2000

Scheduling based on earliness and tardiness criteria in assembly job shops

Supachai Pathumnakul
Iowa State University

Scheduling based on earliness and tardiness criteria in assembly job shops

by

Supachai Pathumnakul

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

Major: Industrial Engineering
Major Professor: Pius J. Egbelu

Iowa State University
Ames, Iowa
2000

Graduate College
Iowa State University

This is to certify that the Doctoral dissertation of

Supachai Pathumnakul

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

Signature was redacted for privacy.

For the Graduate College

TABLE OF CONTENTS

LIST OF FIGURES

LISTS OF TABLES

ABSTRACT

In this research, the following four scheduling problems have been studied: (1) single machine problem with earliness cost minimization, (2) single machine problem with the sum of the weighted earliness and weighted tardiness cost minimization, (3) assembly job shop problem with earliness cost minimization, and (4) assembly job shop problem with the sum of weighted earliness and weighted tardiness cost minimization. Four mathematical models based on these four scheduling problems were developed in an effort to obtain optimal solutions. Six heuristic algorithms have been developed to solve the problems. The performances of the heuristic algorithms were demonstrated on some sample test problems. Quality of solutions and CPU time of solutions were the factors of interest. Several properties of optimal solutions for the single machine scheduling problem with the objective of minimizing the weighted earliness penalty have been identified in the research. Algorithms I, III, V, and VI were developed based on these identified properties while the algorithms II and IV were developed based on the tabu search concept.

Algorithms I and II were developed to solve the first case (1) problem. The results indicate that these two algorithms are able to produce solutions close to optimal in small size problems. The results also show that algorithm I is relatively better than algorithm II in large size problem.

Algorithms III and IV were developed to solve the second case (2) problem. Both algorithms obtained a small average deviation solutions (i.e., less than 2%) from optimal in small size test problems. For all problems tested, the algorithm IV is the best algorithm for solving the earliness/tardiness problems compared to algorithm III and the Ow & Morton algorithm.

Algorithm V was developed to solve the third case (3) problem. It obtained an average deviation solutions less than 1% from the optimal. Algorithm VI was developed to solve the fourth case (4) problem. Algorithm VI obtained an average deviation solutions of 2.53% from the optimal.

In testing all developed heuristics. the computational requirements for solving the problems are less than 2 second in all test problems.

CHAPTER 1

INTRODUCTION

In this research, the problem of scheduling jobs in an assembly job shop is addressed. An assembly job shop refers to a shop that involves both processing operations and assembly operations. It is assumed that each job has a product structure of components and subassemblies that assemble together to build up the end product. Each component or subassembly may need additional processing before mating with the other parts. The main objective of this research is to develop an algorithm to minimize the sum of weighted earliness and weighted tardiness for an assembly job shop problem. Additional to the assembly job shop problem, the following three other scheduling problems are also addressed in this research:

(1) single machine problem with earliness cost minimization,

(2) single machine problem with the sum of the weighted earliness and weighted tardiness cost minimization,

(3) assembly job shop problem with earliness cost minimization.

Six heuristic algorithms have been developed to solve the problems. Algorithms I and II were developed to solve the single machine problem with earliness cost minimization. Algorithms III and IV were developed to solve the single machine problem with the sum of the weighted earliness and weighted tardiness cost minimization. Algorithm V was developed to solve the assembly job shop problem with earliness cost minimization and Algorithm VI was developed to solve assembly job shop problem with the sum of weighted earliness and weighted tardiness cost minimization.

These heuristic algorithms were applied to solve sample problems and the solutions obtained from the heuristic algorithms are found to compare very favorably to optimal solutions obtained from mathematical models.

1.1 Problem statement

Given an assembly shop and a set of *N* final products (jobs) with due-dates, and each job requiring both assembly and processing operations, the objective is to schedule the jobs through the shop to minimize the sum of weighted earliness and tardiness penalties. Each product has a product structure consisting of components and subassemblies that require both machining and assembly operations. Each operation requires a specific machine from a set of *M* machines in the shop. There is an earliness cost for a product completed before its due-date. For assembly components, the earliness cost may be due to the time spent waiting for the corresponding mating component. Tardiness cost is incurred if a product is completed after its due date. The problem is to develop a schedule that minimizes the overall weighted cost due to both earliness and tardiness penalty.

1.2 Research motivation

Assembly shop is one of typical shops in industrial environment. As in the review of Nof *et al.* [30], it states that over 50% of total production time, and 20% of production cost in manufacturing are related to the assembly of manufactured products. One-third of a manufacturing company's labor is involved in assembly work, and 50% of direct labor costs in automotive industry are in assembly shops. These statistics indicate the importance of the assembly shop in real–world industry. They also imply that a good production plan in an assembly job shop may lead to major cost and production time reduction, which are required in today's highly competitive environment. Although the importance of assembly shop is significant, there have been very few reported research that focused on assembly shop scheduling. In 1994, Ramaudin and Marier [33] reported that no benchmark problem for the assembly job shop has been published in the scheduling literature. A primary reason is the complexity of the problem itself. The assembly job shop is more complicated than the traditional job shop, since it contains the staging delays. Staging delays are the process in which one part incurs some delays to wait for the arrival of its mating parts. This condition does not exist in traditional job shop scheduling problems.

With a few published papers on assembly job shop, most of them deal with such regular measures as mean flow time and completion time. To our knowledge, no published paper deals exactly with the minimization of earliness and tardiness costs (E/T) in an assembly job shop. The E/T criterion is a nonregular performance measure. It is an NP-complete problem even in the single machine case [6]. Even though it is not widely studied by researchers when compared to regular measures, it is, however, considered as one of the important measures in Just in time (JIT) production systems. In a JIT environment, jobs that finish early must be stored in the warehouse until their due-dates. This increases the inventory holding cost. On the other hand, jobs that complete after their due-dates incur tardiness penalties, loss of customer goodwill, or loss of future sales. It is clearly obvious that both early completion and tardy completion are undesirable.

Based on the above reasons, assembly job shop scheduling with earliness and tardiness cost minimization becomes an interesting problem since it matches the real problems encountered in industry.

1.3 Research objective

In this research, the problem of minimizing the sum of weighted earliness and weighted tardiness costs in an assembly job shop is studied. Because of the complexity of the problem as mentioned earlier, the optimal solution from exact algorithm is likely to require excessive computational time in large size problems. The development of an exact optimal scheduling algorithm is impractical. This means that the development of an efficient heuristic algorithm is more practical. In this research, we propose the use of heuristic technique to search for the best solution.

In our heuristics, the *m* machines job shop problem is decomposed into *m* separate single machine problems. We simply consider that the schedule of the assembly job shop is the union of sequences of jobs, with precedence constraints, on each single machine in the shop. Under the proposed solution approach, heuristic algorithms for E/T single machine problems are first developed. Thereafter, the algorithms are extended to solve the assembly job shop problem.

Two heuristic algorithms for minimizing weighted earliness cost subject to no tardy job, and two heuristic algorithms for minimizing the sum of weighted earliness and weighted tardiness costs on single machine scheduling are developed in this research. The first algorithm (minimizing weighted earliness cost) is based on the relationship between conflict jobs. Conflict jobs are the jobs whose processing times overlap each other on the same machine, if each job must be completed on its due-date. The second algorithm is also for minimizing weighted earliness cost. It is based on the tabu search heuristic. The third and fourth algorithms are for minimizing the sum of weighted earliness and weighted tardiness penalties. The third algorithm is the extension of the first heuristic and employs the pairwise interchange method. The fourth algorithm is a tabu search heuristic.

Two mathematical models for minimizing weighted earliness, and minimizing the sum of weighted earliness and weighted tardiness on a single machine problem are developed and used to obtain the optimal solutions for small size problems. The Lindo (Commercial Software) is applied to solve the mathematical models. Heuristic algorithms are latter developed to solve the more general forms of the problem. Sample problems are solved and the results of the solutions obtained by the heuristic and exact procedures are compared

The experience gained in solving the single machine problem was employed in developing the heuristic algorithms for the general problem of minimizing the weighted earliness penalty, and the sum of weighted earliness and weighted tardiness penalties in assembly shops involving multiple machines. The performances of the heuristic algorithms were compared with results obtained from exact procedure for some small size sample problems.

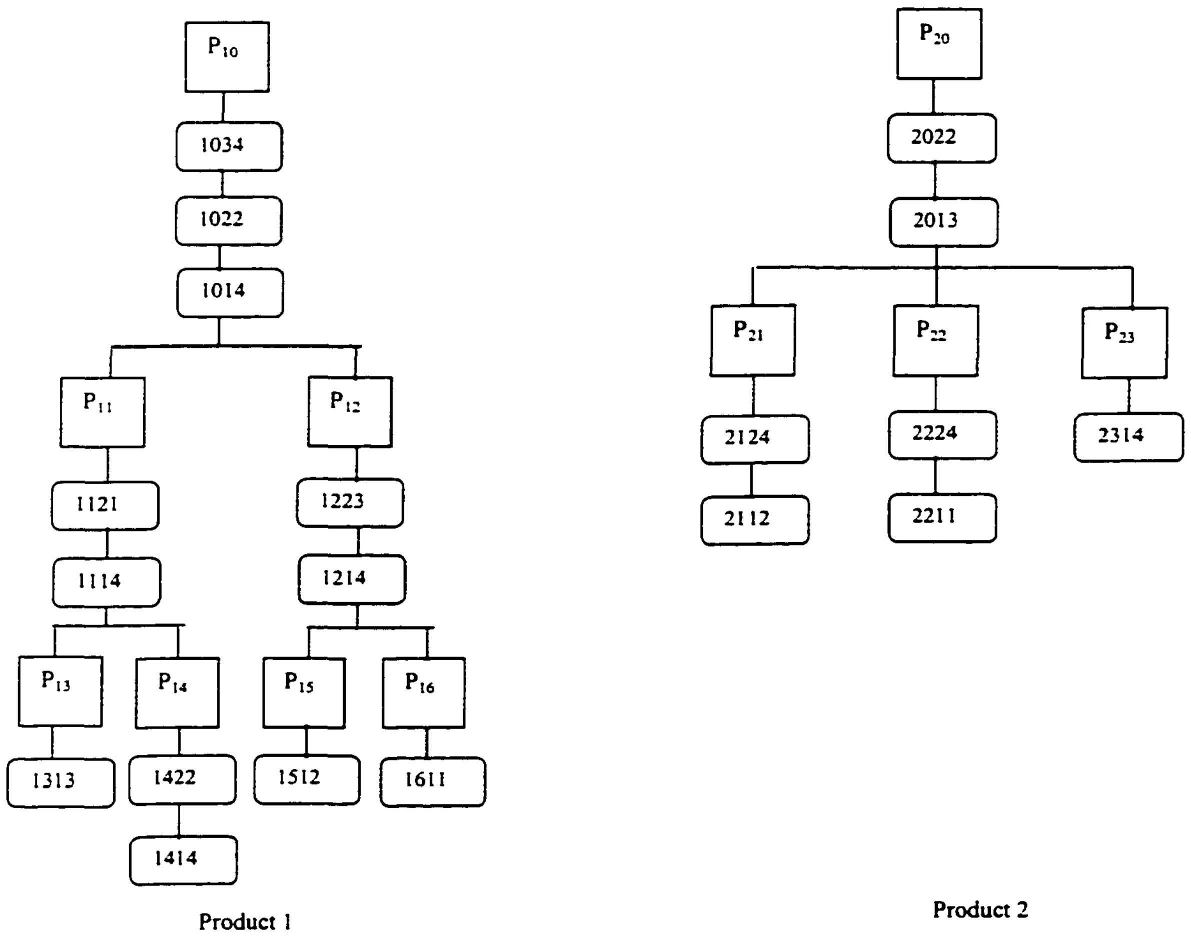
1.4 The assembly job shop environment

Each job in an assembly job shop consists of several components and subassemblies, with each component or subassembly requiring one or more operations. The operations can be both processing (machining) and assembly operations. The operations in an assembly job shop contain both serial and parallel operations (unlike the non-assembled, traditional, job shop where all operations are performed in series). For example, the operations of a component are performed serially, following the precedence relationships, while the operations of the another component belonging to the same assembly or subassembly are also performed serially, but in parallel with the former. As a result, scheduling in an assembly job shop requires the consideration of staging delays which are the delays of a component waiting for the other components to be mated together. The staging delay is different from the delays of components waiting for the availability of machines. The staging delay consideration underlines the complexity of an assembly job shop problem as compared to the non-assembled job shop.

