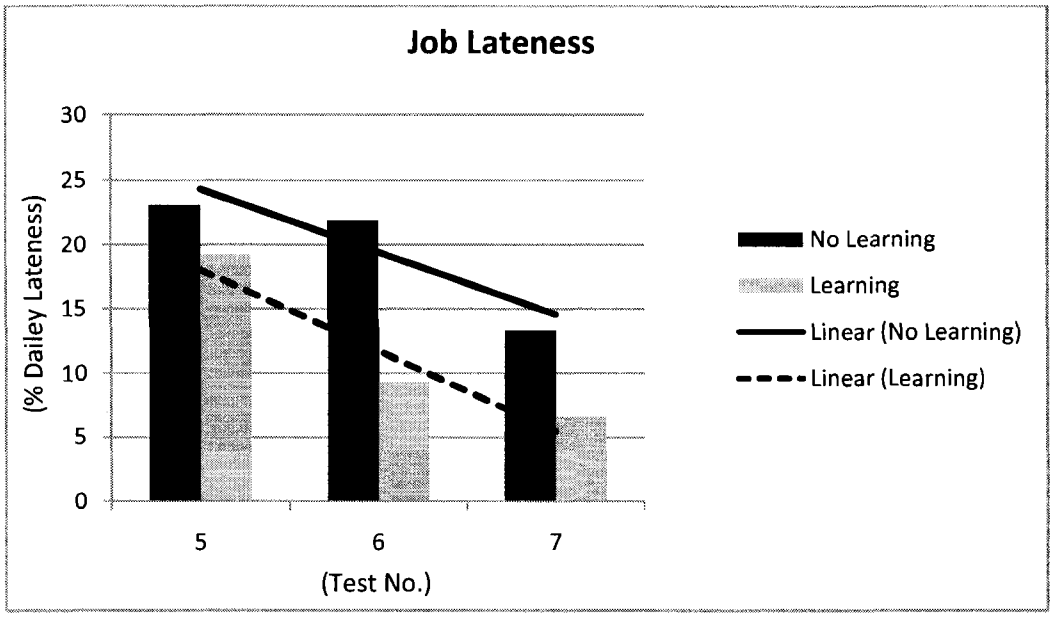


Figure 12: Learning Abilities - Job Lateness

	Jobs / day						Workers / day					
DAY	5NL	6NL	7NL	5L	L6	7L	5NL	6NL	7NL	5L	L6	7L
1	14	9	10	14	9	10	4	4	4	4	4	4
2	6	11	10	6	11	10	4	4	4	4	4	4
3	0	0	0	0	0	0	4	4	3	2	3	1
4	0	0	0	0	0	0	0	1	0	0	0	0

Table 8: Learning Test Data

5.2.3 Priority Tests

Priority plays a large role in determining how much penalty is incurred as it directly affects the penalty value. Because of this, seven tests were developed and tested to determine how important the priority levels of jobs are. Tests 8, 9, and 10 depict separate priority levels consisting of low, medium, and high (refer to section 5.1.2 for clarification on these sets). Tests 11, 12, 13, and 14 consist of mixtures of these priority levels. Figure 13 depicts the relationship of priority for each of the seven tests. The priority levels are steadily increasing while the average processing time for each of the tests remains steadily constant.

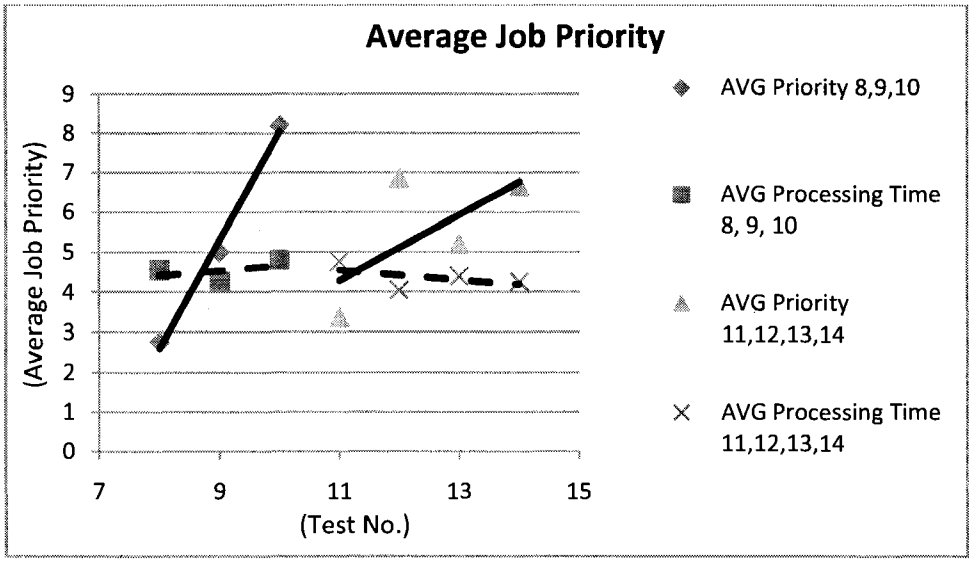


Figure 13: Priority Value - Average Job Prioty / Test

After running the tests, a few areas of interest surfaced, the first being the average penalty per job unit. There was very little fluctuation in these values, the only case being test 10,

which is an extreme case where all priorities lay on the high end of the scale. A possible reason for this stems from the fact that the average job units per test remain fairly constant. Another reason for this could stem from the tests themselves producing fairly low penalty values. In Figure 15 however, there is a downward linear trend with respect to the average penalty per priority unit. The more evenly distributed priority values seem to have the lowest average penalty. The reason is the ratios of priority to penalty will decrease for the schedules with many high priority jobs, and will increase for those with many low priority jobs. Thus it is seen in tests 8 and 11, which contain many low priority jobs, and have corresponding high penalty values, that the ratios are of the highest. Referring back to Figure 13, it can be seen that the average priority level for test 8 is much lower than that of test 10, yet the ratio of penalty to priority remains lower, even though the penalty is more than double.

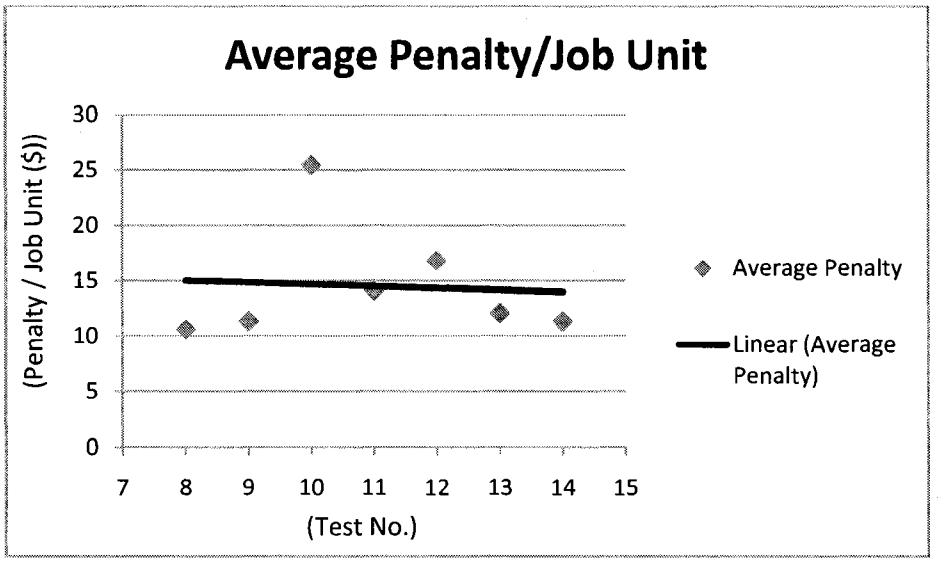


Figure 14: Priority Value - Average Penalty / Job Unit

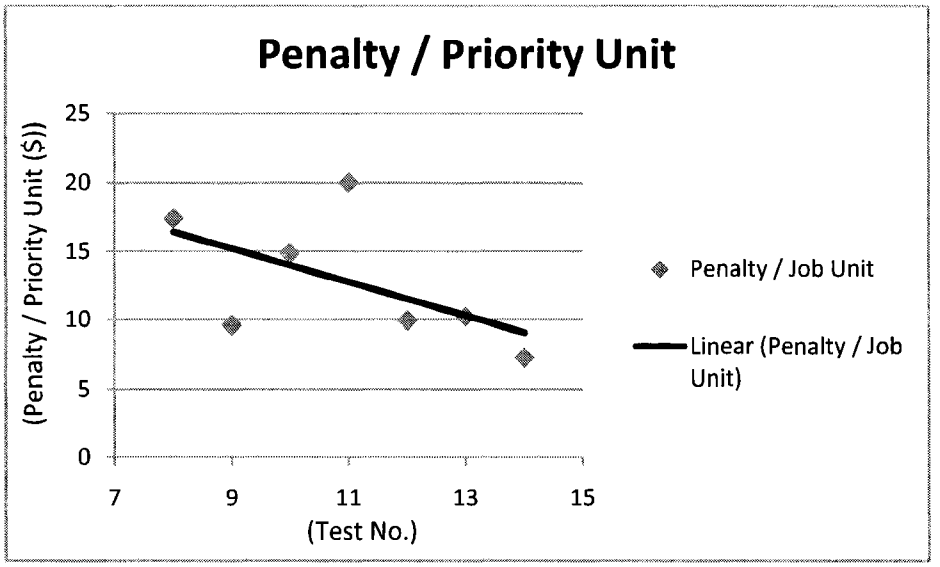


Figure 15: Priority Value - Average Penalty / Priority Unit

Thus it can be seen that priority plays no significant role in determining the outcome of the schedule. When all priority levels are set high, more penalties will be incurred when jobs cannot be processed. The model determines the best mixture of jobs to be processed; hence the next set of tests, which deal with processing times, may have more bearing on the outcome of the system. The only reasonable conclusion to be reached is that although priority directly corresponds to the penalty value of the system, priority seems to place its onus on the arrangement of the jobs that are scheduled. Referring to Appendix V, it can be seen that the jobs to be sacrificed are all of the lowest priorities in the system. Thus priority level does not determine how many jobs can be scheduled to a worker, but it does in turn try to optimize the jobs that the worker should be looking to complete. Thus a new relationship has been formed:

Relationship 5: *Job priority level depicts the jobs positioning within the given schedule*

This relationship is a key portion of the heuristic model, described in Chapter 4.

One last area of interest is how job priority acts within the confines of job lateness. The average percentage of lateness can be seen in Figure 16. There is very little lateness in any of the tests, but it can be noted that because priority dictates which jobs should be processed first, more job lateness can occur, unless optimization techniques are implemented. Again, as can be seen in Figure 16, the tests provided very little job lateness, thus no real relationship can be established.

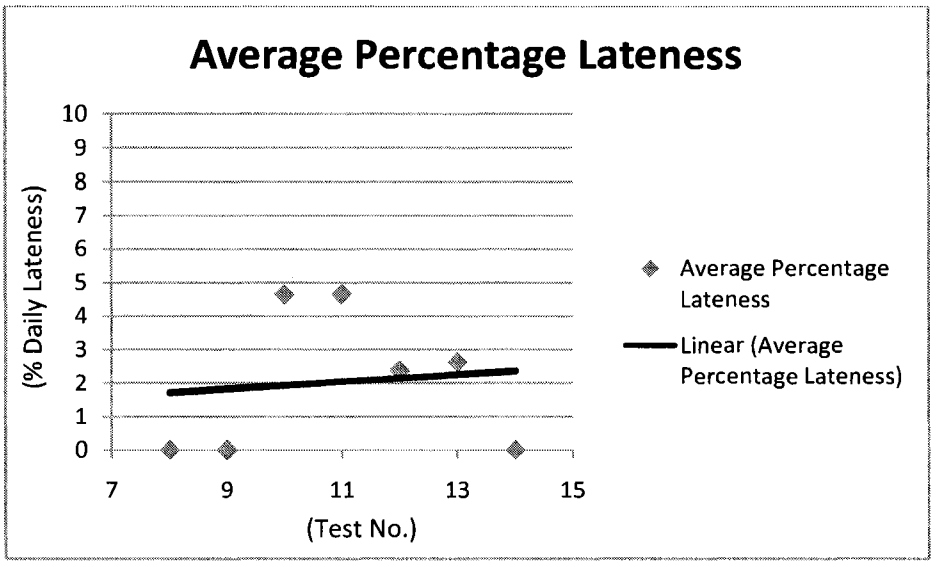


Figure 16: Priority Value - Average Percentage Lateness

5.2.4 Job Processing Units

The final set of small scale testing lies within the job processing units. Again to reiterate, we define time with respect to units, as classical learning depicts the doubling affect with respect to units processed, where we define a single job to contain several units of work. The doubling affect surfaces with respect to the cumulative experience in days a worker possesses, which in turn, increases the number of job units they can process.

Tests 15 – 20 were modeled similarly to tests 8 – 14. The first three tests (15, 16, 17) vary the sets of processing time between low, medium, and high (Please refer back to section 5.1.2 for clarification on these sets). The last three tests contain a mixture of these three sets. Figures 17 and 18 depict the average penalty per job unit and per priority unit. Both figures provide a linear and polynomial trend line to the power of 3. The reason for the polynomial trend line is to show the relationship between tests 15, 16, and 17. There is a sharp rise in penalty for both figures. The reason for this is simple, the workers can not process enough of the jobs on time, thus many jobs are delayed. In test 15, the only penalty incurred is that from the workers wages, which is due to the fact that all the job processing units are low, thus no late jobs. The other extreme is test 17. In this test, all of the processing units are in the high category, thus many jobs are processed late, as can be seen in Table 9. When the tests provide small processing times little to no jobs are processed late (i.e. tests 15 and 18). When tests contain many large jobs to be processed, many jobs are processed late (i.e. tests 17 and 19). See Figure 19 for more detail on job lateness. Thus a new relationship is formed.

Relationship 6: *The mixture of processing units of jobs directly affects the value of penalty and job delay.*

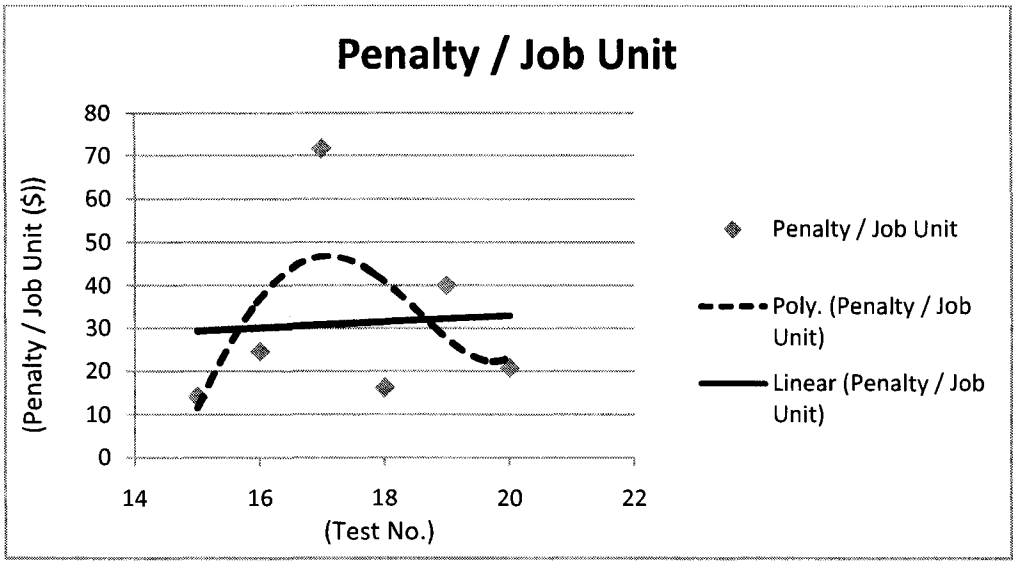


Figure 17: Processing Units - Average Penalty / Job Unit

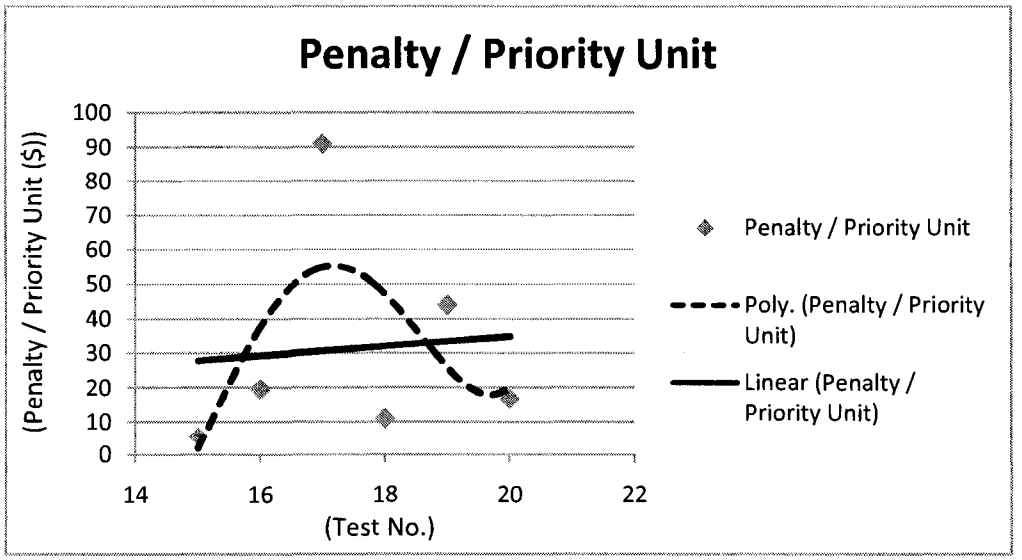


Figure 18: Processing Units - Average Penalty / Priority Unit

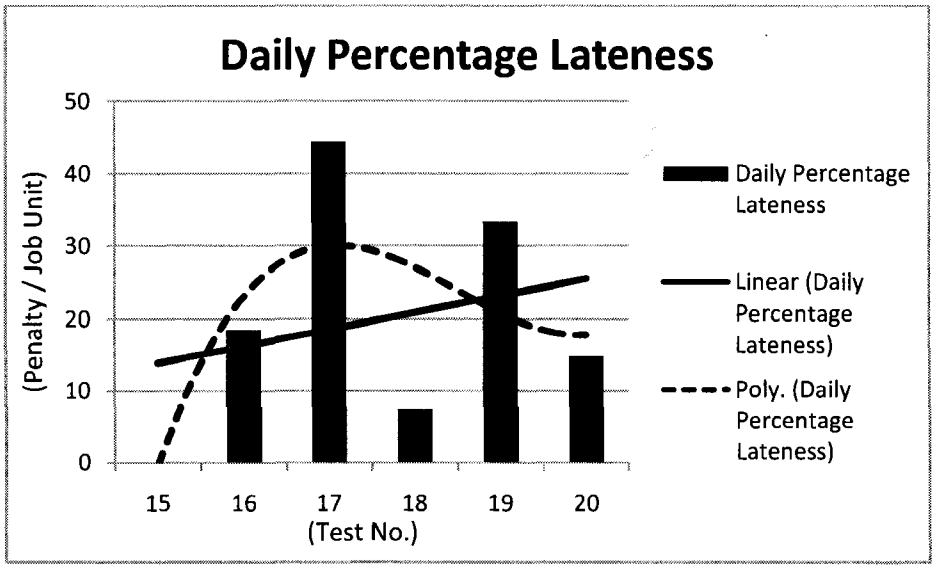


Figure 19: Processing Units - Daily Percentage Lateness

Tests	15	16	17	18	19	20
Jobs Processed Late	0	5	12	2	9	4

Table 9: Comparison of Jobs Processed Late

The last area of interest is how job processing times affect the worker schedule, thus directly affecting the amount of worker costs associated with the system. Figure 20 depicts a linear increase in cost as job processing units increase. The biggest distinction can be seen again in tests 15 – 17, which show two extreme cases. Test 20 utilizes a mixture of all types of job processing units and has a value in the middle of tests 15 – 17. Thus another new relationship can be formed.