In this research, a job consists of a set of component parts, and a set of sub-assemblies parts. The components that go into sub-assemblies and the sub-assemblies themselves may require some machining tasks. A task refers to an operation performed on a job, a component part or a subassembly part at a specific machine or workstation.

The product structure of a typical assembly product is as shown in Figure 1.1. The product structures shown are taken from Mckoy and Egbelu [28,29].

Each product is represented as an inverted tree with the root of the tree at the top. For example, item P_{i0} , for i = 1, 2 in Figure 1.1 is the root node of product i. There are two sets of nodes in the product structure. The first set of nodes labelled P_{ij} represent subassemblies and components. The label P_{ij} is also referred to as the *j-th* subjob (assembly or component) of job i. If j = 0, then P_{ij} represents the end product i, P_{ij}, for $j = 0$, also represents the final assembly. Thus $P_{i0} = P_i$ for all i
and represents the end product i. The final products are constructed by mating the components and subassemblies in some fashion. The P_{ij} nodes in a product tree



$\boxed{ijkl}$ Operation required by subjob P_{ij} , where k represents operation number and l is the required machine for k .

$\boxed{P_{ij}}$ Subjob j of final job i.

Figure 1.1 Product structure for assembly products

are arranged in levels. The levels are labelled sequentially from 0 and counting upward (1, 2, 3,....) down the tree. For example, in Figure 1.1, there are four levels, labelled 0, 1, 2 and 3 for product 1. Node P_{10} is in level 0, while nodes P_{13} and P_{14} are in the second level.

The second set of nodes are represented by four-tuples (*ijkm*). These nodes represent the additional processing operations required by the subjob or P_{ij} node before the subjob (i.e., P_{ij}) can be used as an assembly part of its parent node. For example, P_{11} may require drilling and milling operations before mating with P_{12} to form P_{10}. The four-tuple label (*ijkm*) in this set of nodes refer to the *k-th* operation of subjob *j* of the final product *i* where the *k-th* operation is performed on machine *m*.

1.5 Problem assumptions

Certain assumptions are made in developing the models in the research. These assumptions include the following:

a) all jobs are available at time zero,

b) no preemption is allowed,

c) machine breakdown is not considered,

d) jobs have due dates.

1.6 Contribution of the research

Scheduling is one of the most important issues in production system. A good schedule can result in significant savings both in cost and production time. Even though a high degree of effort is required in generating a good schedule, the scheduling problem remains a persistent problem in production system. Poor schedules are the causes of high work-in-process inventory, job tardiness, and low machine utilization. The computational time required for generating optimal schedules for practical size assembly scheduling problems is high. The scheduling problem of minimizing the total cost due to earliness and tardiness is NP-complete [6]. Therefore, the use of exact solution procedure in solving large size problems is impractical. The development of efficient solution methodologies that generate optimal or near-optimal solutions is attractive. Hence, the benefits derived from this proposed research include:

- The development of heuristic algorithms for minimizing the weighted earliness penalty in a single machine problem.
- *The development of algorithms for minimizing the sum of weighted* earliness and weighted tardiness penalties in a single machine problem.
- The development of a solution methodology for minimizing the weighted earliness penalty for assembly job shop scheduling problem.
- The development of a general-purpose algorithm for minimizing the sum of weighted earliness and weighted tardiness penalties in both the assembly shop and job shop.

1.7 Organization of the dissertation

The remainder of this dissertation is organized into five chapters. In Chapter 2, the literature related to this research are reviewed. Chapter 3 describes the model development in single machine problem. It includes the mathematical models for the problem of minimizing the weighted earliness penalty in single machine scheduling and the problem of minimizing the sum of weighted earliness and weighted tardiness penalties in single machine scheduling. Heuristic algorithms for minimizing weighted earliness penalty, and sum of weighted earliness and weighted tardiness penalties in single machine scheduling problems are presented. Chapter 4 describes the model developments in assembly job shop problem. It includes the mathematical models for the problem of minimizing the weighted earliness penalty in assembly job shop scheduling and the problem of minimizing the sum of weighted earliness and weighted tardiness penalties in assembly job shop scheduling. Heuristic algorithms for minimizing weighted earliness penalty, and sum of weighted earliness and weighted tardiness penalties in assembly job shop problems are presented in this chapter.

Comparisons between solutions obtained from mathematical models and the heuristics in both single machine and assembly job shop problems are given in chapter 5. The conclusion on this research is drawn in the Chapter 6.

The proof of the propositions, the algorithm illustrations, and sensitivity study are also presented in the Appendices of this dissertation.

CHAPTER 2
LITERATURE REVIEW

This chapter reviews previous research work on production scheduling in the single machine problem, general job shop problem, and the assembly job shop problem. It focuses on the scheduling methods developed for minimizing earliness and tardiness costs.

2.1 Single machine problem

The majority of published papers on the scheduling models with earliness and tardiness (E/T) penalties deals with the single machine model. These papers can be classified into three groups. The first group deals only with the earliness cost. The second group focuses on the tardiness cost, and the last group considers both earliness and tardiness costs at the same time.

2.1.1 Earliness cost model

The allowance for no tardy jobs is the basic assumption in the papers that deal only with earliness cost. It is assumed that every job must be completed before or on the due-date, and leave the shop on the due-date. If a job is completed before its due date, then inventory holding cost is incurred. Chand and Schneeberger [7] assume that all jobs are available at time zero, and that the due-date for each job is known. They studied two types of problems which they referred to as the *Weighted Earliness Problem* (WE-Problem), and the *Constrainted Weighted Earliness Problem* (CWE-Problem). In WE-Problem, machine idle time can be inserted, but it is not permitted in CWE-Problem. The authors show that both the CWE-Problem and WE-Problem are NP-hard, and optimal solutions can be obtained by a polynomial time algorithm in some specified situations. Two heuristics which are modified from Smith-heuristic [34] are given to solve the CWE-Problem and WE-Problem, respectively. A dynamic programming procedure was also developed for solving the

WE-Problem, but this method was not recommended for solving the problems that involve larger than 15 jobs. Chhajed [12] introduced a problem where N jobs are to be scheduled on a single machine and assigned to one of two due-dates which are given at equal interval. The due-date cost is considered additional to the earliness cost. The author assumes that customers prefer their jobs to be shipped as early as possible and thus there is a penalty for assigning jobs a late due-date. The problem is shown to be NP-hard. A method to obtain lower and upper bounds of the problem is also provided in the paper. In Asano and Ohta [3], not only is the due-time of each job known, but also the ready time is prespecified. That is, every job is available for processing on its ready time and cannot be processed earlier than this time. The authors propose an optimization algorithm using dominance relation for scheduling problem. The algorithm is based on the branch and bound approach. This method is applied to determine the optimal solution so as to minimize the total earliness cost with respect to the due date of each job. Their dominance relation is determined on the basis of earliest and latest completion times for unassigned jobs to specify the antecedent relation for jobs. Using this dominance relation in the branching operation, the problem size at each node can be reduced as small as possible. The strong lower bound of the algorithm is achieved by determining the minimum overlapping time for all unassigned jobs.

Qi and Tu [32] have studied the scheduling of a single machine to minimize the earliness penalty subject to no tardy jobs. The due-date in their problem is determined by the equal slack (SLK) method. Two cases of the problem are studied in the paper. The first case is the problem with equally weighted monotonous penalty objective function. The second one is the problem with weighted linear penalty objective function. For these two cases, two algorithms are respectively proposed by the authors, and optimal solutions can be obtained within polynomial time.

2.1.2 Tardiness cost model

This type of problem is the opposite of the problem of minimizing earliness cost. Only the tardiness cost is considered. A set of jobs with given due-dates is scheduled on a single machine. If a job is completed after its due-date, tardiness cost is incurred. Lawler [25] and Lenstra *et al.* [26] have shown that the weighted tardiness problem is strongly NP-hard. A pseudopolynomial time algorithm was developed by Lawler [25] for weighted tardiness problems with agreeable weights. For example, the weight for *i* is less than the weight for *j* when processing time of *i* (P_i) is greater than the processing time of *j* (P_j).

Numerous algorithms involving both exact methods and heuristics have been developed to solve the weighted tardiness problem. A set of heuristics has been developed by Fisher [16]. Dynamic programming ([5], [23]) and Branch and bound algorithm [16] have also been proposed.

Ow and Morton [31] presented a dispatch heuristic for scheduling weighted tardiness problem. In their dispatch method, whenever there are jobs waiting to be processed, and the machine is available, a waiting job is selected by using a priority function. The job that has the highest priority is scheduled next. Szwarc and Liu [39] have studied the weighted tardiness single machine scheduling problem with proportional weights. The tardiness penalties are proportional to the processing times. They present a two-stage decomposition method which is applied to solve the problem completely or reducing it to a much smaller problem, and then applying Arkin and Roundy's algorithm [2] to solve the small problem.

2.1.3 Earliness and Tardiness costs (E/T) model

In the review of Baker and Scudder [6], there are various types of assumptions made for the E/T problem. In some E/T problems, due-date is given, while in others the due-date is also a decision variable. In cases where due-date is a decision variable, both the due-date and job sequence are optimized simultaneously. Some have the same due-date for all jobs, which they call, common due-date, while others allow distinct due-dates. Some models contain common penalties, while others permit differences between the earliness and tardiness penalties or unequal penalties among the jobs. With so many variations of the E/T model, there have been many papers published in each category. Some of the interesting papers are reviewed in this research.

In Kanet's paper [22], the problem of minimizing the total unweighted earliness and tardiness around a single common due-date is studied. The major assumption made in his work is that the due-date is not early enough to act as a constraint on the scheduling decision. This assumption is called "unrestrictively late". Under this assumption, a polynomial time algorithm for finding an optimal solution is developed and presented by the author. The Kanet's problem is generalized by Sundaraghavan and Ahmed [36] to scheduling with single machine and several identical parallel machines. Bagchi, Sullivan and Chang [4] also studied another generalization of Kanet's problem. They assume that all jobs have equal earliness and tardiness penalties. The objective is to minimize the mean absolute deviation of job completion times about a common due date. Two more problems which are the generalization of Kanet's problem are considered by Hall and Posner [20], and Hall, *et al.* [21]. In [20], each job j has a weight or value w_j, which is symmetric, but weight may not be equal between jobs. All jobs have the same due-date which is unrestrictively late (the due-date is not early enough to constraint the scheduling decision). The authors have proved that this problem is NP-complete. They described optimality conditions, and presented a dynamic programming algorithm, which is a pseudopolynomial time algorithm. This algorithm can solve the generated scheduling problem with 2000 jobs within two minutes of CPU time on a minicomputer. The Hall, *et al.* [21] paper is the companion paper of [20] but with some differences in assumptions. In [21], the authors consider the problem of minimizing the unweighted earliness and tardiness of jobs with a common due-date that is early enough to constraint the scheduling decision. Similar to [20], the paper contains a description of several optimality conditions of the problem. The problem has been proved to be NP-complete in the ordinary sense. Finally, a pseudopolynomial algorithm based on dynamic programming is proposed. The algorithm can solve problem of up to 1000 jobs in less than three minutes of CPU time on a minicomputer.

When the E/T model has jobs with different due-dates, it makes the problem more difficult to determine a minimum cost schedule than the model with common job due-dates. However, the problem turns out to be simpler, if the due-dates are also treated as decision variables [6]. The E/T problem on a single machine with distinct due-dates has been proved to be NP-complete by Garey *et al.* [19]. Recent papers dealing with this model (E/T with jobs with distinct due-dates) are classified into two different categories. In one category, inserted machine idle time is allowed, while in the other, it is not allowed.

Abul-Razaq and Potts [1] and Ow and Morton [31] in their papers addressed problems that do not allow insert idle time into schedule. In [1], the paper first presented a dynamic programming formulation for this problem, but the number of states required in this formulation is prohibitively large. Branch and bound is then explored for solving the problem instead. However, the dynamic programming state-space relaxation technique which is developed by Christofides *et al.* [13] is used to obtain a lower bound for the branch and bound approach. In this method, a relaxed problem is obtained from a dynamic programming formulation by mapping the original state-space onto a smaller state-space and performing the recursion on this smaller state-space. Their computational results suggest that the solution time is large when dealing with problems containing more than 20 jobs. Discussions for improving the lower bound through the use of penalties and the use of state-space modifiers are also presented. Ow and Morton [31] propose a dispatch algorithm for the E/T problem with distinct due-dates, and no inserted machine idle times. In their dispatch algorithm, the priorities of unscheduled jobs are determined when the machine becomes available. The highest priority job is selected and schedule next. Their priority rule is based on the slack time of unscheduled jobs at the moment that the machine is available, and the value of *k*, where *k* is assigned by the scheduler. Large value of *k* is recommended, when job due-dates are close together and the processing time is not very long. On the other hand, when the due-dates are evenly distributed, *k* should be small. The authors also presented a new search method, Filtered Beam Search, which is modified from the Beam Search method used in artificial intelligence. This search method produced very good solutions compared to their proposed dispatch algorithm.

For problems that allow inserted machine idle time, there have been various proposed algorithms reported in the literature. Fry *et al.* [17] present heuristic for the E/T model with inserted idle time. Their heuristic has two aspects, the sequencing of jobs and the insertion of idle time between jobs. The paper shows that the optimal idle time insertion between jobs can be achieved by solving a linear programming problem when a sequence is found. For locating the best sequence, a solution procedure based on the adjacent pairwise interchange (API) method is applied. This heuristic procedure is used to obtain a local optimal solution. Their computational results suggest that the solutions from this heuristic are within 2% of the optimal solutions. A heuristic based on adjacent pairwise interchange is also presented in Fry *et al.* [18], but it is used to minimize mean absolute lateness when job due-dates are not common and all jobs are unweighted. This heuristic is different from the heuristic of Fry *et al.* [17], One aspect is that the heuristic procedure of Fry *et al.* [17] evaluates only one local optimum as its solution, while the heuristic in Fry *et al.* [18] evaluates nine different local optima and selects the best as its solution. The results from their test problems show that this heuristic performs well, since the solutions from the heuristic are worse than the optimal solutions by 2.49% on the average. Another procedure that first locates an initial sequence, and then inserts idle time optimally into the schedule is presented by Chang and Lee [8]. They present a greedy heuristic for solving a bicriterion single machine scheduling problem. The bicriterion includes the minimization of the makespan and the total absolute deviation. The objective function is the linear combination of these two objectives.

An optimal algorithm and a family of heuristic procedures are developed by Yano and Kim [41] for minimizing the sum of weighted earliness and weighted tardiness on a single machine when the weights are proportional to the processing time of jobs. Branch and bound procedure is applied to find the optimal solution of the problem. The authors derive some dominance criteria for eliminating some possible sequences that must not be considered in branch and bound search.

These dominance properties are also used as a basis for constructing good initial solution for some of their heuristics. The heuristic solutions are improved by using the pairwise interchange method. The results of the tested problems indicate that the composite heuristic which combines several sorting procedures and a simple pairwise interchange method is a very good method compared to the solutions from branch and bound.

Davis and Kanet [14] propose an algorithm called TIMETABLER procedure for optimizing the timing when a sequence is given. The starting times of jobs in a given sequence are shifted to minimize sum of earliness and tardiness costs in the TIMETABLER procedure. Szwarc [38] stated that the arrangement of adjacent jobs in an optimal schedule depends on a critical value of the start time, if we deal with E/T model on single machine with job independent penalties. For example, there exists a single critical value t_{ij} such that i precedes j ($i \rightarrow j$) if processing for this pair starts not earlier than t_{ij} and $j \rightarrow i$ if processing begins before t_{ij}. Due to this property, the branching scheme for the branch and bound procedure is developed. This scheme significantly reduces the search on 70 test problems containing 10 jobs each. To handle much larger problems, the authors suggest that this scheme should be used in any branch and bound procedure along with a good lower bound. Kim and Yano [24] consider a single machine scheduling problem with the objective of minimizing the mean tardiness and earliness when the due-dates are distinct. They investigate some properties of optimal solutions. Based on these properties, a branch and bound algorithm and a heuristic are developed for solving the problem. Problems with 20 jobs can be optimally solved within a modest amount of CPU time by applying the properties. They observed that less widely dispersed due-dates leads to excessive computational time in branch and bound algorithm. Thus, the heuristic should be applied in that situation. Their results also show that some simple sorting heuristics can produce solutions within 30% of optimal solutions, and this gap can be reduced by using some simple improvement method such as the pairwise interchange.

A dynamic single machine scheduling problem where the objective is to minimize the sum of weighted tardiness and weighted earliness costs is considered by Sridharan and Zhou [35]. They develop a single-pass heuristic based on decision theory for generating a schedule with embedded idle times, instead of inserting idle time after constructing a sequence. Their heuristic works as a dispatch procedure. At each decision point, waiting jobs in the queue plus the jobs that will arrive soon are considered to determine the next job to be processed and at what time. Chang [9] proposed a branch and bound approach for single machine scheduling with unweighted earliness and unweighted tardiness problem. In this approach, Chang first constructs an ideal JIT solution in which each job is completed at its due-date. If overlapping in processing time (i.e. processing conflict) among the jobs does not exist, the solution is optimal, but it rarely happens in practice. To deal with the overlapping problem, Chang provides some properties and theorems for eliminating processing time overlap among the jobs. Based on these properties and theorems, a bounding scheme for calculating different lower bounds for branch and bound procedure is presented.

2.2 Job shop with assembly considerations

As stated earlier, research in the deterministic production assembly job shop has been less extensive than that of typical job shops. One of the major reasons is that the assembly job shop contains both precedence relations between jobs as well as operations. It also deals with the staging delay that is the time that a part waits for the other parts to be mated together.

According to the complexity of the problem, the generation of optimal schedule requires excessive computational time. It is impractical to solve reasonably large size problems by using pure optimal scheduling methods. For this reason, most of researchers dealing with assembly job shop problems focus their efforts on developing efficient solution algorithms that generate near-optimal solutions with measurable performances [27].

Chen and Wilhelm [10] present a heuristic for kitting problem in multi-echelon assembly system. The objective is to allocate on-hand stock and anticipate future deliveries to kits in order to minimize total cost which consists of job earliness, job tardiness, and in-process holding costs. They describe the kitting problem and compare the performance of their heuristic with two heuristics that are commonly used in industry. A decision variable in this problem is the day on which each kit is to be composed. Their heuristic consists of two stages. The first stage is to determine the sequence position in the final schedule of each job (single end product). This procedure is based on the slack time of each job and the earliest dispatching rule. The second stage is to determine the kitting days of subassemblies of each job based on the sequence position of jobs from the first stage. The computational results indicate that this heuristic outperforms the other two heuristics which are commonly used in industry.

A heuristic algorithm for maximizing the machine utilization in an assembly job shop subject to satisfying job due-dates is developed by Doctor *et al.* [15]. Their heuristic is based on the construction of nondelay schedule. Idle time is allowed only when inserting idle time improves the performance measure. The heuristic first selects the operation which creates minimum slack time. The selected operation is the next to be scheduled on its required machine. However, some idle period may occur after scheduling the selected operation. Then the heuristic seeks one or more other operations to fill the imbedded idle time. The results indicate that this heuristic produces good solutions for the assembly jobs with no more than three levels of assemblies.

Chen and Wilhelm [11] developed an approach for minimizing the total cost of allocating resources in multi-echelon assembly system. This kitting problem is the same problem presented in Chen and Wilhelm [10]. Their optimizing method bases on the Lagrangian relaxation. It decomposes the problem into subproblems. Several preprocessing steps are included in this optimizing method. This procedure also consists of dominance properties to enhance the efficiency of a specialized branching rule and a dynamic programming algorithm to solve the subproblems. The results show that this approach outperforms OSL, a standard mathematical programming package.

McKoy and Egbelu [28] propose a heuristic algorithm for minimizing the production flow time (makespan) in a job shop scheduling problem. They also constructed a mathematical model for the problem and solved some problem instances to obtain optimal solution. Heuristic is developed and tested against an exact solution procedure on some test problems.

McKoy and Egbelu [29] consider the problem of scheduling jobs with both processing and assembly requirements in a job shop (assembly job shop). The objective is to minimize the production completion time. Both exact and heuristic algorithms are developed in this paper, Two production strategies are considered in the problem. The first strategy is that batches of identical parts generated from the bill of materials (BOMs) of the various end products are integrated together to form a supper batch. This strategy benefits from the minimization of the number of machine setups and machine setup times. In the second strategy, each batch is treated as individual job, no integration of batches of identical parts was considered. The performance of the heuristics based on these two strategies are above the optimal solution by 1.20% on the average. The paper indicates that the first strategy outperforms the second strategy when the setup times are low. On the other hand, there is no clear indication that one product strategy is better than the other when the setup times are moderate to large.

CHAPTER 3

MODEL DEVELOPMENT IN SINGLE MACHINE PROBLEM

In this chapter, the solution methodologies for solving the problem of minimizing weighted earliness penalty, and the sum of weighted earliness and weighted tardiness penalties in a single machine problem along with their mathematical models are presented.

3.1 Single machine problem with earliness cost minimization

The basic assumption for this problem is that job tardiness is not allowed. Only the earliness cost is considered. The problem is to schedule a set of jobs for minimizing the total weighted earliness cost. Jobs may have different processing times, distinct due-dates, and unequal weighted earliness cost. This problem is proved to be NP-hard [7]. The problem was first modeled mathematically. The mathematical formulation is as given in section 3.1.1.

3.1.1 Problem formulation

The mathematical model of the single machine problem with the objective of minimizing the weighted earliness penalty is as given below.

Objective function

$$Min \sum_{i=1}^{N} (E_i * (D_i - C_i)) \qquad (1)$$

Subject to

$C_i - S_i = t_i$ *for* $\forall i$ *(Processing time constraints)* *(2)*

$C_i \leq D_i$ *for* $\forall i$ *(No tardy job constraints)* *(3)*

$C_j - C_i + \alpha(1 - X_{ij}) \geq t_j$ *for* $\forall i, j$ *(Disjunctive constraints)* *(4)*

$C_i - C_j + \alpha X_{ij} \geq t_i$ *for* $\forall i, j$ *(Disjunctive constraints)* *(5)*

$S_i \geq 0$ *for* $\forall i$ *(6)*

$X_{ij} \in \{0, 1\}$, *integer,* *for* $\forall i$ *(7)*

where N = number of jobs,

S_i = starting time of job i,

C_i = completion time of job i,

t_i = processing time of job i,

D_i = due-date of job i,

α = large positive number,

E_i = earliness cost of job i (penalty per unit time of earliness for job i),

$$X_{ij} = \begin{cases} 1, & \text{if job } i \text{ precedes job } j, \\ 0, & \text{otherwise.} \end{cases}$$

Constraints *(2)* express that processing time of a job is equal to the difference between its starting time and its completion time. Constraints *(3)* guarantee that there is no tardy job in the system. Constraints *(4)* and *(5)* ensure that no two jobs can be processed simultaneously. Constraints *(6)* express that starting time of job *i* must be positive. Integrality requirement on X_{ij} is described in constraint *(7)*.

The presented mathematical model is impractical for solving reasonable size problem. Therefore, heuristic algorithms are developed for solving problems in this research. The first heuristic, algorithm I, is developed based on the local optimality condition (section 3.1.2). The second heuristic, algorithm II, is based on tabu search.

3.1.2. Local optimality condition

In this section, the local optimality conditions between two jobs are described. Under these conditions, the optimal ordering of any two jobs is derived. A heuristic for minimizing the weighted earliness is developed in the next section based on the derived local optimality conditions.

Let *i, j* denote jobs to be sequenced with processing time t_i, t_j, due-dates D_i, D_j, and earliness cost per unit time, E_i, E_j , respectively. If $D_j \geq D_i$ and $t_j > D_j - D_i$, then there exists an overlap in processing period between jobs *i* and *j*, if both jobs must be completed on their due-dates (see Fig. 3.1).

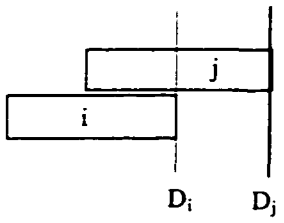


Figure 3.1 Jobs *i* and *j* overlap each other, if each completes on its due date

Proposition 1. For the case where jobs *i* and *j* are not possible to complete exactly on their due-dates due to conflict, if $\frac{E_i}{t_i} \leq \frac{E_j}{t_j}$ and $D_i < D_j$, then the optimal non-conflict ordering between jobs *i* and *j* is that job *i* precedes job *j* ($i \to j$) as shown in Fig. 3.2.