Relationship 7: *As the mixture job processing units increase, the cost to processes those jobs also increases.*

The reasoning behind relationship 7 is fairly simple. It takes more resources to complete the jobs set out by the schedule, as each worker is able to complete less jobs on any given shift. Table 10 depicts the worker distribution per day for each of the six tests.

	Workers / day					
Day	15	16	17	18	19	20
1	4	4	4	4	4	4
2	3	4	4	4	4	4
3	0	4	4	2	4	3
4	0	0	4	0	3	0
5	0	0	4	0	0	0

Table 10: Comparison of Workers scheduled

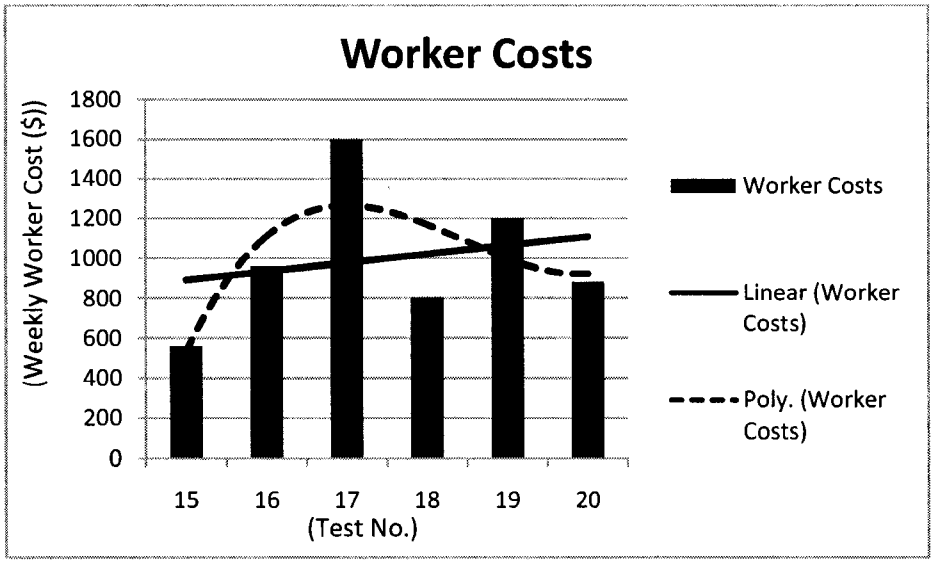


Figure 20: Processing Units - Worker Cost

5.2.5 Large Scale Tests

To test the strength of the model, three large scale tests were invoked. The problem however was the inability for the computer to solve the problem in its standard

state. To encompass this issue, and still maintain feasibility of the problem, an initial solution was invoked into Lingo 9.0 Solver, which was produced with the heuristic. This allowed for Lingo to start off with at least some of the variables in a feasible state. Lingo however, still was able to change the parameters as it saw fit to decrease the objective function. Hence some of the values that are found in the heuristic model may have changed in the optimization model.

The models themselves are quite complex, as they are of the MINLP family. The large scale test would have 100 jobs, 15 workers, 3 shifts, and 5 days, thus producing over 22,000 binary variables. The complexity of this model can be thought of in terms of these binary variables, which produces as many as 2^{22000} possible combinations, which is unreasonable to solve. Thus to simplify the problem, as was done in the small scale models, 3 shifts is reduced to 1 shift, bringing the total combinations to 2^{7500}, still a large sum; hence the need for an initial solution. The large scale problems took anywhere from 48 – 100 hours to solve, even with the initial solution. One instance, the model could still not be solved, given the initial solution. Three models were able to obtain feasible solutions, and it is these three models that are under investigation.

To begin the analysis of these three models, a basis must be chosen for comparison. Tests 5NL, 6NL, and 7NL are used as a comparison method for the large scale tests. The reason for choosing these tests is because they had no restrictions on the priority, due dates, or processing units, hence they are prime candidates. To begin, penalty/ job unit and penalty / priority unit will be looked at. Figures 22 and 23 show these relationships. What can be drawn from these two figures is that the average penalty is much higher for the small scale tests. Although the penalty values, as shown in Figure

21, are approximately the same for both tests, as there are more jobs to be processed in the larger scale models, the penalty ratios will drop. One thing to mention however is the fact that penalty values are so similar. This leads to the conclusion that the size of the problem does not necessarily affect the lateness penalty; however there is more susceptibility to larger penalties depending on the distribution of due dates, and processing units (i.e. relationships 2 and 6).

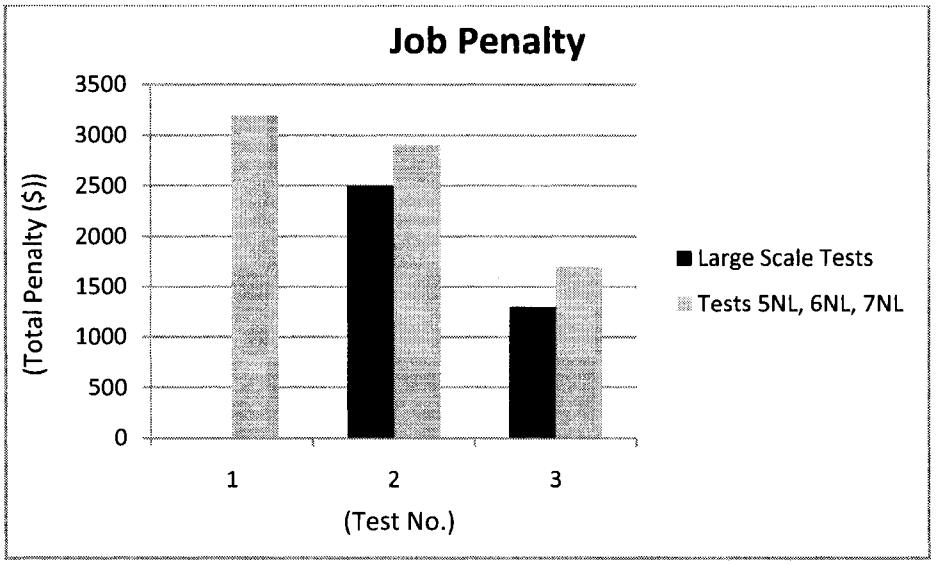


Figure 21: Large Scale - Job Penalty

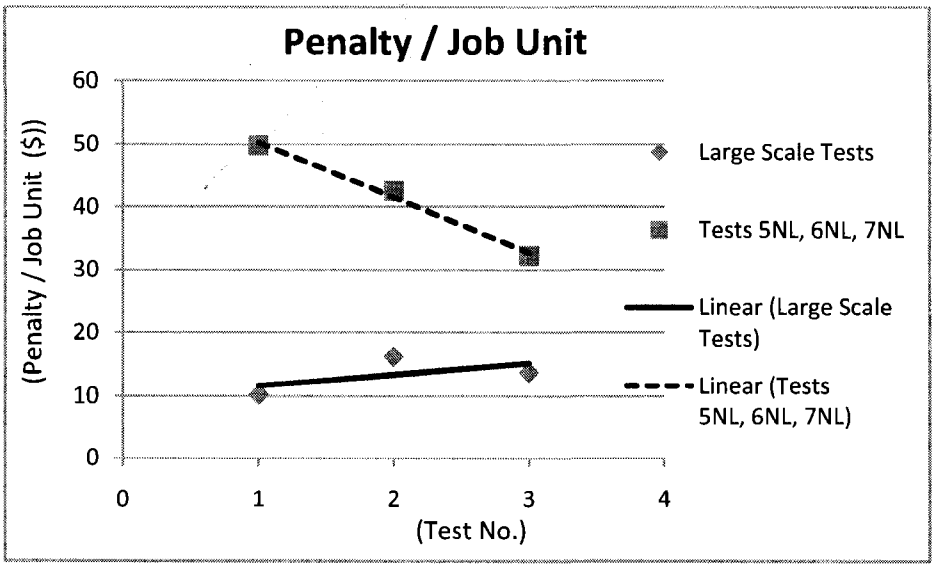


Figure 22: Large Scale - Average Penalty / Job Unit

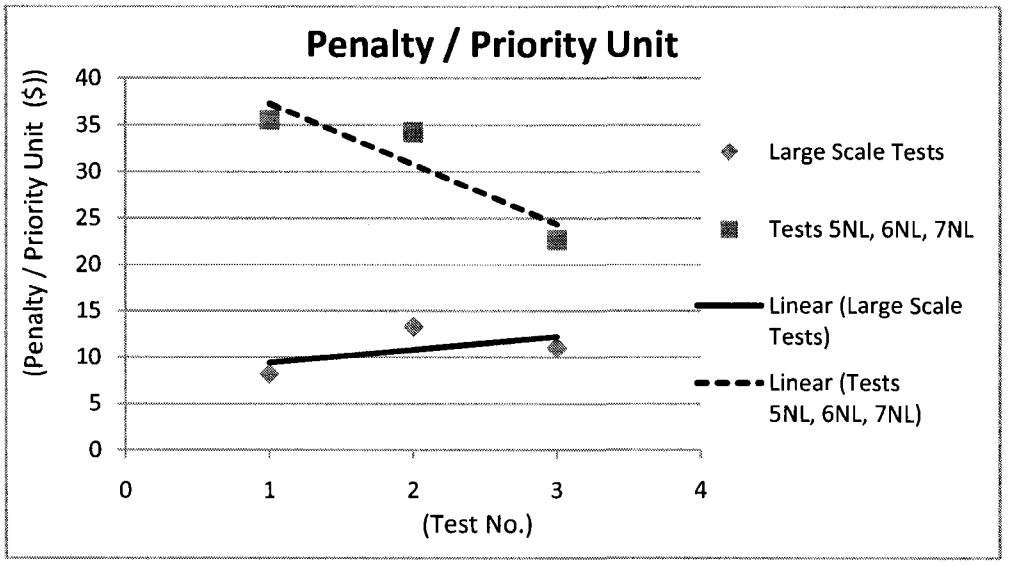


Figure 23: Large Scale - Average Penalty / Priority Unit

If job penalty does not necessarily make a difference between large and small scale problems, then the problem really comes down to worker scheduling. It would be infeasible to think that four workers found in a small scale problem could handle 100 jobs

in one week. Thus, with large scale problems, a significant portion of cost is related to the resources available, i.e. the workers. By comparison, Figure 24 shows the costs associated with both large scale and small scale problems. Tests 5NL, 6NL, and 7NL have an average cost of around $1000, while the large scale problems have an average cost of almost $4650, more than quadruple the small scale problems. Hence, large scale problems require much more resources to fulfill the job orders, while still maintaining a minimum penalty cost.

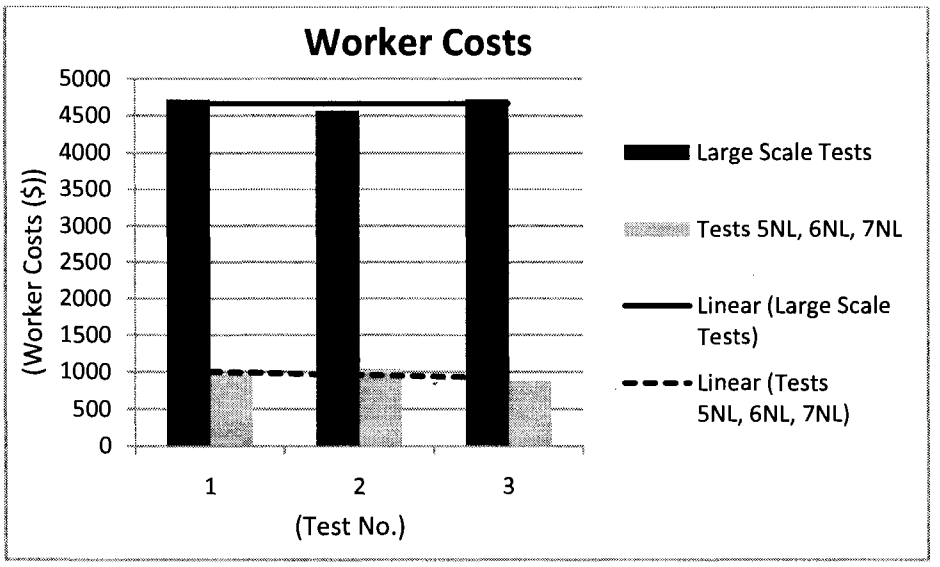


Figure 24: Large Scale - Worker Costs

Again, reiterating the aforementioned, as there are more jobs to be processed in large scale problems, the ratio of percentage of daily lateness will decrease. Although similar quantities of jobs are processed late with regards to the large scale and small scale tests, the ratio of jobs complete is much higher in large scale problems than in small scale

problems. Thus the average percentage of daily lateness for small scale tests 5NL, 6NL, and 7NL range much higher than that of all three large scale tests.

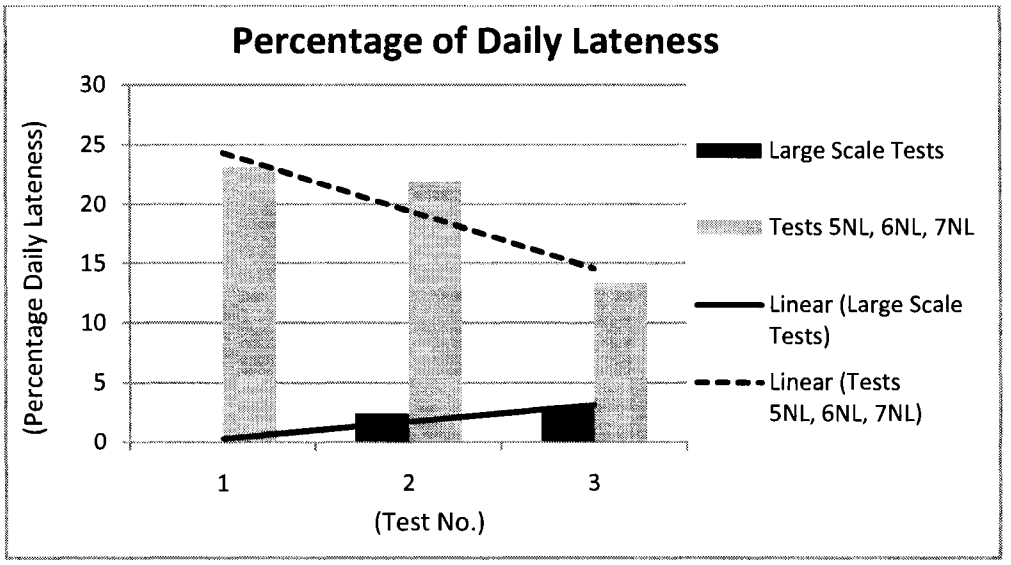


Figure 25: Large Scale - Percentage Daily Lateness

In performing the large scale tests, it should be noted that the computational time is far greater than the small scale tests. On average, it takes 1 – 6 hours to perform one small scale test, whereas a large scale test can not be solved without an initial solution, thus the necessity for such small scale tests. The initial solutions were provided by the heuristic, but Lingo was still able to change the parameters as it saw fit.

5.2.6 Heuristic Comparisons

All of the tests were performed in the heuristic model to be compared to the optimization models. As stated in section 5.1.1, the heuristic provided a basis for an

initial solution in the large scale problems. The necessity for the heuristic is due to the computational time savings gained. For use in industry, one can not wait a week for a schedule to be made for that week, thus, the heuristic method is invoked as it is able to give a good solution.

There are 26 tests in total that are used for comparison between the heuristic and the optimization models. These tests are described in previous sections within this chapter. The first base for testing is solely on job penalty. Of the 26 tests, the heuristic was able to achieve the same job penalty for 5 tests. On average however, the heuristic tends to be roughly 40% higher with respect to job penalty. The reason for this being so high is due to the fact that sometimes the optimization model finds a solution where zero jobs are late, and if the heuristic does not find a solution to match, any job will cause 100% in difference, thus skewing the results slightly. If we eliminate any of the values that fall within this category, the average penalty drops to approximately 30% higher. As can be seen by the graphical representation of job penalty in Figure 26, both polynomial trend lines follow a similar path, the heuristic however shows much more curvature, as it is on average larger.