Proof. (see Appendix A).

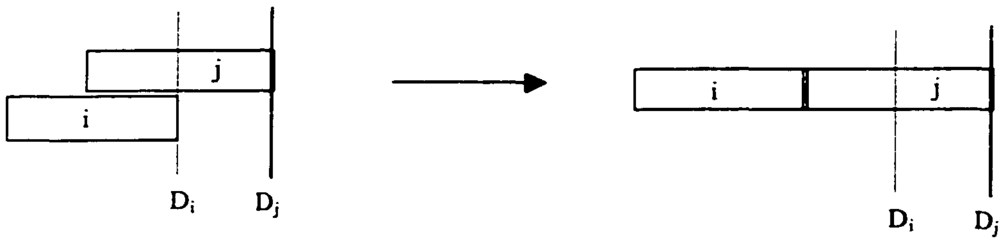


Figure 3.2 Illustration of proposition 1

Proposition 2. For the case where $\frac{E_i}{t_i} > \frac{E_j}{t_j}$, $D_i < D_j$, and both jobs *i* and *j* are not possible to complete exactly on their due-dates due to conflict, if $(D_i - D_j) \le \frac{t_i E_j - t_j E_i}{(E_i + E_j)}$, then the optimal non-conflict ordering between jobs *i* and *j* is that job *i* precedes job *j* $(i \rightarrow j)$, otherwise $j \rightarrow i$ (see Fig 3.3).

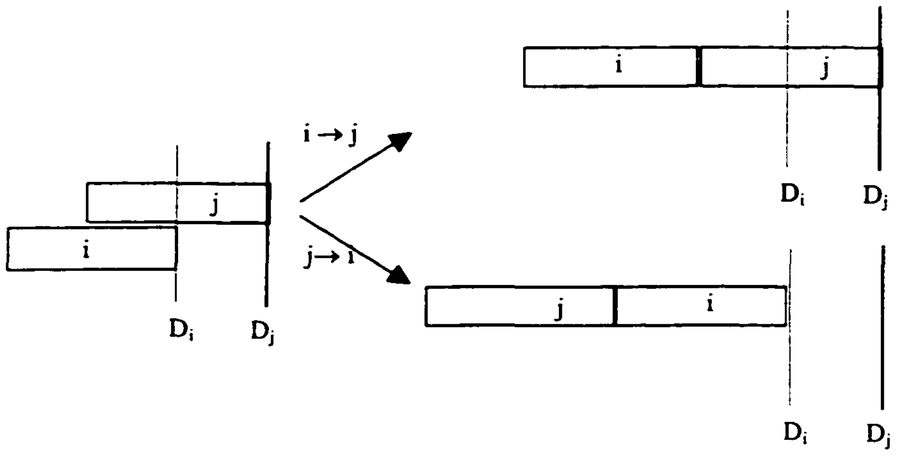


Figure 3.3 Illustration of proposition 2

Proof. (see Appendix A).

Proposition 3. For the case where jobs *i* and *j* are not possible to complete exactly on their due-dates due to conflict between jobs *i* , *j* and *k*, where *k* is already scheduled, If $\frac{E_i}{t_i} \le \frac{E_j}{t_j}$, then the optimal non-conflict ordering between jobs *i* and *j* is that job *i* precedes job *j* .

For example, in Fig. 3.4, suppose another job (i.e. job *k*) is scheduled for processing between the time period from *B* to *B** and the due-dates of jobs *i* and *j* are in this time period (i.e. $B \le D_i, D_j \le B^*$). Then jobs *i* and *j* can not be processed

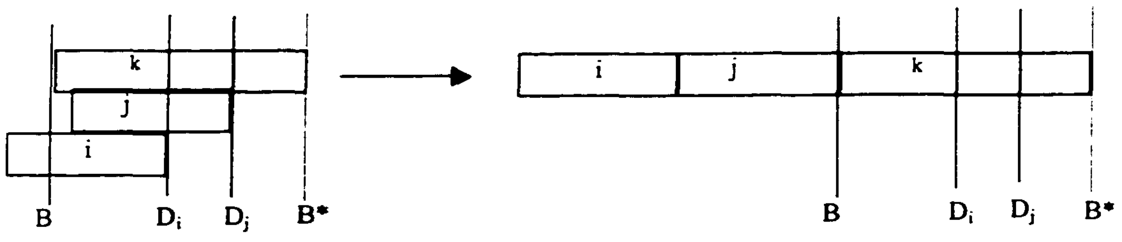


Figure 3.4 Illustration of proposition 3

between B to B^*. If $\frac{E_i}{t_i} \leq \frac{E_j}{t_j}$, then the optimal non-conflict ordering is that $i \rightarrow j$ and the completion time of job j is at time B.

Proof. (see Appendix A).

Proposition 4. Consider the case where job k is already scheduled. Job j is not possible to complete exactly on its due-date due to conflict between jobs j and i, and jobs j and k. Job i is not possible to complete exactly on its due-date due to conflict between jobs i and j (see Fig. 3.5(a)). If $\frac{E_i}{t_i} \leq \frac{E_j}{t_j}$, then the optimal non-conflict ordering between jobs i and j is that job i precedes job j $(i \rightarrow j)$.

For example, in Fig 3.5, suppose another job (i.e. job k) is scheduled for processing between the time period from B to B^* and the due-date of job j is in this time period (i.e. $B \leq D_j \leq B^*$). Thus job j can not be processed from time B to D_j. On the other hand, job i has conflict with job j, but not with job k. If $\frac{E_i}{t_i} \leq \frac{E_j}{t_j}$, then the optimal non-conflict ordering is that $i \rightarrow j$ and the completion time of job j is at time B.

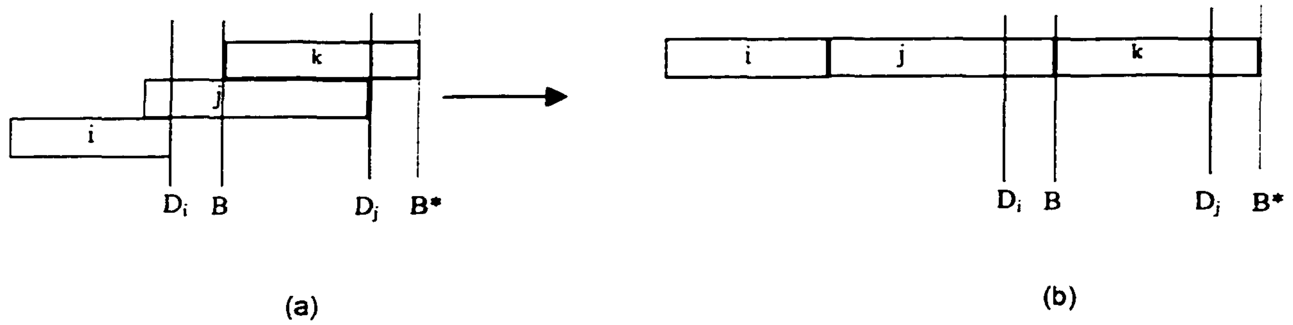


Figure 3.5 Illustration of proposition 4

Proof (see Appendix A).

Proposition 5. Consider the case where job *k* is already scheduled. Job *j* is not possible to complete exactly on its due-date due to conflict between jobs *j* and *i*, and jobs *j* and *k*. Job *i* is not possible to complete exactly on its due-date due to conflict between jobs *i* and *j* (see Fig. 3.6(a)). If $\frac{E_i}{t_i} > \frac{E_j}{t_j}$ and $(D_i - D_j + T) \le \frac{t_iE_j - t_jE_i}{(E_i + E_j)}$ where *T* is the lenght of time that job *j* can not be processed until its due-date due to the conflict between jobs *j* and *k*, then the optimal non-conflict ordering between jobs *i* and *j* is that job *i* precedes job *j* ($i \rightarrow j$). On the other hand, if $\frac{E_i}{t_i} > \frac{E_j}{t_j}$ and $(D_i - D_j + T) > \frac{t_iE_j - t_jE_i}{(E_i + E_j)}$, then the optimal non-conflict ordering between jobs *i* and *j* is that job *j* precedes job *i* ($j \rightarrow i$). This proposition can be shown as in Fig 3.6.

In Figure 3.6, suppose that job *k* is already scheduled for processing between the time period from *B* to *B** and the due-date of job *j* is in this time period (i.e. $B \le D_j \le B^*$). Therefore job *j* can not be processed from *B* to D_j . On the other hand, job *i* has conflict with job *j*, but not with job *k*. In this case, $T = D_j - B$. If $\frac{E_i}{t_i} > \frac{E_j}{t_j}$ and $(D_i - D_j + T) \le \frac{t_i E_j - t_j E_i}{(E_i + E_j)}$, then $i \rightarrow j$ as in Fig. 3.6(b). If $\frac{E_i}{t_i} > \frac{E_j}{t_j}$ and $(D_i - D_j + T) > \frac{t_i E_j - t_j E_i}{(E_i + E_j)}$, then $j \rightarrow i$ as in Fig. 3.6(c).

Proof. (see Appendix A)

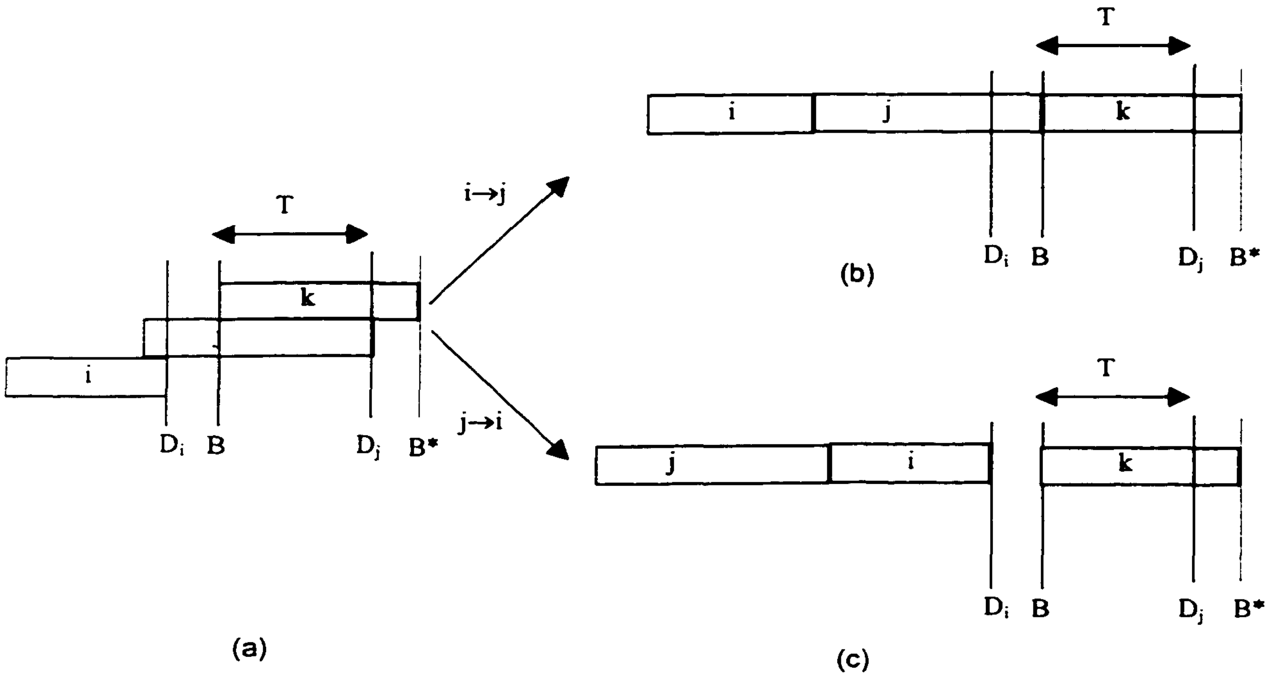


Figure 3.6 Illustration of proposition 5

3.1.3 Heuristics for the early problem

As stated earlier, single machine problem with earliness cost minimization is NP-hard [7]. The computation time required for generating optimal schedules from mathematical programming procedure for practical size problems (i.e. larger than 15 jobs) is high. Thus the mathematical programming procedure is not a practical approach for solving the problems. The development of efficient heuristics that generate optimal or near-optimal solutions is attractive.