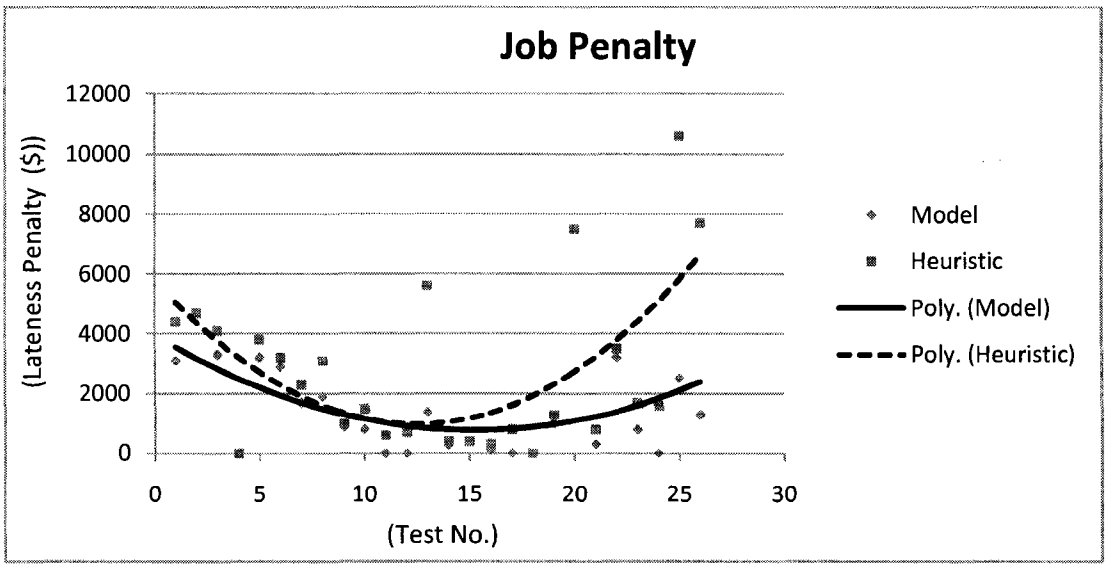


Figure 26: Heuristic Comparison - Job Penalty

Another base for comparison is the worker distribution. The results are almost identical to that of the optimization models. On average there is a 5% variance in results and for 6 of the cases, there was no difference at all, the same worker schedules were found. Figure 27 provides graphical representation of this comparison. As can be seen, both linear trend lines are almost identical, the only reason being is that for the large scale tests, the heuristic had on average 5 more workers scheduled, which is fairly close as they are both dealing with 55 – 65 workers.

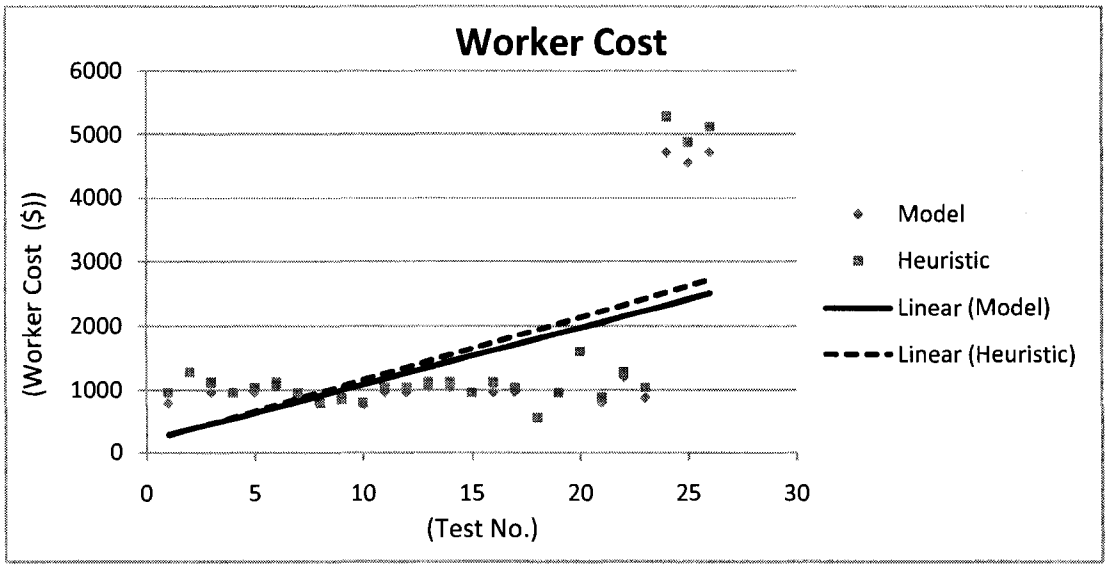


Figure 27: Heuristic Comparison - Worker Cost

The total cost of the system for both optimization model and heuristic can be found in Figure 28. The total cost is the cost of workers and penalty cost of jobs combined. On average the heuristic proves to be roughly 24% higher, but in 5 instances, the costs were exactly the same. Both polynomial trend lines follow the same curvature, and again, as the heuristic is on average larger, the slope of the heuristic trend line is steeper at the head and tail.

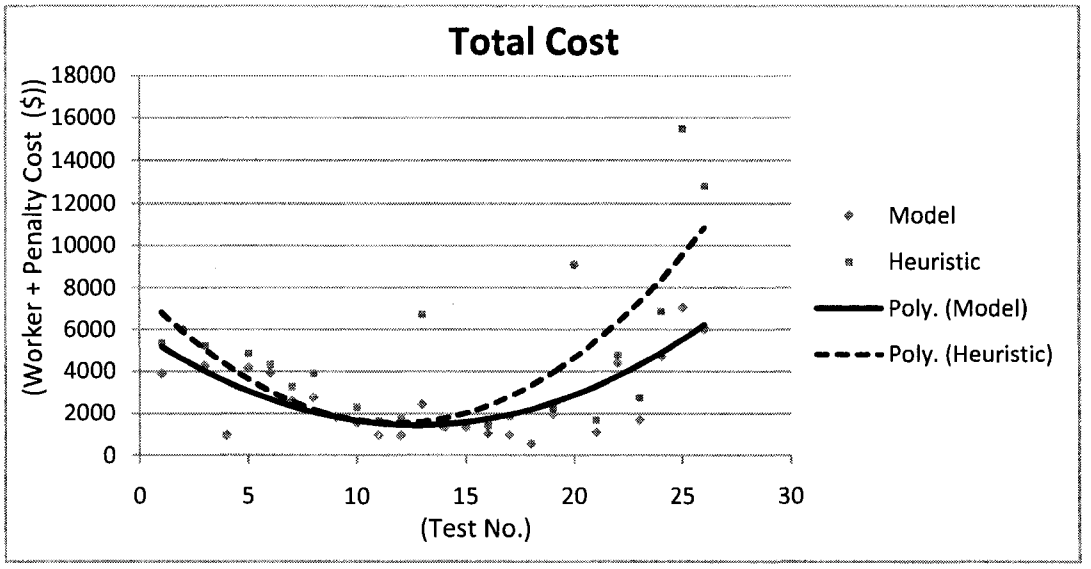


Figure 28: Heuristic Comparison - Total Cost

Lastly, the number of late jobs for the heuristic is approximately 32% higher. This number is again slightly skewed as sometimes the optimization model can find a solution with zero late jobs where the heuristic may have one or two late jobs, thus increasing this value larger. Eliminating these instances brings this value down to 20%, a fairly good statistic.

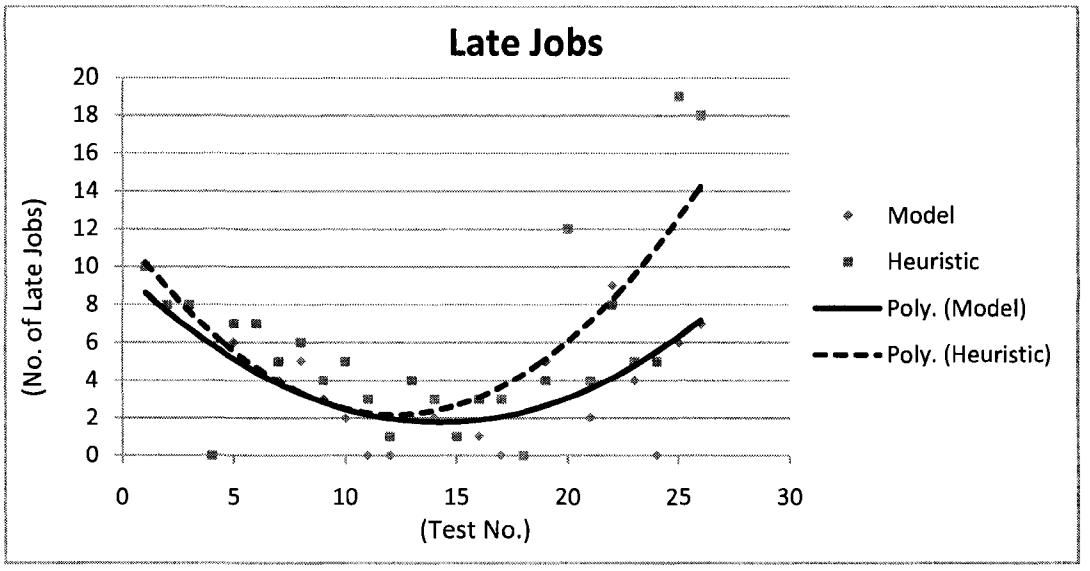


Figure 29: Heuristic Comparison - Late Jobs

In general the heuristic provides a much faster solution than the optimization models. The heuristic needs only a few minutes to produce a large scale problems solution, and mere seconds for the small scale tests. The results of the heuristic are slightly higher, but the time savings are immense, and the solution provided by the heuristic is feasible. It is able to produce optimal solutions as is seen in the given test problems, but there is no guarantee that this solution will be reached.

5.2.7 On-Line Tests

The last area to analyze is the semi-online portion of the model. It has been seen that the model is able to produce a useable worker schedule with job assignment, but what happens if a job enters the system when it hasn't been previously scheduled. The three tests that are used again are tests 5NL, 6NL, and 7NL, due to the nature of these

tests. For these tests, it is assumed that an emergency job enters the system on day 2, and as there is only one shift per day, it will enter on shift 1. For each of the tests, the jobs that have been previously processed on day 1 have been removed from the system, as they no longer need processing. The remaining jobs can be seen in Appendix V. The entering emergency job processing time and priority level for each test is shown in Table 11. Remember, the priority level of the emergency job shall be set to the largest priority in order to ensure its processing placement, i.e. relationship 5.

	New Job	
Test	Pr	PRr
5NL	3.2	10
6NL	4.3	10
7NL	3.6	10

Table 11: Emergency Job Processing Units and Priority

Once the new job is added into the remaining job set and inputting the worker schedule as fixed with the previous model, the semi-online model was run. It was seen that on average the penalty values were roughly 16% larger than the previous formulation. The main reason for this is due to the shifting of jobs from the original positioning. To be able to encompass the new job into the system some jobs are delayed to allow for the processing of the new emergency job. In test 1 (i.e. 5NL), jobs 4 and 16, which in the previous formulation were processed 1 day late, are now processed 2 days late to make room for the emergency job to be processed. The largest penalty gap is seen in test 2 (i.e. 6NL), as more rearrangements were required. Here job 3 and 12 switch processing days as job 3 has a lower processing unit value. Jobs 12 and 20 are processed later, thus allowing for space for the emergency job to be processed. Lastly, in test 3 (i.e.

7NL), a single job (job 12) is delayed one extra day to encompass the emergency job. Thus, when emergency jobs enter the system, on average the penalty values will rise, unless the processing units are so low that they can fit into an existing idle slot, i.e. relationship 7. For a graphical representation of the penalty costs please see Figure 30.

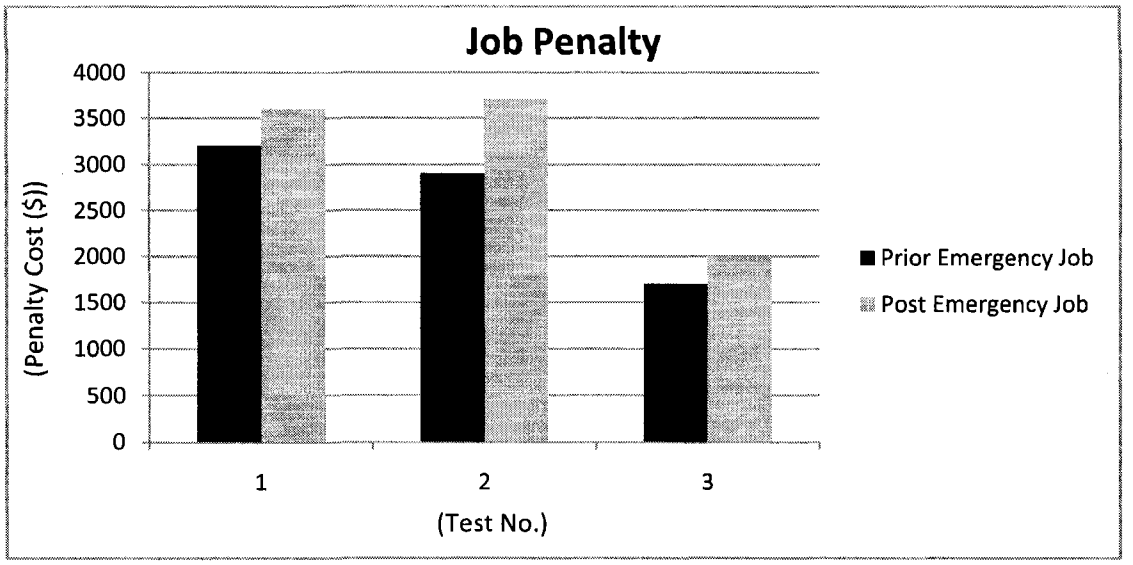


Figure 30: Emergency Model - Job Penalty

Although the worker schedules are fixed, not every case can accept the new jobs, thus it may be necessary to call in a worker, if it is known in advance that the current schedule with the new emergency job can not be met. Such a case was seen in test 2, where the processing units could not be arranged in any schedule to fit with the current state of workers. Thus an extra worker was included on the last day of processing, which in this case was day 3. The current day 3 schedule called for a single worker to be present, but an extra worker was needed. Thus a small variance in the cost of workers is present and can be seen in Figure 31.

Lastly, some jobs from the previous formulations were scheduled late, and it is no different in the semi-online model. The best case scenario would be that the same number of jobs is processed late, where as the worst case scenario is that several more jobs are processed late. In test 1, the best case was found, as the same number of jobs is late, only the jobs themselves have changed, thus increasing the penalty. In tests 2 and 3 however, the jobs had to be arranged in such a way that 1 more job was late than in the previous formulation. The reason being, the processing units of the jobs did not allow for an optimal pairing of jobs, thus it was better to delay certain jobs. This can be seen in Figure 32.

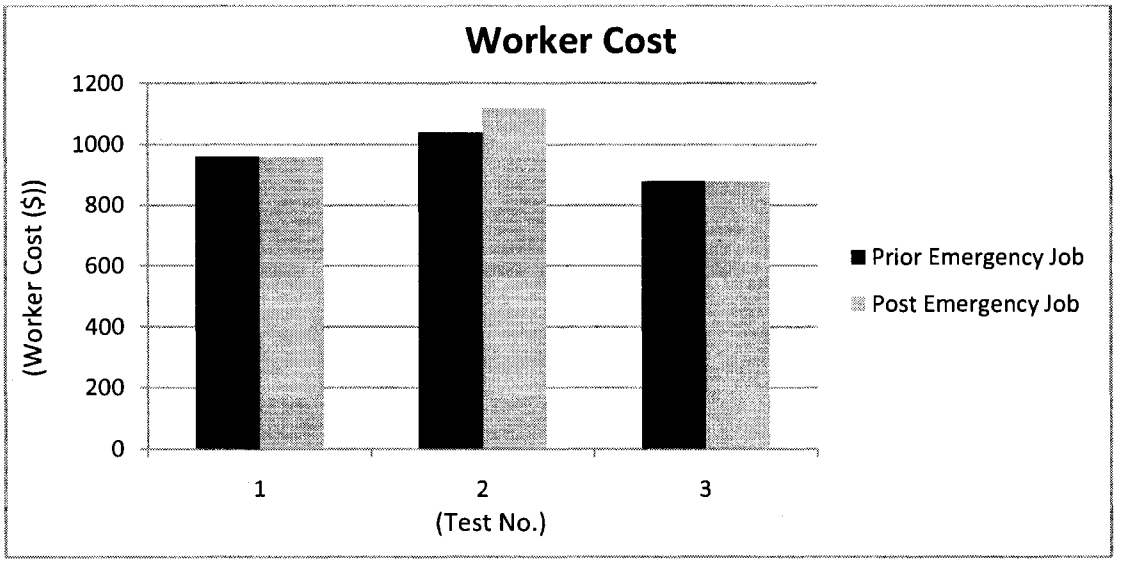


Figure 31: Emergency Model - Worker Costs

Figure 32: Emergency Model - Late Jobs

5.3 Remarks

In closing, the tests provided valuable insight into the abilities of both optimization models and heuristic models. Both have benefits that can be used in industry to aid in the scheduling process. The optimization models, although they computationally take longer to process, can produce very good schedules that can be used in process. The heuristic, still able to give fairly good results, makes most of its luster in its ability to produce a schedule in a relatively short period of time. The tradeoff for time is the heuristics ability to produce optimal results as determined by the optimization models. Both provide feasible schedules that determine worker and job assignments.

Several relationships have been determined from the models. These relationships provide more insight into the models computing abilities, and it is these relationships that have been mimicked into the heuristic model. The relationships found not only aid in the

ability of the heuristic, but can be applied as general guidelines to be followed in industry, when making schedules. Hence, the results provided by this analysis have proven quite valuable.

Chapter 6: Conclusions and Future Work

6.1 Conclusions

This thesis presents a MINLP model in order to obtain an optimal scheduling policy for a real world problem. The problem realizes the input of several job types with similar processing requirements and varying processing times. To optimally assign workers to these jobs is a daunting task, as each worker is seen as its own entity. The workers are learning and growing as they progress throughout their careers, thus the relationship of time based experience has been included within this research. The mathematical models, as discussed in Chapter 3, have formulated these relationships, providing for an accurate depiction of the system in question.

An additional model has been implemented, to solve a semi-on-line state of the system. The original MINLP organizes a full off-line schedule, based on the assumption that jobs will arrive in the system at the beginning of the week, i.e. time zero. The semi-on-line model however is able to accept jobs that may enter the system at some period during the work week. These tasks occur in the real world case, thus must be initialized within this research. After the required steps and assumptions are met, the model is able to rearrange the given tasks in order to accommodate the new emergency job as best as possible. Both models have been solved with Lingo 9.0 solver.

Within Chapter 5, certain relationships have been recognized through vigorous analysis. These relationships have been noted and invoked into heuristic models. The heuristic models have been developed and coded in MATLAB 7.0. The purpose of the

heuristic model is to produce similar results to both Semi-On and Off-line models, at a significantly reduced cost, i.e. time savings. The computational time it takes to solve the large scale optimization models is significantly more than it is to solve the heuristic model. The heuristic model can be seen in Chapter 4, with its corresponding code and flow charts in Appendix IV.

After running the models, as stated before, several relationships have been formed which can aid in the process of scheduling for these types of atmospheres. One consideration however, within the context of this thesis and as is adopted by the company with which we worked (the company wishes to remain anonymous), is the prioritization of jobs. A job receives a priority based on the amount of business that is generated from a single job. Thus the relationships are as follows:

• Relationship 1: *The closer the due dates are scheduled together towards the initial time of job release, causes a larger penalty value to be incurred.*

• Relationship 2: *The closer the due dates lie, and the closer they are scheduled towards the initial time of job release, the larger the percentage of daily job lateness.*

• Relationship 3: *The more experience a worker has, and the more experienced workers that are present, cause significant reductions in the penalty value of the system.*

• Relationship 4: *When experienced workers are present in a mixed experience environment, overall worker cost will decrease as experienced workers can accept more jobs.*

• Relationship 5: *Job priority level depicts the jobs positioning within the given schedule*

• Relationship 6: *The mixture of processing units of jobs directly affects the value of penalty and job delay.*

• Relationship 7: *As the mixture job processing units increase, the cost to processes those jobs also increases.*

Relationships 1 and 2 stem from due date constraints, thus it is imperative to have a proper balance of job due dates. As was seen in Chapter 5, when the due dates all fell within a certain period of each other, there was a significant increase in job tardiness due to the lack of resources available for processing. Hence, proper scheduling of line processing should be invoked to allow for a balanced maintenance period.

Relationships 3 and 4 stem from the issues of worker experience. It was noticed, in chapter 4, that on average, the costs to run the entire system decreased when worker experience was accounted for. Although the average cost per worker has increased, the abilities of experienced workers can significantly outweigh those of inexperienced workers, as was also seen by Stratman et al. (2004). In Chapter 2, the benefits of worker learning and experience were discussed, and within Chapter 5, these benefits were seen in the reduction of job tardiness, and overall worker wage decreases.

Relationship 5 was seen in tests regarding priority level adjustments. Several tests were invoked to determine if any relationships existed between job priority and the overall cost of the system. The main benefit of job priority is that it allows the system to process these jobs sooner, rather than later. As mentioned before, job priority is based on the amount of revenues, i.e. business, that a single job can generate. Thus higher priority jobs will in turn produce more revenues, thus should be scheduled early on to ensure the inflow of said revenues.

Lastly, relationships 6 and 7 stem from the processing times of jobs. By equating the problem at hand to the classical knapsack problem, the duty of the schedule is to try

and find the optimal selection of jobs to place in each knapsack, i.e. each worker. The size of the job can be equated to the size of the item needed to be placed within each knapsack. Hence the sizes of jobs, with relation to the weight of priority for each job will dictate the size of penalty to be incurred, the number of jobs that will be late, and the cost of resources needed to process the jobs.

From these results it is clear that the problem at hand is fairly complex in nature. If equating the problem to a knapsack problem, as workers progress through their careers, the subsequent knapsack will increase in size, thus increasing the complexity of the problem. As workers can not increase in workload indefinitely, the models hold more true to start up type operations. However, most industries disregard learning experience of workers, which can begin to prove inefficient over time.

6.2 Contributions and Future Work

6.2.1 Contributions

Scheduling problems are not new, and many literatures exist in this area. The problem at hand is based off of a real world case, within a manufacturing maintenance area. This thesis was able to provide, through its models, a method to re-create the problem instances in mathematical notation. Most papers deal with general case instances, whereas this thesis was able to work with real problem data, as would be found when working in industry.

Another emerging area in scheduling is on-line scheduling as was discussed in Chapter 2. This thesis was able to produce a semi-on-line case, again based off of real world influence. Due to the company's wish to remain anonymous, certain information can not be revealed, but the problem instance fully mimics the real world scenario. No specific model has been formulated as to the best of the authors' knowledge within this type of setting.

As it is very hard to solve MINLP problems, especially when a significant portion of the variables are binary variables, a heuristic model has been developed and coded within MATLAB 7.0. The heuristic method is able to produce good solutions, within a reasonable time, as compared to the optimization models. The heuristic method shares similar traits to the genetic algorithm, as it utilizes parent-children stems to ensure good pairings of jobs. The heuristic is aimed at the real world case but can be extended into many other areas where job scheduling is required.

No other previous work has invoked worker experience as this thesis has. Classical learning theories relate only to repetitive jobs, whereas this thesis deals with the constant use of skills. As workers hone their skills, their efficiency will increase, as is seen within the industrial problem. To encompass this issue, learning has been related to experience gained by working, thus increasing the ability to process more jobs in a given time. As is seen within the real world case, as workers proceed through their careers, wage incentives, i.e. pay raises, are invoked. Thus worker learning is also related to the workers pay scale, which is also involved within the model as can be seen in chapter 3. To the best of the authors' knowledge, no previous models have been created such as the models presented within this thesis.

Several models have been run, with varying parameters. All tests have been solved successfully in Lingo 9.0, thus proving the worth of the models. Both off-line and semi-on-line models were tested rigorously to ensure their flexibility in receiving several problem types.

Lastly, besides the direct research produced within this thesis, the results found in Chapter 5 provide valuable insight into real world scheduling problems. The relationships formed by running several models can now be applied to several types of industry, as they can be adopted as general guidelines.

6.2.2 Future Work

Based on the mathematical and heuristic models presented, the solution approach, and the results produced within this thesis, future work that has not been covered in this thesis is discussed in this section.

As real world problems can be extremely complex to model and solve, as was seen in the MINLP models presented in this thesis, heuristic models are an excellent alternative. The heuristic models presented in this paper produce good solutions but can be improved to reduce the gap between the solutions found in the optimization model and the heuristic solution.

This thesis utilizes all in house repairs of jobs, but it may sometimes be advantageous to outsource certain large jobs. Thus, another area of interest could be to determine whether or not outsourcing could benefit this type of environment, and if so what are the criteria to initiate the outsourced material.

Lingo 9.0 is a good solver, but better commercial solvers exist. A study could be done to see how results may differ among various solvers, such as solution time, scheduling policies, worker assignments, etc. A quantitative analysis of these various solvers would be interesting to review.

As this thesis deals with the ability for workers to improve through learning and experience, various mixtures of experienced workers could be researched. This thesis deals mainly with the benefits of the inclusion of worker learning and experience into a mathematical model. Work for future researchers could include determining actual learning rates found within various types of industry. Another area of interest could be determining the optimal mixture of induced versus autonomous learning and their benefits within the given work environment. Such work could find relationships between apprentice workers and full time permanent workers and the task assignments given to each.

Lastly, as little research has been done regarding learning and on-line scheduling, more research could be done within this area. The model presented in this thesis is a semi-on-line model, which only includes learning as a by product of the fully off-line model. Hence it may be interesting to try and utilize learning benefits in a fully on-line setting.

REFERENCES

Adler, P., Clark, K., 1991, Behind the learning curve - A sketch of the learning process, *Management Science*, **37**(3), p.267

Adzakpa, K., Adjallah, K., Yalaoui, F., 2004, On-line maintenance job scheduling and assignment to resources in distributed systems by heuristic-based optimization, *Journal of Intelligent Manufacturing*, **15**, p.131

Allwood, J., Lee, W., 2004, The impact of job rotation on problem solving skills, *International Journal of Production Research*, **42**(5), p.865

Amor, J., 2002, Scheduling programs with repetative projects using composite learning curve approximations, *Project Management Journal*, **33**(3), p.16

Amor, J., Teplitz, C., 1998, An efficient approximation for project composite learning curves, *Project management journal*, **29**(3), p.28

Arditi, D., Tokdemir, O., Suh, K., 2001, Effects of learning on line of balance scheduling, *International Journal of Project Management*, **19**, p.265

Artigues, C., Roubellat, F., 2001, A Petri net model and a general method for on and off-line multi-resource shop floor scheduling with setup times, *International Journal of Production Economics*, **74**, p.63

Bailey, C., 1989, Forgetting and the learning curve - A laboratory study, *Management Science*,**35**(3), p.340

Biskup, D., Simons, D., 2004, Common due date scheduling with autonomous and induce learning, *European Journal of Operational Research*, **159**, p. 606

Bostrom, L., Lassen, L., 2006, Unraveling learning, learning styles, learning stratagies and meta-cognition, *Education and Training*,**48**(2/3), p.178

Chambers, S., Johnston, R., 2000, Experience curves in services - Macro and micro level approaches, *MCB UP Limited (MCB)*

Chen, W., Liao, C., 2005, Scheduling with different maintenance policies in a textile company, *Journal of Quality in Maintenance Engineering*, **11**(1), p.43

Cheng, T., Ng, C., Kotov, V., 2006, A new algorithm for online uniform-machine scheduling to minimize the makespan, *Information Processing Letters*, **99**, p.102

Croce, F., Gupta, J., Tadei, R., 2000, Minimizing tardy jobs in a flowshop with common due date, *European Journal of Operational Research*, **120**, p. 375

Dorroh, J., Gulledge, T., Womer, N., Investment in knowledge - A generalization of learning by experience, *Management Science*, **40**(8), p.947

Finch, B., Luebbe, R., 1991, Risk associated with learning curve estimates, *Production and Inventory management Journal,* **32**(3), p.73

Fortin, D., Tsevendorj, I., 2004, Global optimization and multi knapsack: A percolation algorithm, *European Journal of Operational Research*, **154**, p. 46

Gambosi, G., Nicosia, G., 2000, On-line scheduling with setup costs, *Information Processing Letters*, **73**, p.61

Geiger, C., Uzsoy, R., Aytug, H., 2006, Rapid modeling and discovery of priority dispatching rules - An autonomous learning approach, *Journal of Scheduling,* **9**, p. 7

Jaber, M., Bonney, M., 1996, Production breaks and the learning curve - the forgetting phenomenom, *Applied Math Modelling*, **20**, p.162

Jaber, M., Bonney, M., 1997, A comparative study of learning curves with forgetting, *Applied Mathematical Modelling*, **21**, p.523

Jaber, M., Kher, H., 2004, Variant versus invariant time to total forgetting the learn forget model revisited, *Computers & Industrial Engineering*, **46**, p.697

Jaber, M., Kher, H., Davis, D., 2003, Countering forgetting through training and deployment, *International Journal of Production Economics*, **85**, p.33

Jaber, M., Sikstrom, S., 2004, A numerical comparison of three potential learning and forgetting models, *International Journal of Production Economics*, **92**, p.281

Kubzin, M., Strusevich, V., 2006, Planning Machine Maintenance in a Two machine shop scheduling, *Operations Research*, **54**(4), p.789

Lam, K., Lee, D., Hu, T., 2001, Understanding the effect of the learning forgetting phenomenon to duration of the projects construction, *Journal of Project Management*, **19**, p.411

Lee, C., Lin, C., 2001, Single-machine scheduling with maintenance and repair rate-modifying activities, *European Journal of Operational Research*, **135**, p.493

Li, R., Huang, H., 2007, Improved algorithm for a generalized on-line scheduling problem on identical machines, *European Journal of Operational Research*, **176**, p.643

Meeter, M., Murre, J., Janssen, S., 2005, Remembering the news - Modeling retention data from a study with 14000 participants, *Memory & Cognition*, **33**(5), p.793

Megow, N., Uetz, M., Vredeveld, T., 2006, Models and Algorithms for Stochastic Online Scheduling, *Mathematics of Operations Research*, **31**(3), p.513

M'Hallah, R., Bulfin, R., 2005, Minimizing the weighted number of tardy jobs on parallel processors, *European Journal of Operational Research*, **160**, p. 471

M'Hallah, R., Bulfin, R., 2007, Minimizing the weighted number of tardy jobs on a single machine with release dates, *European Journal of Operational Research*, **176**, p. 727

Mosheiov, G., Sidney, J., 2003, Scheduling with general job-dependant learning curves, *European Journal of Operational Research*, **147**, p.665

Mosheiov, G., Sidney, J., 2004, Note on scheduling with general learning curves to minimize number of tardy jobs, *Journal of the Operational Research Society,* **56**, p.110

Naroska, E., Schwiegelshohn, U., 2002, On an on-line scheduling problem for parallel jobs, *Information Processing Letters*, **81**, p.297

Nembhard, D., Osothsilp, N., 2001, An Empirical Comparison of Forgetting Models, *IEEE Transactions on Engineering Management*, **48**(3), p.283

Nembhard, D., Uzumeri, M., 2000, Experiential learning and forgetting for manual and cognitive tasks, *International Journal of Industrial Ergonomics*, **25**, p. 315

Pazzani, M., 1993, Learning causal patterns - making a transition from data-driven to theory-driven learning, *Machine Learning*, **11**, p.173

Pratsini, E., 2000, The capacitated dynamic lot size problem with variable technology, *Computers and Industrial engineering,* **38**, p.493

Pratsini, E., Camm, J., Raturi, A., 1993, Effect of process learning on manufacturing schedules, *Computers and Operations Research,* **20**(1), p.15

Rabinowitz, G., Goren, S., Mehrez, A., 2000, Scheduling two machines that require multiple types of maintenance for a single operation, *European Journal of Operational Research*, **127**, p.546

Renkema, A., 2006, Individual learning accounts - a strategy for life long learning, *Journal of Workplace Learning,* **18**(6), p.384

Ruiz, R., Diaz, J., Maroto, C., 2007, Considering scheduling and preventive maintenance in the flowshop sequencing problem, *Computers and Operations Research*, **34**, p. 3314

Serel, D., Dada, M., Moskowitz, H., Plante, R., 2003, Investing in quality under autonomous and induced learning, *IIE Transactions,***35**, p.545

Stark, R., Gruber, H., Renkl, A., Mandl, H., 1998, Instructional effects in complex learning - do objective and subjective learning outcomes converge?, *Learning and Instruction*, **8**(2), p.117

Stratman, J., Roth, A., Gilland, W., 2004, The deployment of temporary production workers in assembly operations: a case study of the hidden costs of learning and forgetting, *Journal of Operations Management,* **21**, p.689

Tan, Z., Cao, S., 2006, Semi-online Machine Covering on Two Uniform Machines with Known Total Size, *Journal of Computing*, **78**, p.369

Tyler, K., 2000, Focus on Training, *HRMagazine*, **45**(5), p. 94

Woeginger, G., 1993, A new on-line scheduling heuristic, *European Journal of Operational Research*, **71**, p.463

Weaver, P., 2006, "A BRIEF HISTORY OF SCHEDULIN - BACK TO THE FUTURE". Available online : http://www.pmforum.org/library/papers/2006/A_Brief_History_of_ Scheduling.pdf. (Downloaded September 13, 2007)

Appendix I: Simplified Preventative Maintenance Model

The original model took into account jobs that could be processed on multiple shifts. Due to this ability, jobs could be broken and processed as deemed fit, which allowed for an extremely large number of solutions. To simplify this ability, a new model, which only allows for jobs to be processed as full jobs, is modeled. The objective function is very similar to the previous model; the only change is within the third section which utilizes the variable Q_{ijkr} instead of PQ_{kr}. As there are no partial jobs anymore, this variable can be eliminated. Constraints 1 – 5 remain the same. Constraint 6 from the previous formulation dealt with the partial jobs, so a simple conversion is used bringing the partial processing time to the full processing time. Constraints 7, 8, 10, and 11 from the previous formulation can be omitted as the all deal with partial job completion. Constraint 9 from the previous model is now constraint 7 and has changed to dictate that all jobs must be processed only once. Constraint 12 from the previous model is now constraint 8 and has changed to incorporate Q_{ijkr} instead of PQ_{kr}. Constraints 13 and 14 from the prior model are now constraints 9 and 10 and have not changed. The model produces similar results and decreases the complexity of the model immensely. The model is now as follows:

$$MIN\ Z = \sum_{I}\sum_{j}\sum_{K}(C_i * x_{ijk}) + \sum_{i}\sum_{j}\sum_{k}\sum_{q}(CI_{qik} * x_{ijk}) + \sum_{i}\sum_{j}\sum_{k}\sum_{r}(Q_{ijkr} * PRr_r * Pc * S_r)$$

s.t.