In this section, two heuristic algorithms are developed. The first one is based on the local optimality conditions derived in the previous section. The second one is a tabu search heuristic. In the first algorithm, each job is first scheduled with its completion time corresponding to its due-date, called ideal solution. If there is no conflict between jobs, the solution is optimal, otherwise it is an infeasible solution. Any conflict between a pair of jobs can be eliminated by applying the local optimality conditions presented in section 3.1.2. The conflict elimination algorithm starts from the latest conflict where the latest conflict is the conflict of jobs whose due dates are latter than the other conflict jobs in the system. After the latest conflict is eliminated, the algorithm moves backward to eliminate the next latest conflict which now becomes the new latest conflict in the system. The algorithm moves backward until all conflicts are eliminated. After eliminating all conflicts from the ideal solution, some imbedded idle periods of machine may occur. The algorithm will search for better solution by filling any job into the imbedded idle period. The due-date constraint of the job selected to fill in the machine idle period can not be violated. After placing a selected job into an imbedded idle period, jobs that are previously scheduled before the imbedded idle period may no longer be in the best sequence. For example, suppose the obtained sequence after eliminating all conflicts is $1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 6$, and there is an imbedded idle time between jobs *2* and *5*. Job *4* can be partially placed in the imbedded period without violating its due-date and the new sequence (i.e.$1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$) now gives a lower cost than the sequence $1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 6$. Now the best sequence is $1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$. The sequence of jobs after job *4* (i.e. $5 \rightarrow 6$) is still the best sequence based on the propositions, since the order is unchanged. However, the sequence of jobs before job *4* (i.e.*1→3→2*) may not be the best sequence based on the propositions, since the sequence is changed. From this point of view, the schedule may be improved by searching for the best sequence for jobs *1, 2* and *3* based on the propositions. This can be done by pegging the sequence of job *4* and the jobs after job 4 (i.e. *4→5→6),* and setting the jobs before job *4* (i.e. jobs *1, 2, 3*) in an ideal schedule (set their completion times at their due-dates). Then, any conflict between jobs 1, 2, 3 occurring on the schedule is eliminated by conflict elimination procedure again.

In the second algorithm, the initial solution is simply constructed by scheduling jobs based on their due-dates. The earliest due-date job is first scheduled as early as possible, then the second earliest due-date job is scheduled as soon as the earliest due-date job is completed, and so on. After all jobs are scheduled, each job is right shifted in time (*i.e.* postponed) to its due-dates as close as possible. The tabu search heuristic is applied to improve the initial sequence. In each tabu search iteration, all jobs are scheduled as early as possible, and then right shifted in time as in the initial solution. The algorithm is stopped when the stopping criteria is met.

The following notations are used in both algorithms:

3.1.3.1 Algorithm I development

Input parameters

N = Number of jobs.

J = Set of jobs.

$i, j \in J$ where $i, j = 1,2,3......, N$.

E_i = Earliness cost of the i^{th} job (cost/ unit time).

t_i = Processing time of the i^{th} job.

$Y_i = \dfrac{E_i}{t_i}$.

D_i = Due-date of the i^{th} job.

System variables

Z = Total earliness cost.

Z' = Total earliness cost of the current best sequence.

Z_{mj} =Total earliness cost after assigning job j to fill the imbedded idle time interval m.

k = The k^{th} imbedded idle period on machine (*i.e.* $k = 1,2,3,\ldots.$).

k' = The k'^{th} imbedded idle period on machine (*i.e.* $k' =1, 2, 3,\ldots..$) in the current best sequence.

k^{mj} = The k^{mj} imbedded idle period of machine (*i.e.* $k =1, 2, 3,\ldots..$) after assigning job j to fill the imbedded idle interval m^{th}.

$\bar{k}$ = Number of idle time intervals on machine.

$\bar{k}'$ = Number of idle time intervals on machine in the current best sequence.

$\bar{k}^{mj}$ = Number of idle time intervals on machine after assigning job j to fill the imbedded idle interval m^{th}.

S_i = Starting time of the i^{th} job.

S_i'= Starting time of the i^{th} job in the current best sequence.

S_i^{mj} =Starting time of the i^{th} job after assigning job j to fill the imbedded idle time slot m.

C_i = Completion time of the i^{th} job.

C'_i = Completion time of the i^{th} job in the current best sequence.

C_i^{mj} = Completion time of the i^{th} job after assigning job j to fill the imbedded idle interval m.

$TIME$ = Latest available time on machine.

$\Delta_k = (L_k, U_k)$, the k^{th} interval of imbedded idle time on machine (*e.g.*, $\Delta_1 = (10,20)$, implies the first imbedded idle time on the machine occurs in the period between the 10^{th} time unit and the 20^{th} time unit).

$\Delta'_k = (L'_k, U'_k)$, the k^{th} interval of imbedded idle time on machine in the current best sequence.

$\Delta_k^{mj} = (L^{mj}_k, U^{mj}_k)$, the kth interval of imbedded idle time on machine after assigning job j to fill the imbedded idle interval m.

L_k = The beginning time of the kth imbedded idle time period.

U_k = The end time of the kth imbedded idle time period.

σ = Set of unscheduled jobs.

π = Set of scheduled jobs.

δ_i = Set of jobs that conflict with job i based on the ideal schedule.

β_k = Set of jobs that are able to fill in Δ_k (i.e. $j \in \beta_k$, if j is a job scheduled before Δ_k and $D_j > L_k$). For example, a sequence is $1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 6$ and there is an imbedded idle time between jobs 2 and 5. Thus L_k is at the completion time of job 2 and U_k is at the starting time of job 5. Let $D_4 > L_k$, then job $4 \in \beta_k$. Job 4 is able to fill in Δ_k, the new schedule is $1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$. In case where $t_4 > U_k - L_k$, then it will cause a leftward shift of jobs 1, 3, 2 on the schedule.

Algorithm I:

Step 0. Initialization

0a. Set $k = 0$, $\sigma = J$, $\pi = \phi$, $TIME = \max_{i \in \sigma}\{D_i\}$.

0b. For each $i \in \sigma$, set $\delta_i = \phi$, $Y_i = \frac{E_i}{t_i}$.

Step 1. Construct an ideal solution

1a. For each $i \in \sigma$, set $C_i = D_i$, and $S_i = C_i - t_i$.

1b If any conflict exists between jobs (i.e. for each $i \in \sigma$, $j \in \sigma - \{i\}$, $S_j \le S_i \le C_j$ or $S_j \le C_i \le C_j$), go to step2, otherwise stop.

Step 2. Let P be the set of jobs $i \in \sigma$ with $D_i \ge TIME$ (i.e., $P = \{D_i \ge TIME, i \in \sigma\}$).

Select job i^* where $Y_{i^*} = \max_{k \in P}\{Y_k\}$. If $P = \phi$, then select job i^*, which $D_{i^*} = \max_{k \in \sigma}\{D_k\}$. Break ties arbitrarily.

Step 3. Set $C_{i^*} = \min\{TIME, D_{i^*}\}$, $S_{i^*} = C_{i^*} - t_{i^*}$.

For each $j \in \sigma - \{i^*\}$, if job j conflicts with job i^*, set $\delta_{i^*} \leftarrow \delta_{i^*} + \{j\}$.

Step 4. If $Y_j \le Y_{i^*}$, for all $j \in \delta_{i^*}$, go to step *7* (follows propositions *1, 3, 4*), otherwise go to step *5*.

Step 5. Select job $j^* \in \delta_{i^*}$ where $D_{j^*} = \max_{k \in \delta_{i^*}}\{D_k\}$ and $Y_{j^*} \ge Y_{i^*}$. Break ties arbitrarily.

Step 6. Find the relationship between i^* and j^* based on local optimality conditions.

6a. Set $T_{i^*} = \max\{TIME - D_{i^*}, 0\}$.

6b. For $Y_{i^*} \le Y_{j^*}$, $D_{i^*} > D_{j^*}$, and $T_{i^*} \ge 0$,

if $(D_{j^*} - D_{i^*} + T_{i^*}) \le \dfrac{t_{j^*}E_{i^*} - t_{i^*}E_{j^*}}{(E_{i^*} + E_{j^*})}$ (follows propositons 2 for $T = 0$, and 5 for $T > 0$), set $\delta_{i^*} \leftarrow \delta_{i^*} - \{j\}$ and go to step 4, otherwise set $C_{i^*} = D_{i^*}$, $S_{i^*} = C_{i^*} - t_{i^*}$ (i.e., set job i^* back to ideal form), $i^* = j^*, \delta_{i^*} = \phi$ (i.e., job j^* is selected to schedule) and go to step 3.

Step 7. Schedule job i^* on machine

7a. Check the idle time period. If the completion time of job i^* is not at *TIME*, then an imbedded idle time is incurred (i.e. If $C_{i^*} < TIME$, set $k = k+1$, $L_k = C_{i^*}$, and $U_k = TIME$, $\Delta_k = (L_k, U_k)$), otherwise there is no imbedded idle slot recorded by the algorithm.

7b. Set *TIME* as the starting time of the job that is just scheduled (i.e., set $TIME = S_{i^*}$).

7c. Delete job i^* from the set of unscheduled jobs and add i^* to the set of scheduled jobs (i.e. $\sigma \leftarrow \sigma - \{i^*\}$ and $\pi \leftarrow \pi + \{i^*\}$).

7d. If all jobs are scheduled ($|\sigma| = 0$), then go to step 8, otherwise go back to step 2.

Step 8. Calculate total earliness cost, $Z = \sum_{i=1}^{N}[E_i * \max(0, D_i - C_i)]$.

Step 9. Consider the solution from step 8 as the best solution. Set $S'_i = S_i$, $C_i' = C_i$ for $i \in J$. Set $k' = k$, $\Delta'_m = \Delta_m$, and $Z' = Z$.

Step 10. Check existing imbedded idle periods. If there is no imbedded idle period (i.e. $k' = 0$) in the sequence, then the sequence is the best sequence obtained by the algorithm and go to step 14, otherwise go to step 11 since this sequence can be improved.

Step 11. Search for the new lower cost sequence by assigning job to fill existing imbedded idle time periods. Set the best sequence (i.e. S'_i, C'_i, and Z') be the current sequence, called R

Repeat the following steps (*11a to 11e*) for any existing imbedded idle time interval, Δ'_m, where $m = 1, 2, 3, \ldots, k'$. Start with $m = 1$ and advance to the last interval k.

11a. Set β_m as the set of jobs that are scheduled before Δ'_m and are able to fill in Δ'_m (i.e. $j \in \beta_m$, where $C'_j < L'_m$, and $D_j \geq L'_m$).

Explore filling the interval Δ'_m by each job $j \in \beta_m$ by repeating the following steps (*11b to 11e*) for each job $j \in \beta_m$ and starting each time with sequence R.

11b. Assign job j to fill Δ'_m (i.e. set $C_j = min\ (D_j, U'_m)$, $S_i = C_i - t_i$). If starting time of job j after filling in Δ'_m is less than L'_m (i.e., the imbedded idle slot is not enough to fit job j), jobs scheduled before job j are leftward shifted.

11c. Reschedule all jobs that are scheduled before job j, since they may not longer be in the best sequence based on the propositions. Let A be the jobs scheduled before job j (i.e. $A = \{ i\ /\ C_i < C_j \}$). Set all $i \in A$ in an ideal forms (i.e. completion time at due-date) and consider all $i \in A$ as the unscheduled jobs by setting $\sigma = A$. Set $TIME = S_j$ and repeat step 2 to step 8 for all jobs in σ.

11d. Calculate $Z_{mj} = \sum_{i=1}^{N} [E_i * max(0, D_i - C_i)]$, where Z_{mj} is the total earliness cost after filling job j into the imbedded idle time Δ'_m.

11e. Call the sequence obtained by filling β_m by job j as $S(\beta_m, j)$.

Step 12. Select the best sequence $S(\beta_m, j')$ resulting from the exploration sequences obtained by trying to fill Δ'_m in R by each job $j \in \beta_m$ (i.e. $Z_{mj'} = \min_{\forall mj} \{Z_{mj}\}$).

Step 13. If the total cost of the sequence obtained from step 12 is less than the cost of the current sequence (i.e. $Z_{mj'} < Z'$), then set the new obtained sequence as the best sequence (i.e. set $Z' = Z_{mj'}$, $S'_i = S_i^{m'j'}$ and $C'_i = C_i^{m'j'}$ for all $i \in J$, and $k' = k^{m'j'}$ and $\Delta'_m = \Delta_m^{m'j'}$ for $m = 1,2,\ldots,k'$) and go to step 10 otherwise, go to step 14.

Step 14. Output the best sequence determined.

Numerical example 3.1

To illustrate the steps of the algorithm, consider the problem described in Table 3.1 below. The objective is to minimize the total earliness penalty. A partial listing of the execution of the algorithmic steps follows. The complete solution procedure is presented algorithmically in Appendix C.

Table 3.1 Numerical example 3.1

Job	1	2	3	4	5	6	7	8
t_i	3	4	9	10	10	5	7	2
D_i	80	80	75	66	64	60	50	43
E_i	6	4	9	2	3	7	5	1
Y_i	2	1	1	0.2	0.3	1.4	0.71	0.5

Step 0. Initialization

0a. Set $k=0$, $\sigma = \{1,2,3,4,5,6,7,8\}$, $TIME = 80$.

0b. For each $i \in \sigma$, set $\delta_i = \phi$, $Y_i = \frac{E_i}{t_i}$ (i.e., $Y_1=2$, $Y_2=1$, $Y_3=1$, $Y_4=0.2$, $Y_5=0.3$, $Y_5=0.3$, $Y_6=1.4$, $Y_7=0.71$, $Y_8=0.5$).

Step 1. Construct an ideal solution

1a. $C_1 = 80$, $S_1 = 77$, $C_2 = 80$, $S_2 = 76$, $C_3 = 75$, $S_3 = 66$, $C_4 = 66$, $S_4 = 56$,

$C_5 = 64$, $S_5 = 54$, $C_6 = 60$, $S_6 = 55$, $C_7 = 50$, $S_7 = 43$, $C_8 = 43$, $S_8 = 41$ (see Fig. 3.7).

1b. There are conflicts between jobs. Go to step 2.

Step 2. Select $i^* = 1$ $(D_1 = TIME)$.

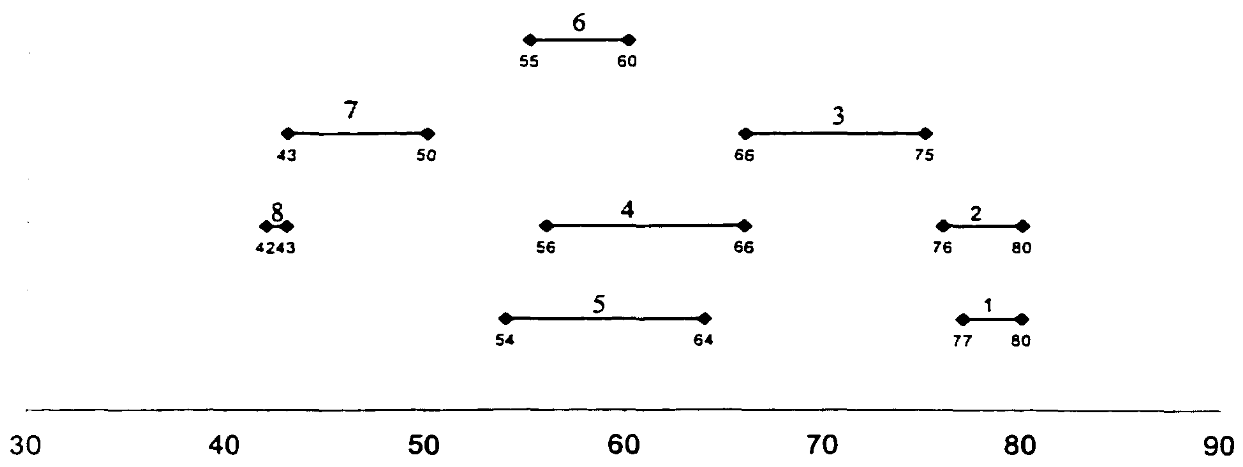


Figure 3.7 Ideal solution for example 3.1

Step 3. Set $C_1 = 80$, $S_1 = 77$, and $\delta_1 = \{2\}$ (since $D_2 > S_1$).

Step 4. Since $Y_1 > Y_2$, go to Step 7 .

Step 7a. No imbedded idle period.

Step 7b. $TIME = 77$. Schedule job 1 at $C_1 = 80$, $S_1 = 77$ (see Fig. 3.8).

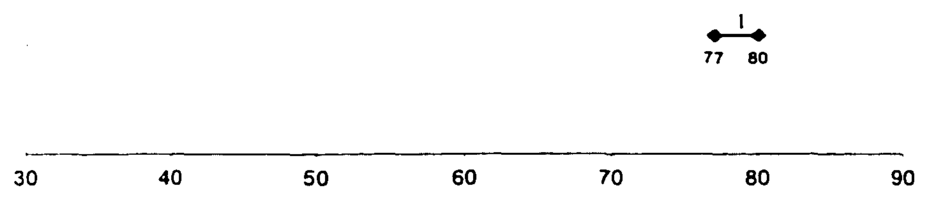


Figure 3.8 Schedule job 1 of example 3.1

Step 7c. $\sigma = \{2,3,4,5,6,7,8\}$, $\pi = \{1\}$

Step 7d. $|\sigma| \neq 0$ go to step 2.

Step 2. Select $i^* = 2$ $(D_2 > TIME)$.

Step 3. Set $C_2 = 77$, $S_2 = 73$, and $\delta_2 = \{3\}$

Step 4. Since $Y_2 = Y_3$, go to Step 7 .

Step 7a. No imbedded idle period.

Step 7b. $TIME = 73$. Schedule job i^* at $C_2 = 77$, $S_2 = 73$ (see Fig. 3.9).

Step 7c. $\sigma = \{3,4,5,6,7,8\}$, $\pi = \{1,2\}$

Step 7d. $|\sigma| \neq 0$ go to step 2.

Algorithm is repeated until all jobs are scheduled. The sequence obtained for example 3.1 is shown in Fig. 3.10

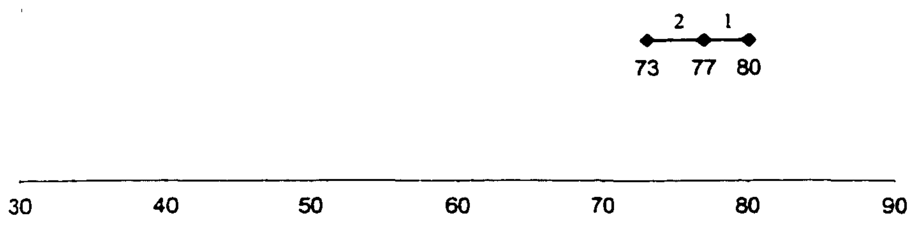


Figure 3.9 Schedule job 2 of example 3.1

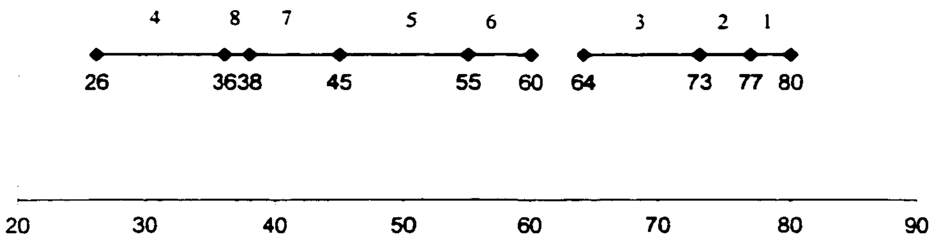


Figure 3.10 A schedule for example 3.1

After job 4 is scheduled, the problem contains $|\sigma| = 0$. Then, the algorithm moves to step 8 for calculating the total weighted earliness cost.

Step 8. Calculate total weighted earliness cost , $Z = 147$ unit cost.

Step 9. Keep the solution from step 8 as the current best solution by setting

$S'_1 = 77$, $C'_1 = 80$, $S'_2 = 73$, $C'_2 = 77$, $S'_3 = 64$, $C'_3 = 73$, $S'_4 = 26$, $C'_4 = 36$, $S'_5 = 45$, $C'_5 = 55$, $S'_6 = 55$, $C'_6 = 60$, $S'_7 = 38$, $C'_7 = 45$, $S'_8 = 36$, $C'_8 = 38$, $K' = 1$, $\Delta'_1 = (60, 64)$, $Z' = 147$.

Step 10. There is an imbedded idle time, $\Delta'_1 = (60, 64)$, then go to step 11.

Step 11a. Set β_1 as the set of jobs that are scheduled before Δ'_1 and are able to fill in Δ'_1 , $\beta_1 = \{5\}$.

Step 11b. Assign job 5 to fill $\Delta'_1 = (60, 64)$. $C_5 = 64$, $S_5 = 54$. Since $S_5 < 60$, thus jobs 6,7,8 and 4 must be leftward shifted.

Step 11c. Set jobs 6,7,8 and 4 in an ideal form, set $\sigma = \{4,6,7,8\}$ and set *TIME* . $= S_5$ (i.e. 54). The schedule is as shown in Fig.3.11.

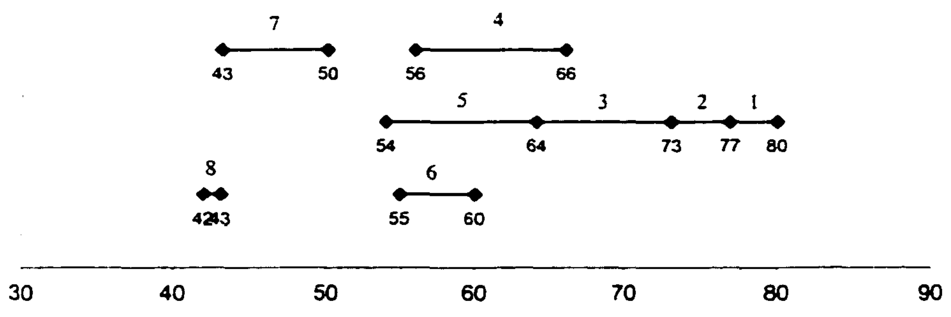


Figure 3.11 Assign job 5 to fill in an imbedded idle time period Δ'_1 = *(60, 64)*

Now, the algorithm repeats step 2 to step 8 for rescheduling jobs 4, 6, 7, 8. The final schedule is as shown in Fig. 3.12. The total cost of final schedule is 130 which is less than the penalty before assigning job 5 to fill in Δ'_1 = *(60, 64)*. Since the new schedule does not contain any imbedded idle period, the algorithm stops and the new schedule (Fig. 3.12) is the best schedule for example 3.1 obtained by algorithm I.

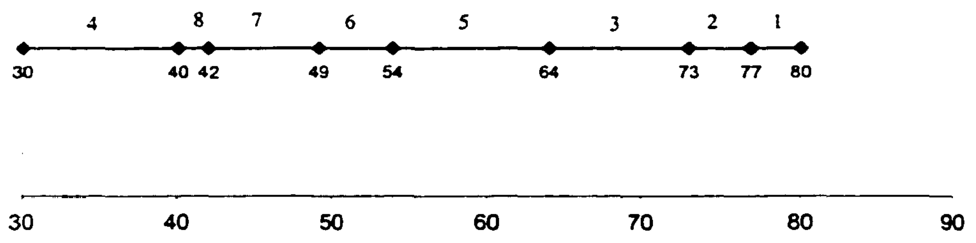


Figure 3.12 The best schedule for example 3.1 by algorithm I

3.1.3.2 Algorithm II development

The following additional notations and definitions are used in the presentation of the algorithm.

Definitions

Tabu size: A parameter that designates the number of iterations that a pair of jobs is forbidden to be swapped.

Aspiration criteria: A criteria that the tabu status is overridden when the swapping between the declared tabu pair of jobs yields a lower cost that is better than the lowest cost found so far (i.e., the current best solution).

Input parameters

I_{max} = Maximum number of iterations allowed in tabu search (stopping criteria).

n_{max} =The number of iterations allowed after the current best solution is found (stopping criteria).

tabu_size = The number of iterations that a pair of jobs is forbidden to be swapped.

System variables

ETIME = Earliest available time of machine.

R_{ij} = The sequence of jobs after swapping positions between job i and j.

$Z(R_{ij})$ = Total earliness cost of the sequence after swapping positions between job i and j.

$S_i(R_n)$ = Starting time of job i in sequence R_n.

$C_i\,(R_n)$ = Completion time of job i in sequence R_n.

γ = Set of jobs which have been shifted to the right.