$$\sum_{j} x_{ijk} \leq 1 \qquad \forall i,k \tag{1}$$

$$\sum_{i} x_{ijk} \leq \propto \qquad \forall j,k \tag{2}$$

$$T_{ik} = \sum_{j}\sum_{m=1}^{k} x_{ijm} \qquad \forall i,k \tag{3}$$

$$W_{ijk} = \gamma * T_{ik}^{l_i} * x_{ijk} \qquad \forall i,j,k \tag{4}$$

$$\sum_{r} Q_{ijkr} \leq \delta * x_{ijk} \qquad \forall i,j,k \tag{5}$$

$$\sum_{r} Q_{ijkr} * P_r \leq W_{ijk} \qquad \forall i,j,k \tag{6}$$

$$\sum_{i}\sum_{j}\sum_{k} Q_{ijkr} = 1 \qquad \forall r \tag{7}$$

$$Q_{ijkr} * k \leq D_r + S_r \qquad \forall i,j,k,r \tag{8}$$

$$CI_{qik} = \beta * y_{qi} \qquad \forall q,i \tag{9}$$

$$TI_q \geq T_{ik} * (1 - y_{qi}) \qquad \forall q,i,k \tag{10}$$

$x_{ijk}, Q_{ijkr}, y_{qi} = 0,1$
$T_{ik} \in Z^+$
$CI_{qik}, S_r, W_{ijk} \geq 0$

Appendix II: Simplified On-line Maintenance Model

The new model does not break up jobs; rather it finds an optimal mixture of jobs to process. Only a few differences are present. Constraint 2 changes by utilizing the variable P_r rather than $PC_{j\,k\,r}$ as it must assign the full job hours if job r is to be assigned. Constraint 3 from the previous model can be negated as it dealt with partial jobs not being processed on the same shift. Constraints 4 from the original formulation can be omitted as it dictated that the partial job processing time must equal the whole. Constraint 6 from the original formulation can also be omitted as no partial jobs are being processed, hence no jobs will be assigned with a zero value. Constraint 4 of the new formulation relates to constraint 5 of the previous model. It however has changed as it now dictates that each job must be assigned only once. The objective function, constraints 1, and 4 of the new formulation have not changed. Constraint 4 of the new formulation relates to constraint 7 of the previous model. The model is now formulated as follows. The variable descriptions can be found in chapter 3.

$$MIN\ Z = \sum_i \sum_j \sum_k \sum_r (Q_{i\,j\,k\,r} * PRr_r * Pc * S_j)$$

s.t.

$$\sum_r Q_{i\,j\,k\,r} \leq \delta * x_{i\,j\,k} \qquad \forall\, i,j,k \tag{1}$$

$$\sum_r Q_{i\,j\,k\,r} * P_r \leq W_{i\,j\,k} \qquad \forall\, i,j,k \tag{2}$$

$$\sum_i \sum_j \sum_k Q_{i\,j\,k\,r} = 1 \qquad \forall\, r \tag{3}$$

$$Q_{i\,j\,k\,r} * (j + (k-1) * J) \leq D_j + S_j \qquad \forall\, j,k,r \tag{4}$$

$$Q_{i\,j\,k\,r} = 0,1$$
$$S_j \geq 0$$

Appendix III: Heuristic Example Problem

First Iteration - Two job combinations

	J2	J3	J4	J5	J6	J7	J8	J9	J10
J1	NF	NF	NF	(11,8)	NF	NF	NF	NF	NF
J2		(11,7)	NF	NF	NF	NF	NF	NF	NF
J3			(12,5)	NF	(14,6)	NF	(12,7)	(15,6)	(14,7)
J4				NF	NF	NF	NF	(11,5)	NF
J5					NF	NF	NF	NF	NF
J6						NF	NF	(13,6)	(12,7)
J7							NF	NF	NF
J8								(11,7)	NF
J9									(13,7)

First Iteration - Three job combinations

	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10
J1J5		NF	NF	NF		NF	NF	NF	NF	NF
J2J3	NF			NF	NF	NF	NF	NF	NF	NF
J3J4	NF	NF			NF	(18,8)	NF	NF	(19,8)	NF
J3J6	NF	NF		(18,8)	NF		(16,8)	NF	NF	NF
J3J8	NF	NF		NF	NF	NF	NF		NF	NF
J3J9	NF	NF		(19,8)	NF	NF	(16,8)	NF		NF
J3J10	NF	NF		NF	NF	NF	NF	NF	NF	
J4J9	NF	NF	(19,8)		NF	(17,8)	NF	NF		NF
J6J9	NF	NF	NF	(17,8)	NF		NF	NF		NF
J6J10	NF	NF	NF	NF	NF		NF	NF	NF	
J8J9	NF	NF	NF	NF	NF	NF	NF			NF
J9J10	NF	NF	NF	NF	NF	NF	NF	NF		

Second Iteration - Two job combinations

	J2	J5	J6	J7	J8	J10
J1	NF	(11,8)	NF	NF	NF	NF
J2		NF	NF	NF	NF	NF
J5			NF	NF	NF	NF
J6				NF	NF	(12,7)
J7					NF	NF
J8						NF

Second Iteration - Three job combinations

	J1	J2	J5	J6	J7	J8	J10
J1J5		NF		NF	NF	NF	NF
J6J10	NF	NF	(13,8)		NF	NF	

Third Iteration - Two job combinations

	J2	J7	J8
J1	NF	NF	NF
J2		NF	NF
J7			NF

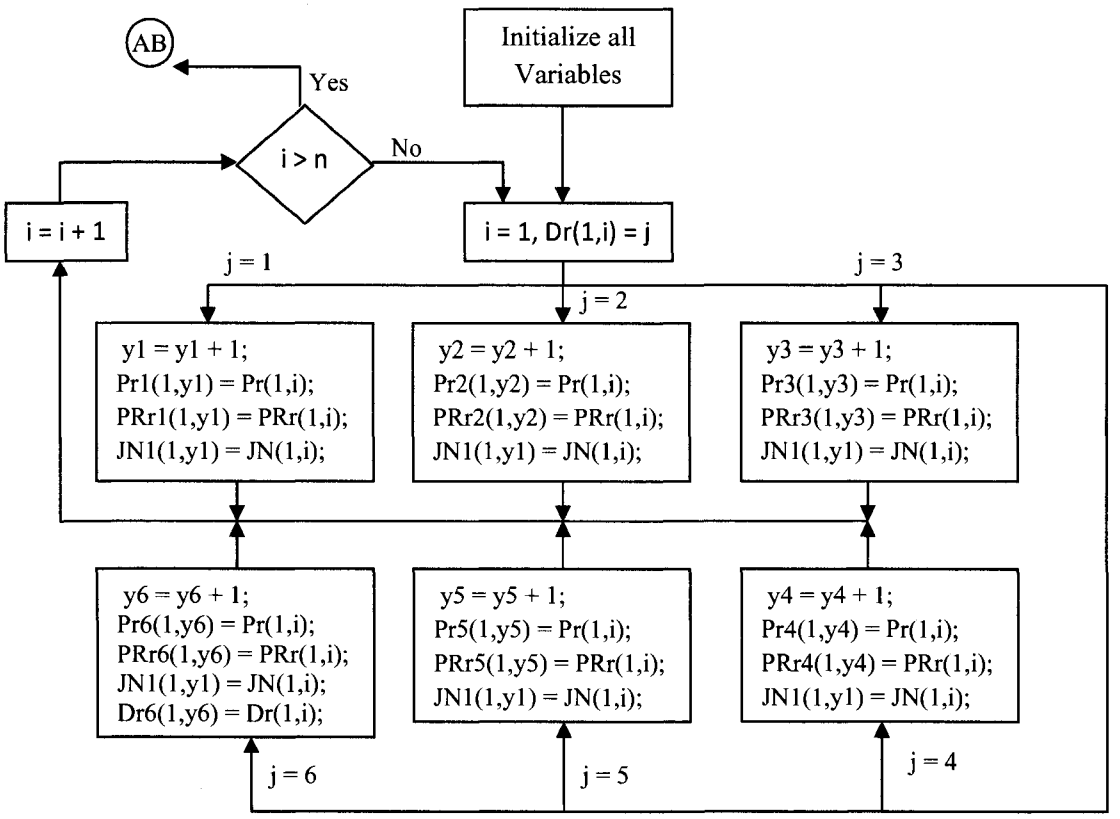
Fourth Iteration - Two job combinations

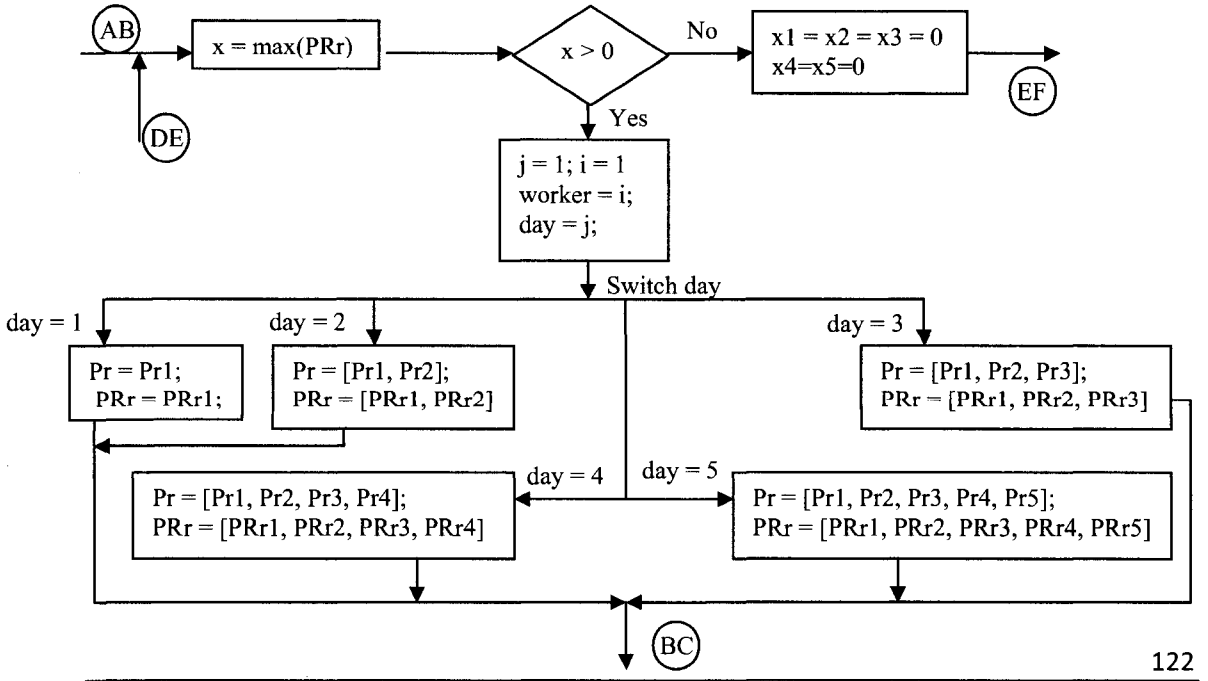
	J7	J8
J2	(5,6)	(7,8)
J7		(6,6)

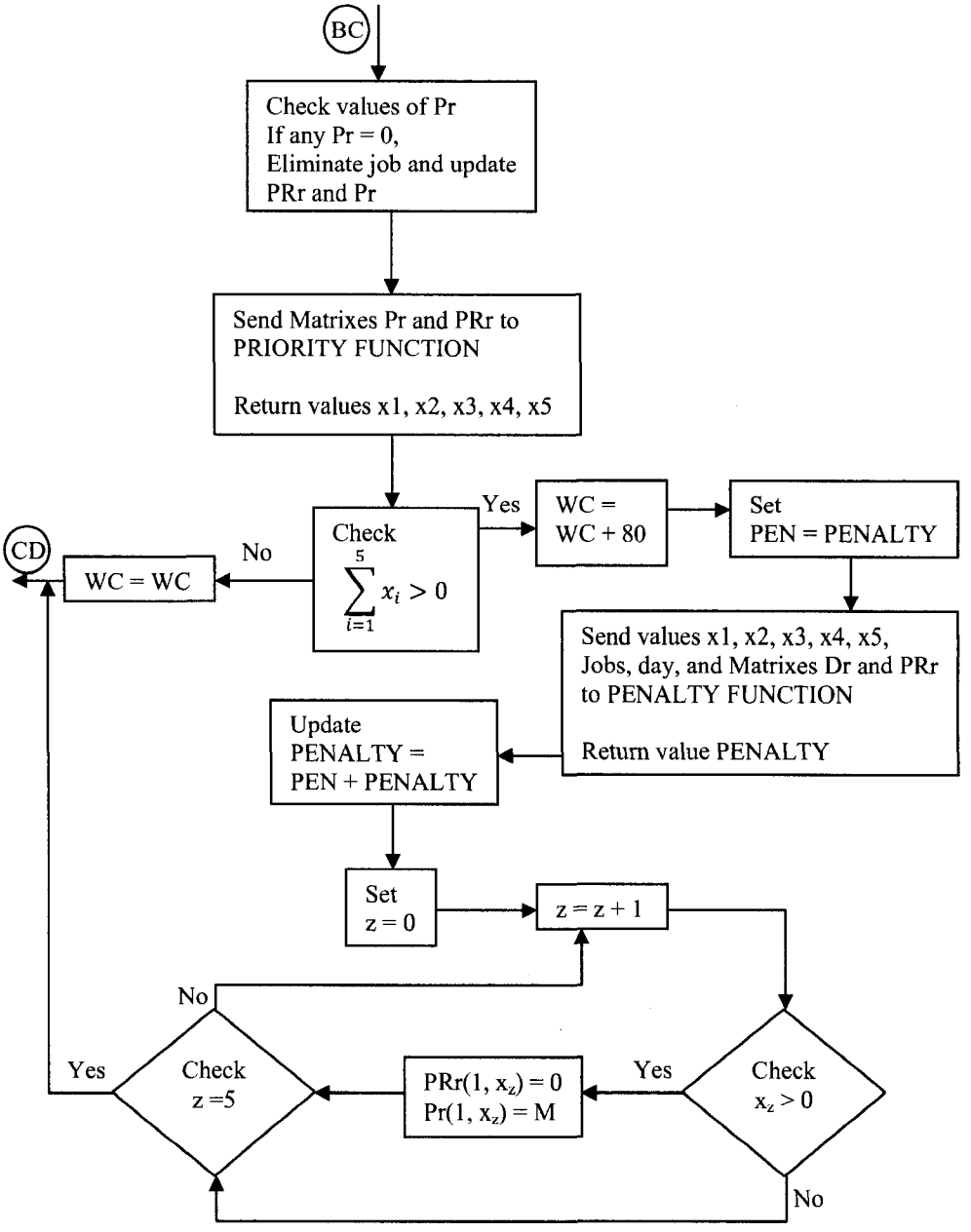
Fourth Iteration - Three job combinations