Algorithm II (Tabu search)

Phase I Initial solution construction

Step 0. Set $\sigma = J$, $\pi = \phi$. $ETIME = 0$, $TIME = \max_{i \in \sigma}\{D_i\}$.

Set the *tabu_size*, n_{max}, and I_{max}. Set $C_i = D_i$ for all $i \in J$.

Step 1. Select $i^* \in \sigma$ with the earliest due-date (i.e. $D_{i^*} \le D_j$, $\forall j \in \sigma$). Break ties arbitrarily.

Step 2. Schedule i^* with $S_{i^*} = ETIME$, $C_{i^*} = S_{i^*} + t_{i^*}$. Set $\sigma \leftarrow \sigma - \{i^*\}$, $\pi \leftarrow \pi + \{i^*\}$ and $ETIME = C_{i^*}$.

Step 3. If $|\sigma| \neq 0$, go to step 1, otherwise go to step 4.

Step 4. While maintaining feasibility, shift right in time each scheduled job to its due-date as close as possible.

4a. Set $\gamma = \phi$. Let P be the set of jobs not yet shifted right in time.

4b. For the last job i (i.e. $\max_{j \in P}\{C_j\}$) , if the $C_i < D_i$, then shift job i to its due-date (i.e. $C_i = D_i$, $S_i = C_i - t_i$,$TIME = S_i$), set $\gamma \leftarrow \gamma + \{i\}$ and set $P \leftarrow P - \{i\}$, otherwise leave C_i, S_i unchanged and go to step 5.

4c. For the next latest job i (i.e. $\max_{j \in P}\{C_j\}$), if the $C_i < D_i$ and $C_i < TIME$, set $C_i = \min(D_i, TIME)$, $S_i = C_i - t_i$,$TIME = S_i$, set $\gamma \leftarrow \gamma + \{i\}$ and $P \leftarrow P - \{i\}$.

4d. Repeat step 4c until all jobs either are right shifted or left unchanged. Go to step 5.

Step 5. Calculate total earliness cost, $Z = \sum_{i=1}^{N} [E_i * \max(0, D_i - C_i)]$.

Step 6. Consider the solution from step 5 as the current best solution, by setting $S'_i = S_i$, $C_i' = C_i$ for $\forall i \in J$, and setting $Z' = Z$.

Phase II Solution improvement (Tabu search).

Step 7. Consider each job pair i,j where $i \neq j$, $i, j \in J$ are a candidate pair to be swapped and place the pairs in a candidate list.

Step 8. Evaluation of all job pairs in the candidate list.

Repeat the following procedure for all candidate pairs i,j, where R_c is the current sequence.

8a. Swap the positions of job i, j from current sequence R_c. to obtain the new sequence R_{ij} where the R_{ij} is the sequence obtained from swapping jobs i and j.

8b. Schedule all jobs on R_{ij} as early as possible (i.e. set the start time of the first job in R_{ij} at time zero, and the starting time of the next job in R_{ij} is at the completion time of the immediate preceding job in R_{ij}, and so on). If R_{ij} is infeasible (i.e., some jobs are tardy), then discard R_{ij} and go to step 8e.

8c. Employ step 4 to right shift in time each scheduled job in R_{ij} to its due date as close as possible.

8d. Calculate the total weighted earliness cost $Z(R_{ij})$ of the R_{ij}.

8e. Swap the jobs i, j back to their original positions to obtain the current sequence R_c.

Step 9. Select the best pair to be swapped. The sequence R_{ij} that provides the lowest total weighted earliness cost is selected to be the new sequence R_n.

Step 10. If the new swapping pair is tabu pair and does not satisfy the aspiration criteria, it is disregarded and eliminated from the candidate list. Go to step 8. If the new swapping pair is not a tabu pair or satisfies the aspiration criteria, go to step 11.

Step 11 Set $R_c = R_n$. If the $Z(R_n) < Z'$, set $Z' = Z(R_n)$, $S'_i(R_c) = S_i(R_n)$, $C'_i(R_c) = C_i(R_n)$ for $\forall i \in J$.

Step 12. Declare the tabu status for the recent swapped pair.

Step 13. Check the stopping criteria.

13a. If the Z' has not decreased for the last n_{max} iterations, stop otherwise go to step 13b.

13b. If the maximum number of iterations is met go to step 14, otherwise go to step 7.

Step 14. Output the best sequence determined.

Numerical example

In this section, the example problem 3.1 is solved using algorithm II. The algorithmic steps involved in solving the problem are as follows:

Phase I Initial solution construction

Step 0. Set $\sigma = J$, $\pi = \phi$. $ETIME = 0$, $TIME = 80$, $tabu\_size = 3$, $n_{max} = 5$, $I_{max} = 30$.

Step 1. Select $i^* = 8$.

Step 2. $S_8 = 0$, $C_8 = 2$. $\sigma = \{1,2,3,4,5,6,7\}$, $\pi = \{8\}$ and $ETIME = 2$.

Step 3. $|\sigma| \neq 0$, go back to step 1.

Step 1 to step 3 of the algorithm are repeated until all jobs are scheduled. After all jobs are scheduled, the schedule of jobs is as shown in Figure 3.13. Next, the algorithm proceeds to step 4.

Step 4. While maintaining feasibility, shift right in time each scheduled job to its due-date as close as possible (see Fig. 3.14).

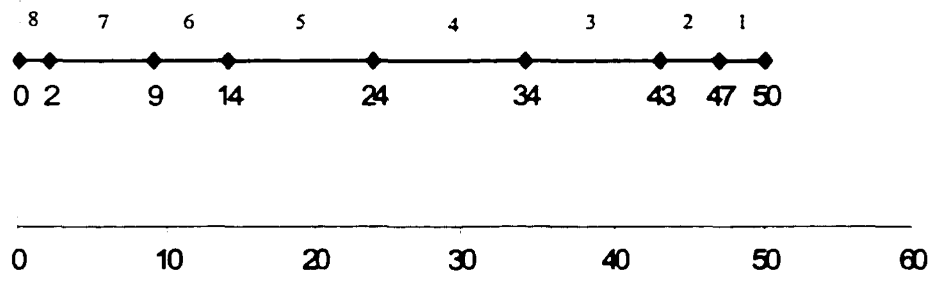


Figure 3.13 Initial solution of example 3.1 (i.e. jobs are scheduled as early as possible)

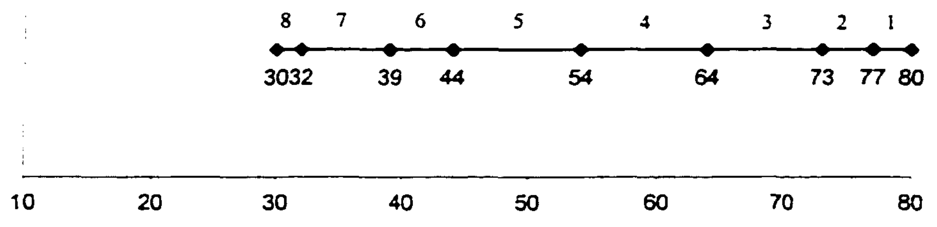


Figure 3.14 Initial solution of example 3.1 after shifting right in time

Step 5. Total earliness cost $Z = 244$.

Step 6. Consider the solution from step 5 as the current best solution, by setting $S'_1 = 77$, $C'_1 = 80$, $S'_2 = 73$, $C'_2 = 77$, $S'_3 = 64$, $C'_3 = 73$, $S'_4 = 54$, $C'_4 = 64$, $S'_5 = 44$, $C'_5 = 54$, $S'_6 = 39$, $C'_6 = 44$, $S'_7 = 32$, $C'_7 = 39$, $S'_8 = 30$, $C'_8 = 32$, $Z' = 244$.

Phase II Solution improvement (Tabu search)

Step 7. Generate candidate pair list = *{(1,2), (1,3), (1,4), (1,5), (1,6), (1,7), (1,8), (2,3), (2,4), (2,5), (2,6), (2,7), (2,8), (3,4), (3,5), (3,6), (3,7), (3,8), (4,5), (4,6), (4,7), (4,8), (5,6),(5,7), (5,8), (6,7), (6,8), (7,8)}*

Step 8. Evaluate all job pairs in the candidate list.

8a Current sequence $R_c = \{8 \rightarrow 7 \rightarrow 6 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1\}$. Swap the first job pair in the candidate list *(1,2)* to obtain the sequence $R_{12} = \{8 \rightarrow 7 \rightarrow 6 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 1 \rightarrow 2\}$.

8b Schedule all jobs on R_{12} as early as possible. The schedule is as shown in Fig.3.15.

8c. While maintaining feasibility, shift right in time each scheduled job to its due-date as close as possible. The schedule is shown as in Fig. 3.16, and the algorithm proceeds to step 8d.

8d Total weighted earliness cost $Z_{12} = 254$.

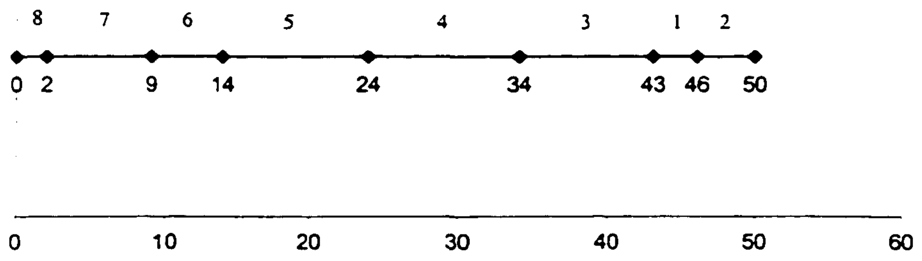


Figure 3.15 Schedule of jobs on R_{12} (i.e. jobs are scheduled as early as possible)

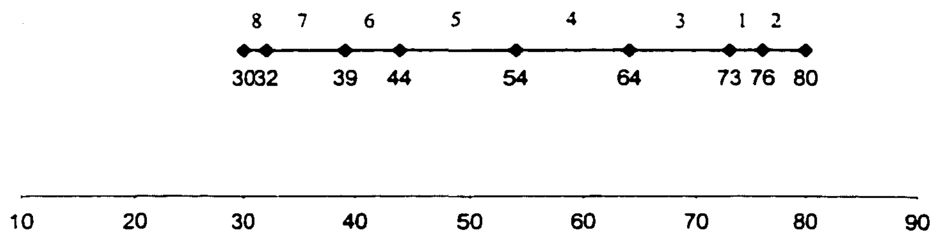


Figure 3.16 Schedule of jobs on R_{12} after shifting right in time.

8e Swap the jobs *i, j* back to their original positions to obtain the current sequence R_c, where $R_c = \{8 \rightarrow 7 \rightarrow 6 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1\}$.

The algorithm repeats step 8 for the rest of the job pairs in the candidate list. Some job pairs are ignored after the algorithm finds that they create infeasible solutions. Infeasibilty occurs if at least a job is tardy. For example, the swapping of jobs *1* and *8* produces an infeasible solution. The schedule of $R_{18} = \{1 \rightarrow 7 \rightarrow 6 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 8\}$ is infeasible and candidate the pair (1,8) is not considered in the algorithm, since job 8 is tardy (see Fig. 3.17).

Step 9. Select the best pair to be swapped. The swapping between jobs 4 and 5, (i.e., (4,5)), provides the lowest cost among the other pairs in the candidate list, $Z(R_{45}) = 232$.

Set the new sequence $R_n = R_{45} = \{8 \rightarrow 7 \rightarrow 6 \rightarrow 4 \rightarrow 5 \rightarrow 3 \rightarrow 2 \rightarrow 1\}$.

Step 10. Go to step 11, since the pair (4, 5) does not have tabu status.

Step 11. Set $R_c = R_n$. Since the $Z(R_{45}) < Z'$, then set $Z' = Z(R_{45}) = 232$, $S'_i = S_i(R_{45})$, $C'_i = C_i(R_{45})$ for all $i \in J$.

Step 12. Declare tabu status for the pair (4,5). Jobs 4 and 5 can not be swapped in the next three iterations, since the tabu size is three.

Step 13. Since the stopping criteria is not yet met, the algorithm goes back to step 7.

The algorithm continues to iterate until the stopping conditions is met. In this example, after the stopping criteria is met, the best sequence obtained is $4 \rightarrow 8 \rightarrow 7 \rightarrow 6 \rightarrow 5 \rightarrow 3 \rightarrow 2 \rightarrow 1$. This corresponds to the same sequence obtained by algorithm I (see Fig. 3.12).