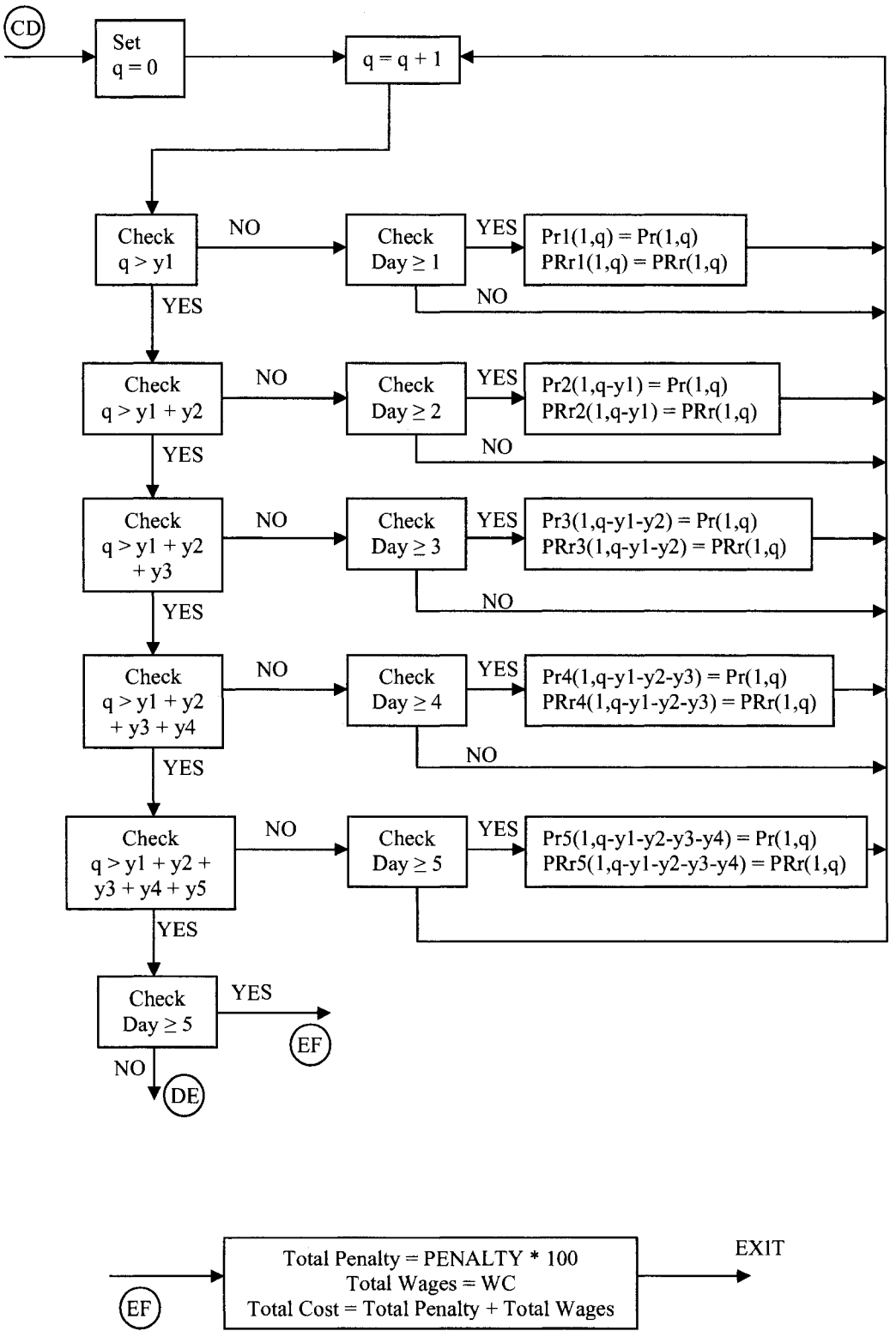
	J2	J7	J8
J2J7			NF
J2J8		NF	
J7J8	NF		

Appendix IV: Heuristic Flow Charts

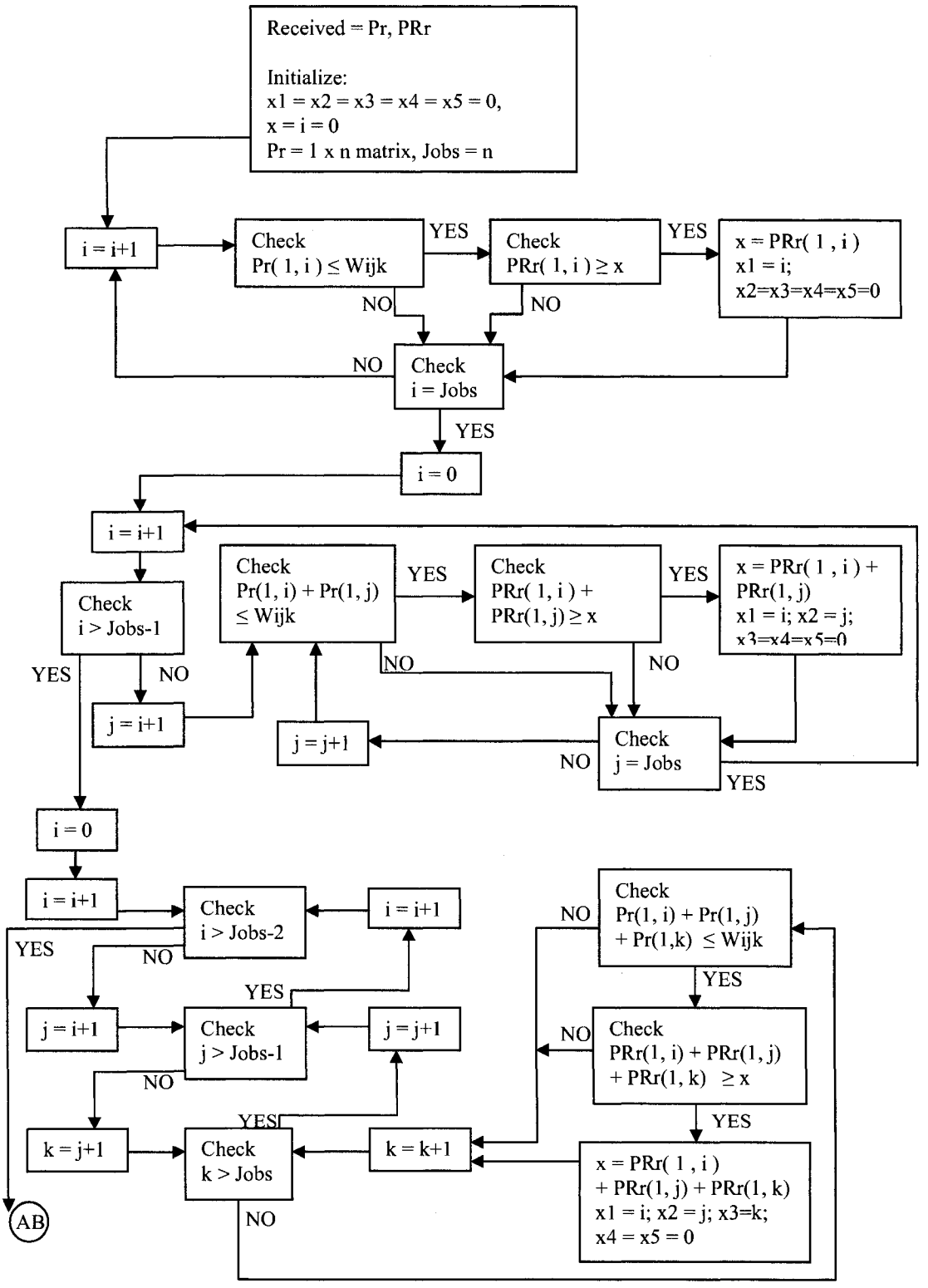


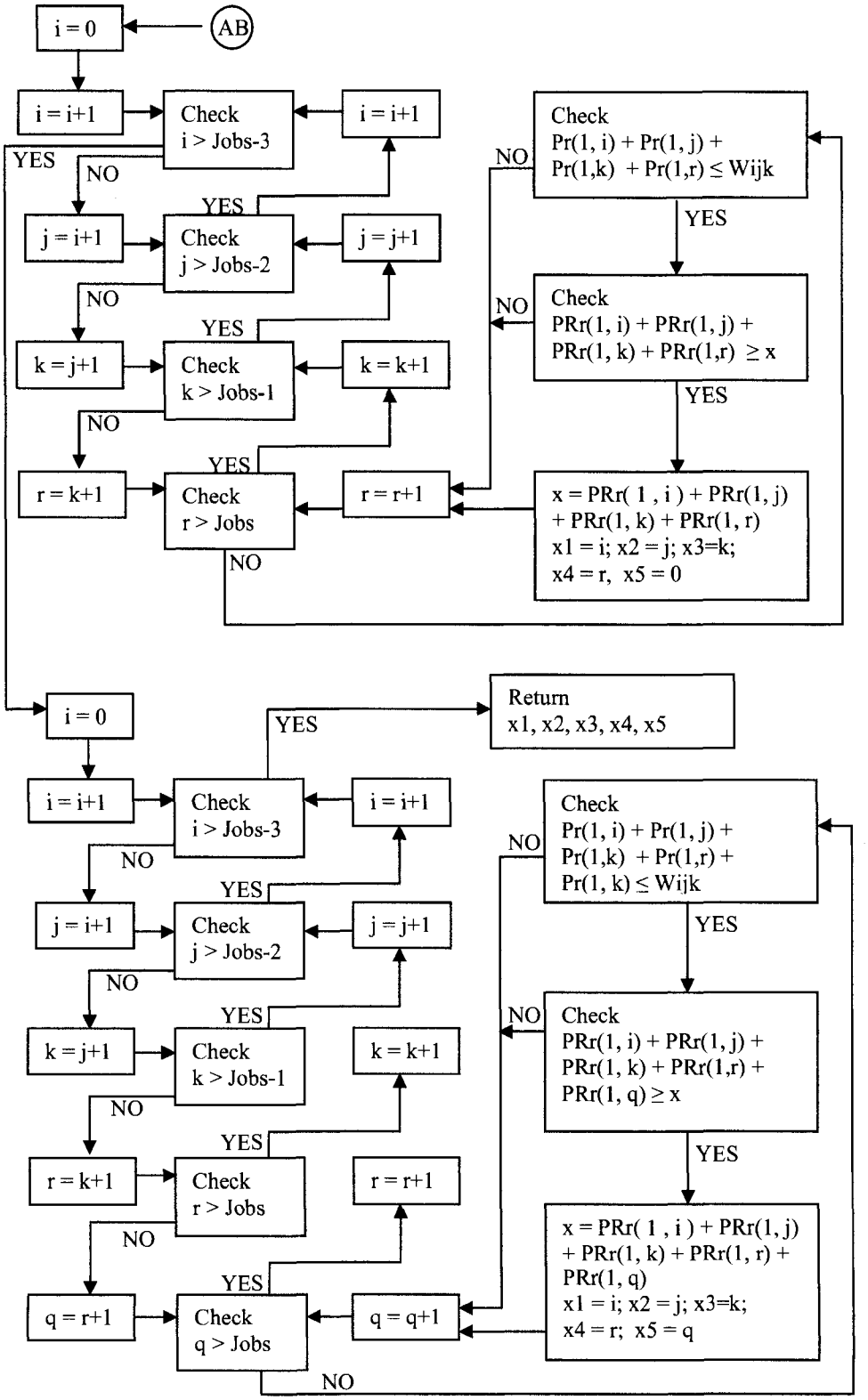



BC
Check values of Pr
If any Pr = 0,
Eliminate job and update
PRr and Pr
Send Matrixes Pr and PRr to
PRIORITY FUNCTION
Return values x1, x2, x3, x4, x5
Check
$\sum_{i=1}^{5} x_i > 0$
Yes
WC =
WC + 80
Set
PEN = PENALTY
Send values x1, x2, x3, x4, x5,
Jobs, day, and Matrixes Dr and PRr
to PENALTY FUNCTION
Return value PENALTY
Update
PENALTY =
PEN + PENALTY
No
WC = WC
CD
Set
z = 0
z = z + 1
Check
z =5
Yes
No
PRr(1, x_z) = 0
Pr(1, x_z) = M
Check
$x_z > 0$
Yes
No

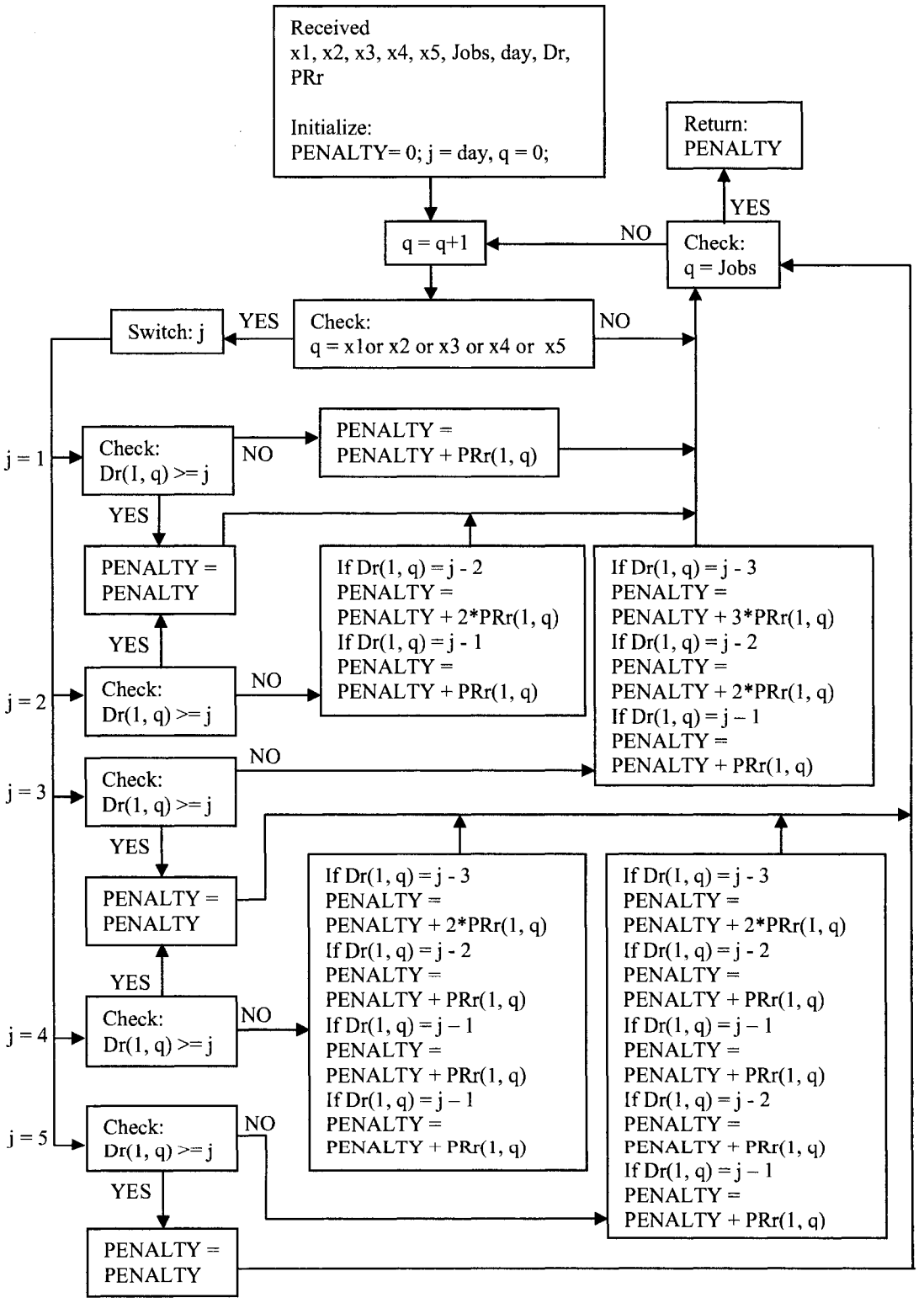

CD
Set
q = 0
q = q + 1
Check
q > y1
NO
Check
Day ≥ 1
YES
Pr1(1,q) = Pr(1,q)
PRr1(1,q) = PRr(1,q)
NO
YES
Check
q > y1 + y2
NO
Check
Day ≥ 2
YES
Pr2(1,q-y1) = Pr(1,q)
PRr2(1,q-y1) = PRr(1,q)
NO
YES
Check
q > y1 + y2
+ y3
NO
Check
Day ≥ 3
YES
Pr3(1,q-y1-y2) = Pr(1,q)
PRr3(1,q-y1-y2) = PRr(1,q)
NO
YES
Check
q > y1 + y2
+ y3 + y4
NO
Check
Day ≥ 4
YES
Pr4(1,q-y1-y2-y3) = Pr(1,q)
PRr4(1,q-y1-y2-y3) = PRr(1,q)
NO
YES
Check
q > y1 + y2 +
y3 + y4 + y5
NO
Check
Day ≥ 5
YES
Pr5(1,q-y1-y2-y3-y4) = Pr(1,q)
PRr5(1,q-y1-y2-y3-y4) = PRr(1,q)
YES
Check
Day ≥ 5
YES
EF
NO
DE
EF
Total Penalty = PENALTY * 100
Total Wages = WC
Total Cost = Total Penalty + Total Wages
EXIT

PRIORITY FUNCTION




i = 0
AB
i = i+1
Check
i > Jobs-3
i = i+1
YES
NO
j = i+1
Check
j > Jobs-2
j = j+1
NO
YES
k = j+1
Check
k > Jobs-1
k = k+1
NO
YES
r = k+1
Check
r > Jobs
r = r+1
NO
YES
Check
Pr(1, i) + Pr(1, j) +
Pr(1,k) + Pr(1,r) ≤ Wijk
NO
YES
Check
PRr(1, i) + PRr(1, j) +
PRr(1, k) + PRr(1,r) ≥ x
NO
YES
x = PRr(1 , i) + PRr(1, j)
+ PRr(1, k) + PRr(1, r)
x1 = i; x2 = j; x3=k;
x4 = r, x5 = 0
i = 0
YES
Return
x1, x2, x3, x4, x5
i = i+1
Check
i > Jobs-3
i = i+1
NO
YES
j = i+1
Check
j > Jobs-2
j = j+1
NO
YES
k = j+1
Check
k > Jobs-1
k = k+1
NO
YES
r = k+1
Check
r > Jobs
r = r+1
NO
YES
q = r+1
Check
q > Jobs
q = q+1
NO
YES
Check
Pr(1, i) + Pr(1, j) +
Pr(1,k) + Pr(1,r) +
Pr(1, k) ≤ Wijk
NO
YES
Check
PRr(1, i) + PRr(1, j) +
PRr(1, k) + PRr(1,r) +
PRr(1, q) ≥ x
NO
YES
x = PRr(1 , i) + PRr(1, j)
+ PRr(1, k) + PRr(1, r) +
PRr(1, q)
x1 = i; x2 = j; x3=k;
x4 = r; x5 = q

PENALTY FUNCTION



Appendix V: Lingo 9.0 Test Results and Problem Data

Test 1

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	2	2	1	2	2	1	200
1	1	1	6	1	4	1	0	0
1	1	1	10	1	7	2	0	0
1	1	1	11	1	6	1	0	0
1	1	1	19	1	10	4	0	0
1	1	2	20	1	3	6	1	300
2	1	1	7	1	10	8	0	0
2	1	2	12	1	3	7	1	300
2	1	2	18	1	2	1	1	200
3	1	3	3	1	1	4	2	200
3	1	1	4	1	10	1	0	0
3	1	1	8	1	5	5	0	0
3	1	2	13	1	3	5	1	300
3	1	1	15	1	10	2	0	0
3	1	2	16	1	2	2	1	200
4	1	1	1	1	7	5	0	0
4	1	3	5	1	3	8	2	600
4	1	2	9	1	3	1	1	300
4	1	1	14	1	9	3	0	0
4	1	2	17	1	5	7	1	500

Test 2

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	2	1	7	6	0	0
1	1	1	19	1	10	1	0	0
1	1	2	12	2	6	6	0	0
1	1	3	10	2	5	5	1	500
1	1	4	5	1	2	4	3	600
2	1	1	18	1	8	6	0	0
2	1	2	4	2	10	1	0	0
2	1	2	6	2	10	5	0	0
2	1	3	8	1	5	7	2	1000
2	1	4	13	2	2	7	2	400
3	1	1	15	1	9	5	0	0
3	1	1	20	1	5	2	0	0
3	1	2	7	2	4	4	0	0
3	1	2	17	2	8	4	0	0
3	1	3	14	2	4	7	1	400
3	1	4	3	1	3	7	3	900
4	1	1	9	1	7	8	0	0
4	1	2	11	2	7	7	0	0
4	1	3	16	2	6	5	1	600
4	1	4	1	1	1	8	3	300

Test 3

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	5	1	8	8	0	0
1	1	2	1	2	2	2	0	0
1	1	2	18	1	4	6	1	400
1	1	3	8	2	3	6	1	300
2	1	1	15	1	4	6	0	0
2	1	1	10	1	10	2	0	0
2	1	2	6	1	6	8	1	600
2	1	3	2	2	5	7	1	500
3	1	1	4	1	8	3	0	0
3	1	1	13	1	10	5	0	0
3	1	2	9	1	7	5	1	700
3	1	2	17	2	4	3	0	0
3	1	3	20	2	3	6	1	300
4	1	1	3	1	10	1	0	0
4	1	1	7	1	10	4	0	0
4	1	1	11	1	1	3	0	0
4	1	2	12	1	4	6	1	400
4	1	2	14	2	9	1	0	0
4	1	2	16	2	6	1	0	0
4	1	3	19	2	1	6	1	100

Test 4

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	2	1	4	1	0	0
1	1	1	3	3	8	6	0	0
1	1	1	13	3	6	1	0	0
1	1	2	6	3	3	2	0	0
1	1	2	7	2	2	3	0	0
1	1	2	10	4	5	3	0	0
1	1	3	5	4	7	5	0	0
1	1	3	14	5	4	3	0	0
1	1	4	11	4	4	8	0	0
1	1	5	12	5	8	8	0	0
2	1	1	4	1	5	2	0	0
2	1	1	16	2	1	4	0	0
2	1	2	17	3	3	4	0	0
2	1	2	18	3	8	4	0	0
2	1	3	19	3	3	8	0	0
2	1	4	15	5	2	2	0	0
2	1	5	20	5	7	7	0	0
4	1	4	8	5	1	8	0	0
4	1	5	1	5	5	5	0	0
4	1	5	9	5	2	3	0	0

Test 5NL

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	20	1	8	4.2	0	0
1	1	1	12	2	9	3.5	0	0
2	1	1	3	1	7	2.3	0	0
2	1	1	10	1	3	3.1	0	0
2	1	1	13	1	9	1.9	0	0
3	1	1	2	1	9	1.9	0	0
3	1	1	17	1	9	2.4	0	0
3	1	1	18	1	10	3.4	0	0
4	1	1	7	1	9	7.1	0	0
1	1	2	4	1	5	7.3	1	500
2	1	2	11	2	5	1.5	0	0
2	1	2	15	2	5	5.3	0	0
2	1	2	19	2	2	1.5	0	0
3	1	2	6	1	7	7.5	1	700
4	1	2	9	2	4	3.3	0	0
4	1	2	14	2	6	3.7	0	0
1	1	3	16	1	1	5.1	2	200
2	1	3	5	1	3	6.3	2	600
3	1	3	8	1	3	6.1	2	600
4	1	3	1	1	3	6.2	2	600

Test 5L

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	2	1	9	1.9	0	0
1	1	1	3	1	7	2.3	0	0
1	1	1	17	1	9	2.4	0	0
1	1	1	20	1	8	4.2	0	0
2	1	1	7	1	9	7.1	0	0
2	1	1	18	1	10	3.4	0	0
3	1	1	6	1	7	7.5	0	0
4	1	1	8	1	3	6.1	0	0
4	1	1	13	1	9	1.9	0	0
1	1	2	10	1	3	3.1	1	300
1	1	2	15	2	5	5.3	0	0
1	1	2	19	2	2	1.5	0	0
2	1	2	4	1	5	7.3	1	500
2	1	2	12	2	9	3.5	0	0
3	1	2	9	2	4	3.3	0	0
3	1	2	14	2	6	3.7	0	0
4	1	2	1	1	3	6.2	1	300
4	1	2	11	2	5	1.5	0	0
1	1	3	5	1	3	6.3	2	600
4	1	3	16	1	1	5.1	2	200

Test 6NL

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	10	1	6	6.1	0	0
2	1	1	9	1	5	6.1	0	0
2	1	1	7	2	3	1.7	0	0
3	1	1	8	1	9	1.3	0	0
3	1	1	19	1	6	5.8	0	0
4	1	1	1	1	9	5.2	0	0
4	1	1	2	1	10	2.3	0	0
1	1	2	16	1	9	4.6	1	900
1	1	2	13	2	7	3.6	0	0
2	1	2	11	2	7	4.5	0	0
2	1	2	15	2	4	3.6	0	0
3	1	2	5	2	7	2.7	0	0
3	1	2	14	2	9	5.7	0	0
4	1	2	17	2	10	5.2	0	0
1	1	3	18	2	4	5.9	1	400
2	1	3	4	2	4	6.3	1	400
3	1	3	12	1	2	7.2	2	400
4	1	3	6	2	1	3.5	1	100
4	1	3	20	2	1	4.4	1	100
2	1	4	3	1	2	7	3	600

Test 6L

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	1	1	9	5.2	0	0
1	1	1	19	1	6	5.8	0	0
2	1	1	2	1	10	2.3	0	0
2	1	1	6	2	1	3.5	0	0
2	1	1	16	1	9	4.6	0	0
3	1	1	7	2	3	1.7	0	0
3	1	1	9	1	5	6.1	0	0
4	1	1	8	1	9	1.3	0	0
4	1	1	10	1	6	6.1	0	0
1	1	2	17	2	10	5.2	0	0
1	1	2	18	2	4	5.9	0	0
2	1	2	4	2	4	6.3	0	0
2	1	2	15	2	4	3.6	0	0
3	1	2	5	2	7	2.7	0	0
3	1	2	14	2	9	5.7	0	0
4	1	2	11	2	7	4.5	0	0
4	1	2	13	2	7	3.6	0	0
2	1	3	12	1	2	7.2	2	400
3	1	3	20	2	1	4.4	1	100
4	1	3	3	1	2	7	2	400

Test 7NL

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	9	1	8	4.5	0	0
1	1	1	18	1	7	3.5	0	0
2	1	1	11	1	2	2.8	0	0
2	1	1	16	1	9	4.8	0	0
3	1	1	4	1	9	4.6	0	0
3	1	1	15	1	2	3.4	0	0
4	1	1	7	1	3	2.3	0	0
4	1	1	20	1	5	5.5	0	0
1	1	2	2	2	9	1.4	0	0
1	1	2	5	2	10	1.4	0	0
1	1	2	17	2	5	5.1	0	0
2	1	2	14	1	6	7.2	1	600
3	1	2	1	2	9	6.3	0	0
3	1	2	19	2	9	1.9	0	0
4	1	2	3	2	3	1.5	0	0
4	1	2	12	2	3	4	0	0
4	1	2	13	2	8	2.2	0	0
1	1	3	10	2	2	7.5	1	200
2	1	3	8	1	4	5.9	2	800
3	1	3	6	2	1	4.2	1	100

Test 7L

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	7	1	3	2.3	0	0
1	1	1	15	1	2	3.4	0	0
1	1	1	20	1	5	5.5	0	0
2	1	1	4	1	9	4.6	0	0
2	1	1	8	1	4	5.9	0	0
3	1	1	11	1	2	2.8	0	0
3	1	1	16	1	9	4.8	0	0
4	1	1	9	1	8	4.5	0	0
4	1	1	18	1	7	3.5	0	0
1	1	2	6	2	1	4.2	0	0
1	1	2	12	2	3	4	0	0
2	1	2	2	2	9	1.4	0	0
2	1	2	13	2	8	2.2	0	0
2	1	2	14	1	6	7.2	1	600
3	1	2	3	2	3	1.5	0	0
3	1	2	5	2	10	1.4	0	0
3	1	2	17	2	5	5.1	0	0
4	1	2	1	2	9	6.3	0	0
4	1	2	19	2	9	1.9	0	0
1	1	3	10	2	2	7.5	1	200

Test 8

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	1	1	3	1	0	0
1	1	1	3	3	4	7	0	0
2	1	1	8	1	2	7	0	0
3	1	1	6	1	4	3	0	0
3	1	1	13	1	3	5	0	0
4	1	1	5	1	3	1	0	0
4	1	1	7	1	4	4	0	0
4	1	1	10	2	2	3	0	0
1	1	2	14	3	2	8	0	0
2	1	2	11	2	1	5	0	0
2	1	2	15	3	4	3	0	0
3	1	2	16	2	2	5	0	0
3	1	2	19	3	3	3	0	0
4	1	2	2	2	3	1	0	0
4	1	2	17	2	2	7	0	0
1	1	3	18	3	4	8	0	0
2	1	3	9	3	2	7	0	0
3	1	3	20	3	3	6	0	0
4	1	3	4	3	2	6	0	0
4	1	3	12	3	2	1	0	0

Test 9

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	2	1	6	5	0	0
1	1	1	6	1	7	2	0	0
1	1	1	10	1	6	1	0	0
2	1	1	5	1	4	8	0	0
3	1	1	3	1	3	2	0	0
3	1	1	13	1	5	6	0	0
4	1	1	7	1	7	2	0	0
4	1	1	17	1	4	6	0	0
1	1	2	4	3	5	7	0	0
2	1	2	8	2	6	7	0	0
3	1	2	15	2	5	8	0	0
4	1	2	9	2	4	3	0	0
4	1	2	12	2	5	1	0	0
4	1	2	19	2	5	2	0	0
1	1	3	20	3	6	8	0	0
2	1	3	14	3	7	7	0	0
3	1	3	1	3	3	4	0	0
4	1	3	11	3	3	2	0	0
4	1	3	16	3	4	1	0	0
4	1	3	18	3	5	3	0	0

Test 10

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	1	2	7	6	0	0
4	1	3	2	3	8	7	0	0
1	1	2	3	2	7	6	0	0
1	1	3	4	3	7	8	0	0
4	1	4	5	3	7	8	1	700
2	1	3	6	3	7	4	0	0
2	1	1	7	2	10	7	0	0
4	1	2	8	2	10	1	0	0
4	1	2	9	2	8	7	0	0
4	1	1	10	1	10	5	0	0
3	1	3	11	3	7	4	0	0
4	1	3	12	3	8	1	0	0
3	1	1	13	2	10	6	0	0
3	1	2	14	2	8	3	0	0
1	1	1	15	1	7	2	0	0
3	1	3	16	2	7	4	1	700
4	1	1	17	1	10	3	0	0
3	1	2	18	2	10	4	0	0
2	1	2	19	2	7	7	0	0
2	1	3	20	3	9	3	0	0

Test 11

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	7	1	3	7	0	0
1	1	1	10	1	1	1	0	0
2	1	1	16	1	4	8	0	0
3	1	1	2	1	5	2	0	0
3	1	1	18	1	2	6	0	0
4	1	1	9	1	1	5	0	0
1	1	2	13	2	2	7	0	0
2	1	2	12	2	5	3	0	0
2	1	2	17	2	7	4	0	0
3	1	2	4	2	3	4	0	0
3	1	2	6	2	6	3	0	0
4	1	2	3	1	1	7	1	100
1	1	3	5	3	4	4	0	0
1	1	3	20	3	3	4	0	0
2	1	3	8	3	3	7	0	0
3	1	3	11	3	1	2	0	0
3	1	3	14	3	6	6	0	0
4	1	3	1	3	5	1	0	0
4	1	3	19	3	4	7	0	0
4	1	4	15	2	1	7	2	200

Test 12

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	4	1	4	6	0	0
1	1	1	5	1	5	2	0	0
2	1	1	6	1	7	5	0	0
2	1	1	10	1	4	3	0	0
3	1	1	1	1	6	7	0	0
4	1	1	8	1	10	8	0	0
1	1	2	14	2	8	4	0	0
2	1	2	2	2	9	3	0	0
2	1	2	7	1	4	4	1	400
3	1	2	17	2	10	8	0	0
4	1	2	3	2	8	6	0	0
4	1	2	13	2	7	1	0	0
1	1	3	18	3	10	5	0	0
2	1	3	11	3	4	4	0	0
2	1	3	19	3	4	3	0	0
3	1	3	9	3	5	4	0	0
3	1	3	12	3	6	1	0	0
4	1	3	15	3	6	3	0	0
4	1	3	16	3	10	1	0	0
4	1	3	20	3	10	3	0	0

Test 13

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	7	3	1	5	0	0
1	1	1	13	1	10	3	0	0
2	1	1	9	1	3	3	0	0
2	1	1	12	1	9	2	0	0
2	1	1	19	1	7	3	0	0
3	1	1	4	1	3	8	0	0
4	1	1	11	1	10	4	0	0
4	1	1	18	1	7	4	0	0
1	1	2	14	2	10	3	0	0
1	1	2	20	3	8	5	0	0
2	1	2	6	2	1	7	0	0
2	1	2	17	2	7	1	0	0
3	1	2	3	2	2	5	0	0
4	1	2	10	1	1	8	1	100
1	1	3	5	3	1	6	0	0
1	1	3	8	3	1	2	0	0
2	1	3	2	3	2	8	0	0
3	1	3	16	3	10	4	0	0
4	1	3	1	3	2	6	0	0
4	1	3	15	3	9	1	0	0

Test 14

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	2	1	8	4	0	0
1	1	1	5	1	8	2	0	0
2	1	1	8	1	6	4	0	0
2	1	1	13	1	7	4	0	0
3	1	1	11	2	4	5	0	0
3	1	1	20	2	5	3	0	0
4	1	1	18	3	10	2	0	0
4	1	1	19	1	6	6	0	0
1	1	2	9	2	6	6	0	0
2	1	2	4	3	3	6	0	0
2	1	2	6	3	7	1	0	0
2	1	2	17	2	9	1	0	0
3	1	2	12	2	8	6	0	0
4	1	2	14	3	2	6	0	0
1	1	3	7	3	7	7	0	0
2	1	3	3	3	9	6	0	0
3	1	3	1	3	3	4	0	0
3	1	3	10	3	8	4	0	0
4	1	3	15	3	9	2	0	0
4	1	3	16	3	8	6	0	0

Test 15

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	2	1	4	3	0	0
1	1	1	3	2	9	1	0	0
1	1	1	5	1	9	2	0	0
1	1	1	7	2	3	1	0	0
2	1	1	4	1	2	3	0	0
2	1	1	6	1	3	3	0	0
2	1	1	15	1	6	1	0	0
2	1	1	17	1	9	1	0	0
3	1	1	10	1	1	2	0	0
3	1	1	12	1	5	3	0	0
3	1	1	18	1	2	2	0	0
4	1	1	1	1	3	3	0	0
4	1	1	8	1	6	3	0	0
1	1	2	9	2	10	1	0	0
1	1	2	11	2	1	2	0	0
1	1	2	13	2	1	2	0	0
1	1	2	16	2	6	1	0	0
2	1	2	19	2	3	2	0	0
2	1	2	20	2	8	1	0	0
4	1	2	14	2	9	3	0	0

Test 16

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	5	1	9	4	0	0
1	1	1	17	1	9	3	0	0
2	1	1	6	1	3	5	0	0
2	1	1	12	1	5	3	0	0
3	1	1	1	1	3	5	0	0
3	1	1	15	1	6	3	0	0
4	1	1	2	1	4	5	0	0
4	1	1	8	1	6	3	0	0
1	1	2	14	2	9	5	0	0
1	1	2	19	2	3	3	0	0
2	1	2	7	2	3	3	0	0
2	1	2	16	2	6	5	0	0
3	1	2	9	2	10	4	0	0
3	1	2	20	2	8	4	0	0
4	1	2	3	2	9	4	0	0
4	1	2	4	1	2	3	1	200
1	1	3	11	2	1	3	1	100
2	1	3	10	1	1	5	2	200
3	1	3	13	2	1	5	1	100
4	1	3	18	1	2	5	2	400

Test 17

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	5	1	9	5	0	0
2	1	1	8	1	6	8	0	0
3	1	1	15	1	6	5	0	0
4	1	1	17	1	9	6	0	0
1	1	2	3	2	9	7	0	0
2	1	2	14	2	9	6	0	0
3	1	2	9	2	10	5	0	0
4	1	2	20	2	8	5	0	0
1	1	3	6	1	3	7	2	600
2	1	3	12	1	5	8	2	1000
3	1	3	16	2	6	7	1	600
4	1	3	2	1	4	6	2	800
1	1	4	18	1	2	6	3	600
2	1	4	19	2	3	8	2	600
3	1	4	7	2	3	5	2	600
4	1	4	1	1	3	8	3	900
1	1	5	11	2	1	6	3	300
2	1	5	10	1	1	6	4	400
3	1	5	4	1	2	7	4	800
4	1	5	13	2	1	6	3	300

Test 18

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	2	1	4	5	0	0
1	1	1	4	1	2	3	0	0
2	1	1	15	1	6	4	0	0
2	1	1	18	1	2	4	0	0
3	1	1	1	1	3	1	0	0
3	1	1	5	1	9	2	0	0
3	1	1	17	1	9	5	0	0
4	1	1	6	1	3	3	0	0
4	1	1	8	1	6	1	0	0
4	1	1	12	1	5	4	0	0
1	1	2	7	2	3	2	0	0
1	1	2	9	2	10	5	0	0
1	1	2	14	2	9	1	0	0
2	1	2	3	2	9	1	0	0
2	1	2	19	2	3	4	0	0
3	1	2	13	2	1	4	0	0
3	1	2	20	2	8	4	0	0
4	1	2	16	2	6	5	0	0
2	1	3	11	2	1	5	1	100
4	1	3	10	1	1	5	2	200

Test 19

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	8	1	6	6	0	0
2	1	1	5	1	9	4	0	0
2	1	1	12	1	5	4	0	0
3	1	1	15	1	6	7	0	0
4	1	1	6	1	3	5	0	0
4	1	1	17	1	9	3	0	0
1	1	2	9	2	10	6	0	0
2	1	2	3	2	9	5	0	0
2	1	2	16	2	6	3	0	0
3	1	2	20	2	8	8	0	0
4	1	2	2	1	4	4	1	400
4	1	2	14	2	9	4	0	0
1	1	3	19	2	3	8	1	300
2	1	3	4	1	2	5	2	400
2	1	3	11	2	1	3	1	100
3	1	3	1	1	3	8	2	600
4	1	3	7	2	3	6	1	300
1	1	4	18	1	2	8	3	600
2	1	4	13	2	1	7	2	200
3	1	4	10	1	1	6	3	300

Test 20

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	15	1	6	4	0	0
1	1	1	17	1	9	4	0	0
2	1	1	1	1	3	1	0	0
2	1	1	5	1	9	5	0	0
2	1	1	8	1	6	1	0	0
2	1	1	10	1	1	1	0	0
3	1	1	2	1	4	3	0	0
3	1	1	6	1	3	5	0	0
4	1	1	12	1	5	7	0	0
1	1	2	18	1	2	5	1	200
1	1	2	19	2	3	3	0	0
2	1	2	9	2	10	7	0	0
3	1	2	3	2	9	4	0	0
3	1	2	7	2	3	2	0	0
3	1	2	20	2	8	2	0	0
4	1	2	14	2	9	3	0	0
4	1	2	16	2	6	5	0	0
1	1	3	4	1	2	6	2	400
3	1	3	13	2	1	8	1	100
4	1	3	11	2	1	5	1	100

Test Large 1

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
2	1	1	7	1	8	2.5	0	0
3	1	1	5	1	3	7.1	0	0
4	1	1	4	1	9	4.5	0	0
5	1	1	75	1	3	1	0	0
6	1	1	6	1	1	6.6	0	0
7	1	1	20	1	2	7.5	0	0
8	1	1	34	1	7	5.6	0	0
9	2	1	32	1	4	6.9	0	0
10	2	1	51	1	6	3.5	0	0
10	2	1	61	1	3	4	0	0
11	2	1	57	1	6	6.8	0	0
12	2	1	62	1	8	5.3	0	0
13	2	1	70	1	5	3.6	0	0
14	2	1	90	1	7	3.4	0	0
15	2	1	98	1	2	5.7	0	0
1	1	1	9	2	4	7.2	0	0
2	1	1	1	2	2	4.3	0	0
2	1	1	83	2	8	1.2	0	0
4	1	1	38	2	9	2.8	0	0
8	1	1	89	2	9	1.5	0	0
13	2	1	82	2	5	3.7	0	0
1	1	2	40	2	2	1.3	0	0
2	1	2	14	2	1	1.9	0	0
4	1	2	13	2	7	7.2	0	0
4	1	2	73	2	3	1.5	0	0
5	1	2	17	2	8	6.5	0	0
5	1	2	88	2	9	1.7	0	0
6	1	2	18	2	10	5.6	0	0
7	1	2	28	2	10	7.9	0	0
8	1	2	37	2	2	5.1	0	0
8	1	2	44	2	8	2.9	0	0
9	2	2	46	2	2	3.8	0	0
9	2	2	47	2	2	4.3	0	0
10	2	2	66	2	10	7.5	0	0
11	2	2	67	2	2	6.8	0	0
12	2	2	68	2	6	7.9	0	0
13	2	2	76	2	5	7.3	0	0
14	2	2	94	2	7	5.9	0	0
15	2	2	96	2	2	7.4	0	0

2	1	2	2	3	5	4	0	0
3	1	2	11	3	9	6.4	0	0
3	1	2	81	3	6	1.2	0	0
6	1	2	42	3	9	3.6	0	0
1	1	3	15	3	3	3.8	0	0
2	1	3	21	3	8	4.7	0	0
4	1	3	24	3	8	6.4	0	0
4	1	3	45	3	1	2.6	0	0
5	1	3	31	3	3	7.7	0	0
6	1	3	49	3	2	3.2	0	0
6	1	3	50	3	9	5.9	0	0
7	1	3	52	3	5	5.1	0	0
7	1	3	53	3	2	3.8	0	0
8	1	3	58	3	5	2.7	0	0
8	1	3	95	3	5	2.2	0	0
9	2	3	56	3	6	4.9	0	0
9	2	3	78	3	5	3.7	0	0
11	2	3	84	3	4	7.4	0	0
12	2	3	86	3	9	7.4	0	0
13	2	3	93	3	2	5.4	0	0
13	2	3	99	3	9	2.8	0	0
5	1	1	8	4	7	6.5	0	0
1	1	2	16	4	9	5.6	0	0
1	1	2	27	4	6	1.2	0	0
2	1	2	23	4	7	3.3	0	0
3	1	2	64	4	4	1.3	0	0
1	1	3	3	4	9	2.9	0	0
1	1	3	19	4	4	1.2	0	0
2	1	3	10	4	5	5.1	0	0
3	1	3	26	4	6	7.2	0	0
3	1	3	72	4	8	1.5	0	0
8	1	3	77	4	10	3.7	0	0
10	2	3	80	4	5	7.8	0	0
1	1	4	25	4	10	3.8	0	0
2	1	4	22	4	5	6.6	0	0
2	1	4	29	4	4	4.1	0	0
3	1	4	30	4	1	3.4	0	0
3	1	4	33	4	5	3.4	0	0
3	1	4	65	4	10	2.5	0	0
4	1	4	36	4	5	7.1	0	0
5	1	4	43	4	5	3.3	0	0

5	1	4	60	4	2	4.9	0	0
6	1	4	63	4	5	4.8	0	0
6	1	4	69	4	9	2.6	0	0
6	1	4	92	4	9	2.4	0	0
7	1	4	74	4	9	7.6	0	0
8	1	4	39	4	4	5.2	0	0
9	2	4	79	4	8	6.6	0	0
10	2	4	87	4	10	4.9	0	0
11	2	4	91	4	4	6.8	0	0
1	1	4	12	5	7	4	0	0
8	1	4	41	5	5	4.6	0	0
1	1	5	54	5	1	7.5	0	0
2	1	5	85	5	5	7.9	0	0
3	1	5	59	5	5	4.7	0	0
3	1	5	71	5	8	4.4	0	0
4	1	5	97	5	4	5	0	0
4	1	5	100	5	9	4.3	0	0
15	2	5	35	5	8	2.8	0	0
15	2	5	48	5	9	4.4	0	0
15	2	5	55	5	8	2.8	0	0

Test Large 2

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
8	1	1	10	1	3	5.4	0	0
13	1	1	21	1	2	5.6	0	0
15	1	1	28	1	2	1.5	0	0
14	1	1	29	1	2	3.7	0	0
4	1	1	30	1	6	3.6	0	0
3	1	1	37	1	3	7.6	0	0
10	1	1	43	1	2	6	0	0
1	1	1	51	1	4	1.5	0	0
12	1	1	57	1	1	6.4	0	0
15	1	1	63	1	2	5.9	0	0
1	1	1	69	1	8	2.7	0	0
14	1	1	93	1	5	2.4	0	0
7	1	1	98	1	7	7.1	0	0
1	1	2	1	2	10	7.1	0	0
14	1	2	2	2	8	2.6	0	0
2	1	1	3	2	10	6.5	0	0
15	1	2	4	2	9	3.1	0	0
2	1	2	12	2	6	6.7	0	0
9	1	2	16	2	3	3.5	0	0
6	1	2	22	2	8	7.1	0	0
11	1	4	25	2	2	7.4	2	400
9	1	2	26	2	9	4.7	0	0
6	1	1	31	2	5	7.7	0	0
14	1	2	33	2	8	3.2	0	0
4	1	2	34	2	5	3.9	0	0
3	1	2	35	2	1	2.2	0	0
10	1	3	38	2	3	5.7	1	300
7	1	5	41	2	2	6.3	3	600
10	1	2	44	2	7	7.2	0	0
3	1	2	46	2	7	5.9	0	0
15	1	2	52	2	3	2.6	0	0
13	1	2	55	2	6	7.2	0	0
9	1	1	60	2	9	6.9	0	0
14	1	2	62	2	3	1.8	0	0
1	1	1	64	2	8	3.8	0	0
5	1	2	65	2	9	7.5	0	0
12	1	2	66	2	9	1.3	0	0
4	1	1	70	2	8	4.3	0	0
5	1	1	72	2	8	6.8	0	0

11	1	1	74	2	6	7.8	0	0
10	1	4	79	2	1	5	2	200
10	1	1	83	2	3	2	0	0
11	1	2	86	2	10	2.6	0	0
8	1	2	87	2	4	4.2	0	0
4	1	2	88	2	7	4.5	0	0
11	1	2	89	2	9	5.8	0	0
7	1	2	91	2	9	7.4	0	0
12	1	2	94	2	10	6.9	0	0
14	1	1	95	2	6	1.8	0	0
15	1	2	99	2	1	2.6	0	0
8	1	2	100	2	2	4.1	0	0
15	1	3	6	3	3	6	0	0
4	1	3	7	3	4	7.8	0	0
8	1	3	8	3	7	5	0	0
4	1	4	9	3	2	3	1	200
7	1	2	11	3	6	1	0	0
2	1	3	14	3	3	6.8	0	0
1	1	3	15	3	9	3.7	0	0
6	1	3	18	3	7	1.4	0	0
9	1	3	19	3	1	1.1	0	0
10	1	3	23	3	8	2.7	0	0
15	1	3	36	3	8	2.1	0	0
14	1	3	39	3	6	4.8	0	0
7	1	3	40	3	9	5.5	0	0
8	1	3	42	3	4	3.1	0	0
6	1	3	49	3	2	6.8	0	0
13	1	3	50	3	9	6.6	0	0
12	1	3	54	3	3	3.5	0	0
5	1	3	56	3	2	7.9	0	0
11	1	3	59	3	4	2.2	0	0
11	1	3	67	3	5	6	0	0
12	1	3	68	3	9	5	0	0
1	1	3	71	3	6	3.9	0	0
2	1	3	73	3	2	1.5	0	0
14	1	3	75	3	4	3.8	0	0
9	1	3	76	3	6	6.5	0	0
3	1	3	80	3	8	2.8	0	0
3	1	3	82	3	2	2.7	0	0
7	1	3	84	3	2	2.9	0	0
3	1	3	85	3	5	1.6	0	0

3	1	3	90	3	7	1.4	0	0
2	1	4	92	3	8	5.3	1	800
13	1	3	97	3	9	1.5	0	0
15	1	4	5	4	5	1.5	0	0
1	1	4	13	4	6	2.2	0	0
12	1	4	17	4	3	5.3	0	0
4	1	4	20	4	6	3.6	0	0
13	1	4	24	4	5	4.8	0	0
13	1	4	27	4	9	2.7	0	0
5	1	4	32	4	2	2.9	0	0
2	1	4	45	4	8	6.6	0	0
1	1	4	47	4	1	2	0	0
3	1	4	48	4	1	2	0	0
4	1	4	53	4	5	1.9	0	0
14	1	4	58	4	5	4.5	0	0
5	1	4	61	4	3	5	0	0
3	1	4	77	4	5	6.8	0	0
15	1	4	78	4	1	3.3	0	0
15	1	4	81	4	8	3.7	0	0
1	1	4	96	4	8	4.3	0	0

Test Large 3

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	1	4	1	4	3.8	0	0
2	1	1	25	1	7	6.6	0	0
3	1	1	8	1	5	7	0	0
4	1	1	21	1	4	6	0	0
4	1	1	68	1	8	1.5	0	0
5	1	1	42	1	4	3.5	0	0
6	1	1	89	1	2	4.7	0	0
7	1	1	27	1	7	5.4	0	0
7	1	1	36	1	4	2.5	0	0
9	1	1	60	1	8	7.1	0	0
10	1	1	56	1	1	2.4	0	0
13	1	1	22	1	9	7.4	0	0
14	1	1	72	1	1	1.4	0	0
15	1	1	84	1	4	6.4	0	0
15	1	1	85	1	7	1.2	0	0
12	1	4	65	1	1	6.8	3	300
5	1	1	10	2	10	3.4	0	0
6	1	1	45	2	5	3.1	0	0
8	1	1	50	2	3	7.5	0	0
10	1	1	41	2	7	2.3	0	0
10	1	1	58	2	7	2.8	0	0
11	1	1	33	2	10	1.5	0	0
12	1	1	9	2	5	3.1	0	0
14	1	1	67	2	3	5.9	0	0
1	1	2	26	2	3	1.5	0	0
1	1	2	29	2	9	6.7	0	0
2	1	2	15	2	2	7.3	0	0
3	1	2	1	2	9	2.1	0	0
4	1	2	30	2	3	5.9	0	0
5	1	2	47	2	6	6.4	0	0
5	1	2	51	2	3	2	0	0
6	1	2	28	2	7	7.2	0	0
7	1	2	94	2	10	3	0	0
7	1	2	100	2	8	5.4	0	0
8	1	2	78	2	8	4.1	0	0
8	1	2	92	2	5	2.9	0	0
9	1	2	57	2	9	3.5	0	0
9	1	2	91	2	8	4.5	0	0
10	1	2	38	2	7	3.2	0	0

10	1	2	99	2	9	5.2	0	0
11	1	2	75	2	5	2	0	0
12	1	2	63	2	8	2.4	0	0
13	1	2	82	2	4	6.7	0	0
14	1	2	13	2	7	1.5	0	0
14	1	2	83	2	6	3.8	0	0
1	1	4	44	2	1	6.9	2	200
6	1	4	69	2	1	5	2	200
8	1	4	93	2	1	6.1	2	200
1	1	1	40	3	8	3.8	0	0
2	1	1	31	3	5	1.1	0	0
11	1	1	79	3	6	5.8	0	0
12	1	1	97	3	6	3	0	0
3	1	2	90	3	7	6.2	0	0
11	1	2	87	3	4	6.2	0	0
12	1	2	7	3	5	5.9	0	0
14	1	2	81	3	7	3	0	0
15	1	2	64	3	10	5.8	0	0
1	1	3	23	3	3	7.2	0	0
1	1	3	53	3	5	1.4	0	0
2	1	3	62	3	7	1.6	0	0
2	1	3	73	3	5	7	0	0
3	1	3	5	3	6	3.6	0	0
3	1	3	37	3	6	4.4	0	0
4	1	3	70	3	4	7.2	0	0
5	1	3	49	3	2	2.9	0	0
5	1	3	88	3	4	5.4	0	0
6	1	3	19	3	9	1.8	0	0
6	1	3	54	3	9	6.7	0	0
7	1	3	52	3	5	6.3	0	0
8	1	3	18	3	9	3.4	0	0
8	1	3	32	3	5	5.2	0	0
9	1	3	3	3	5	1.8	0	0
9	1	3	24	3	6	6.5	0	0
10	1	3	11	3	7	3.5	0	0
10	1	3	39	3	5	4.3	0	0
11	1	3	16	3	9	7.6	0	0
11	1	3	59	3	5	1	0	0
12	1	3	77	3	5	6.3	0	0
13	1	3	46	3	3	3.5	0	0
13	1	3	61	3	6	3.4	0	0

14	1	3	86	3	1	1.4	0	0
14	1	3	96	3	6	7.2	0	0
15	1	3	20	3	6	6.8	0	0
15	1	3	55	3	6	1.5	0	0
5	1	4	34	3	2	6.3	1	200
9	1	4	14	3	1	4	1	100
14	1	4	98	3	1	4.4	1	100
2	1	4	12	4	4	4.3	0	0
2	1	4	43	4	9	3.6	0	0
4	1	4	48	4	4	5.6	0	0
6	1	4	66	4	8	3.3	0	0
7	1	4	35	4	4	5.6	0	0
10	1	4	95	4	3	6.3	0	0
11	1	4	71	4	9	6.1	0	0
13	1	4	80	4	4	7.4	0	0
14	1	4	2	4	4	3.3	0	0
15	1	4	6	4	5	1.1	0	0
15	1	4	17	4	6	2.7	0	0
15	1	4	74	4	8	3.1	0	0
15	1	4	76	4	4	1.8	0	0

Test Online 1

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	3	1	1	3	6.2	2	600
2	1	3	4	1	5	7.3	2	1000
4	1	3	5	1	3	6.3	2	600
1	1	2	6	1	7	7.5	1	700
3	1	3	8	1	3	6.1	2	600
2	1	2	9	2	4	3.3	0	0
3	1	2	11	2	5	1.5	0	0
3	1	2	14	2	6	3.7	0	0
4	1	2	15	2	5	5.3	0	0
2	1	2	16	1	1	5.1	1	100
4	1	2	19	2	2	1.5	0	0

Test Online 2

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
1	1	3	3	1	2	7	2	400
2	1	3	4	2	4	6.3	1	400
3	1	2	5	2	7	2.7	0	0
3	1	3	6	2	1	3.5	1	100
3	1	3	11	2	7	4.5	1	700
1	1	4	12	1	2	7.2	3	600
1	1	2	13	2	7	3.6	0	0
2	1	2	14	2	9	5.7	0	0
4	1	2	15	2	4	3.6	0	0
1	1	2	16	1	9	4.6	1	900
3	1	2	17	2	10	5.2	0	0
4	1	3	18	2	4	5.9	1	400
2	1	4	20	2	1	4.4	2	200

Test Online 3

Worker	Shift	Day	Job	Due	Priority	Processing Time	Sr	Penalty
3	1	2	1	2	9	6.3	0	0
1	1	2	2	2	9	1.4	0	0
4	1	2	3	2	3	1.5	0	0
3	1	2	5	2	10	1.4	0	0
1	1	3	6	2	1	4.2	1	100
2	1	3	8	1	4	5.9	2	800
3	1	3	10	2	2	7.5	1	200
1	1	3	12	2	3	4	1	300
4	1	2	13	2	8	2.2	0	0
2	1	2	14	1	6	7.2	1	600
1	1	2	17	2	5	5.1	0	0
1	1	2	19	2	9	1.9	0	0

VITA AUCTORIS

Patrick Rodd was born in 1982 in Windsor, Ontario. He graduated from the University of Windsor, Windsor, Ontario in 2005 and obtained a Bachelor of Engineering degree in Industrial and Manufacturing Systems Engineering. He then pursued his studies at the Masters level in Industrial and Manufacturing Systems Engineering at the University of Windsor, Windsor, Ontario, from September of 2005. He is currently a Master's degree candidate and intends to enter the workforce after his studies.