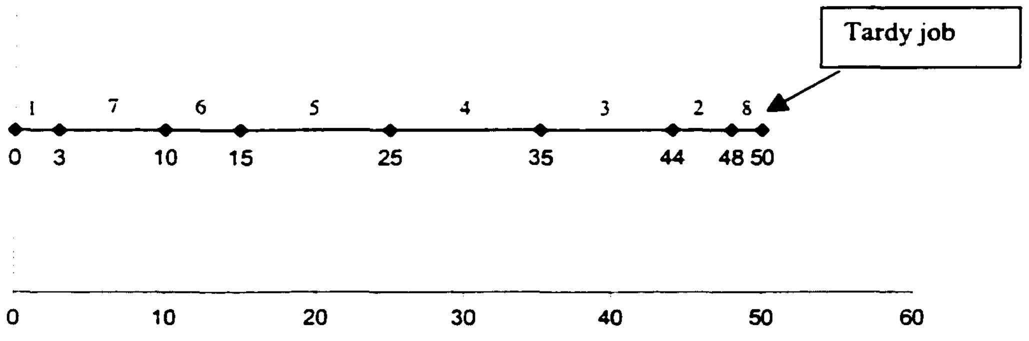


Figure 3.17 Job 8 is tardy on R_{18}

3.2 Single machine problem with sum of the weighted earliness and weighted tardiness cost minimization

In this section, the no tardy job constraint from the previous section is relaxed. This makes it possible for jobs to be tardy. The problem is to schedule a set of jobs to minimize the sum of weighted earliness and weighted tardiness costs. Jobs may have unequal weighted earliness and weighted tardiness penalties. The problem was first modeled mathematically. The mathematical formulation is as given in section 3.2.1

3.2.1 Problem formulation

The mathematical model of the single machine problem with the objective of minimizing the weighted sum of earliness and tardiness penalties is as given below

Objective function

$$Min. \sum_{i=1}^{N} (E_i * \max(0, D_i - C_i) + W_i * \max(0, C_i - D_i))$$

This objective function can be transformed to the following function

$$Min. \sum_{i=1}^{N} (E_i * A_i + W_i * B_i) \qquad (1)$$

Subject to

$C_i - S_i = t_i$	for $\forall i$	*(Processing time constraints)*	(2)
$C_j - C_i + \alpha(1 - X_{ij}) \geq t_j$	for $\forall i, j$	*(Disjunctive constraints)*	(3)
$C_i - C_j + \alpha X_{ij} \geq t_i$	for $\forall i, j$	*(Disjunctive constraints)*	(4)
$A_i \geq D_i - C_i$	for $\forall i$		(5)
$B_i \geq C_i - D_i$	for $\forall i$		(6)
$A_i \geq 0$	for $\forall i$		(7)
$B_i \geq 0$	for $\forall i$		(8)
$S_i \geq 0$	for $\forall i$		(9)
$X_{ij} \in \{0, 1\}$, integer,	for $\forall i$		(10)

where N = *number of jobs,*

S_i = *starting time of job i,*

C_i = *completion time of job i,*

t_i = *processing time of job i,*

D_i = *due-date of job i,*

A_i = *the amount of time of job i completed before its due-date (i.e.* $A_i = D_i - C_i$*).*

B_i = *the amount of time of job i completed after its due-date (i.e.* $B_i = C_i - D_i$*).*

α = *large positive number,*

E_i = *earliness cost of job i (cost /unit of time),*

W_i = *tardiness cost of job i (cost /unit of time),*

$$X_{ij} = \begin{cases} 1, & \text{if job } i \text{ precedes job } j, \\ 0, & \text{otherwise.} \end{cases}$$

Constraints *(2)* express that processing time of a job equals to the difference between its starting time and its completion time. Constraints *(3)* and *(4)* ensure that no two jobs can be processed simultaneously. Constraints *(5)*, *(6)*, *(7)*, *(8)* linearize the nonlinear objective function of the problem. Constraints *(9)* express that starting time of job *i* must be positive. Integrality requirement on X_{ij} is described in *(10)*.

3.2.2 Heuristics for early/tardy problem

As stated earlier, mathematical programming procedure for solving the earliness/tardiness problem in single machine requires excessive computational time for solving practical size problems (i.e. more than 15 job problems). Thus the mathematical programming procedure is not a practical approach for solving the problem. The development of efficient heuristics that generate optimal or near-optimal solutions is attractive. Two heuristic solution methodologies, algorithm III and algorithm IV are developed in this section. Heuristic algorithms III and IV are respectively the extension of heuristic algorithms I and II developed for minimizing the earliness penalties in the previous sections.

Algorithm III is extended from algorithm I. It is applied to problems that permit job tardiness. For example, if a set of jobs can not be scheduled without violating due-dates of some jobs, then it is impossible to apply algorithm I to such a problem since algorithm I requires that all jobs must be completed by their due-dates. In other words, algorithm I is applicable only to problems in which it is feasible to complete all jobs by their due-dates. Therefore, an algorithm is needed to solve the more general-purpose problem that involves both earliness and tardiness in job completion times. In this research, algorithms III and IV are developed to solve problems that permit both earliness and tardiness.

In algorithm III, algorithm I is first employed to build a schedule backward without allowing for tardiness. However, the solution obtained from algorithm I may be infeasible (i.e., to complete all jobs by their due-dates, the algorithm could schedule some jobs to start before time zero). Of course, scheduling of any job before time zero yields infeasible solution. Shifting the schedule to the right in time to avoid starting any job before time would imply that some jobs will be tardy. To obtain a feasible solution, the entire sequence of jobs is shifted right in time until the starting time of the first job on the sequence is at time zero. Now, a feasible solution is obtained and some jobs are tardy. At this point, the algorithm starts to search for a set of jobs that should be tardy jobs by exploring the application of pairwise interchange method. The job pairs identified to yield the lowest cost after interchanging is selected to be interchanged. The pairwise interchange method is employed until no more interchanges that can provide lower cost solution is found. Now a local optima is achieved and the obtained schedule consists of two sets of jobs, a set of tardy jobs and non-tardy jobs. After interchanging, the obtained schedule may no longer be the best schedule in terms of earliness cost minimization point of view, since jobs in the new schedule may not be ordered based on the propositions presented in section 3.1.2. Thus, the solution may be improved by applying algorithm I to reschedule all jobs again.

To reapply algorithm I to the schedule obtained from pairwise interchange, algorithm I can not be simply employed as before. Algorithm I is based on the due-date of jobs. For this case, if real due-dates of all jobs are still used in algorithm I, then algorithm I will always generate an infeasible solution (i.e., starting time of some jobs on the sequence may occur before time zero) which is the same as the infeasible schedule initially generated. To avoid infeasible solution, we need to set virtual due-dates for some jobs. We know that the schedule obtained from the pairwise interchange consists of a set of tardy jobs and a set of non-tardy jobs. We can set virtual due-dates for the tardy jobs at their completion time in the current schedule (i.e., the completion time on the schedule obtained from pairwise interchange). In other word, we have agreed to let this set of jobs to be tardy jobs. For the non- tardy jobs, the due-dates are left unchanged (i.e., kept the same due date as contained in the original data). For example, suppose the real due-date of job i, D_i , is *10*, but after pairwise interchange, the job i is completed at time *12*. We consider the time *12* as the virtual due-date, D^*_i. of job i. Based on the virtual due-dates of the tardy jobs and the real due-dates of non-tardy jobs, a feasible schedule (i.e.,starting time of the first job of the schedule is non-negative value) can be built by algorithm I.

The new sequence obtained after reapplying algorithm I consists of two set of jobs, a set tardy jobs and a set of non-tardy jobs. Non-tardy jobs in the new schedule are now sequenced based on minimizing the earliness cost. If the new schedule obtained now is different from the schedule before reapplying algorithm I, pairwise interchange method is again used to search for a new improved set of tardy jobs and then algorithm I is employed to reschedule for minimizing earliness cost of non - tardy jobs. This algorithm is repeated until the criteria for termination are met.

3.2.2.1 Algorithm III development

The following additional variable and parameter definitions are provided for algorithm III.

updated_cost : the total cost of the sequence in each iteration.

I_c = Iteration count.

The algorithm appears as follows:

Algorithm III

Step 0. Employ algorithm I to generate a solution (i.e. the solution may be infeasible because the starting time of some jobs on the schedule may be negative value). Set iteration count $I_c = 1$. If the schedule is feasible, consider this solution as the current best solution, by setting $S'_i = S_i$, $C'_i = C_i$ for $i \in J$, and setting $Z' = Z$ and set the current sequence, R_c , = the current best solution and go to step 2. Otherwise go to step 1.

Step 1. Shift right in time the entire sequence until the starting time of the first job is at time zero. Consider this solution as the current best solution, by setting $S'_i = S_i$, $C'_i = C_i$ for $i \in J$, and setting $Z' = Z$. Set the current sequence, R_c , = the current best solution.

Step 2. Consider each job pair i, j where $i \neq j$, and $i, j \in J$ in the current best sequence, $R_{c.}$ as a candidate pair to be interchanged. Set *updated_cost* = Z. Repeat the following procedure for all candidate job pairs i, j (i.e. evaluating all job pairs in the candidate list).

2a. Interchange the positions of jobs i, j in the current sequence, R_c, to the new sequence R_{ij} , where R_{ij} is the sequence obtained from interchanging jobs i and j.

2b. Calculate the sum of total weighted earliness and weighted tardiness cost

$$Z(R_{ij}) = \sum_{i=1}^{N} [E_i * max(0, D_i - C_i) + W_i * max(0, C_i - D_i)] \quad .$$

2c. Reset the jobs *i, j* back to their orginal positions in R_c.

Step 3. Select the best job pair to be interchanged. The sequence R_{ij} that provides the lowest cost is selected to be the new sequence R_n.

Step 4. 4a. If $Z(R_n) \leq Z'$, set $Z' = Z(R_n)$, $S'_i = S_i(R_n)$, $C'_i = C_i(R_n)$ for $i \in J$.

4b. If $Z(R_n) <$ *updated_cost*, set $R_c = R_n$, *updated_cost* $= Z(R_n)$ and delete the selected pair from the candidate list, and go to step 2 to search for a new job pairs to interchange to improve the solution.

4c. If no interchange jobs yield lower cost, interchange yields a lower cost solution, go to step 5.

Step 5. Set $I_c = I_c + 1$. Check the stopping criteria. If the maximum number of iterations is met (i.e. $I_c = I_{max}$), go to step 7, otherwise go to step 6.

Step 6. Re-apply algorithm I to search for a new sequence. The objective of this step is to reschedule the non-tardy jobs in the schedule obtained from pairwise interchange to minimize earliness cost, since non-tardy jobs in the schedule obtained from pairwise interchange may no longer be ordered based on the propositions in section 3.1.2.

6a. For job *j*, in R_c (i.e. schedule obtained from pairwise interchange), If its completion time is greater than its real due-date (i.e., tardiness jobs in schedule obtained from pairwise interchange), set its virtual due-date as its completion time (i.e. we agree to let this set of jobs to be tardy jobs).

6b. For job *j*, in R_c (i.e. schedule obtained from pairwise interchange), If its completion time is not greater than its real due-date (i.e. non-tardy jobs in schedule obtained from pairwise interchange), leave the due date unchanged (i.e. we agree to let this set of jobs to be non-tardy jobs).

6c. Repeat algorithm I by using the due-dates obtained from 6a and 6b to reschedule all jobs to minimize the earliness cost based on the propositions in section 3.1.2.

6d. Calculate the total cost of the new obtained sequence R_n .

6e. If $R_c \neq R_n$, set $R_c = R_n$. and *updated_cost* = $Z(R_n)$ and go to step 2. Otherwise (i.e., $R_c = R_n$) go to step 7.

Step 7. Output the current best sequence R_c based on S'_i and C'_i for all $i \in J$.

Numerical example 3.2

The problem to be scheduled is as given in Table 3.2 below.

Table 3.2 Numerical example 3.2

Job	1	2	3	4	5
t_i	5	5	3	6	3
E_i	10	8	5	5	4
W_i	17	10	7	10	10
D_i	20	20	17	15	10

Step 0. Employ algorithm I to generate an infeasible solution (i.e. starting time of the some jobs on the schedule is negative value) as in Fig. 3.18. Set iteration count $I_c = 1$.

Step 1. Shift right in time the entire schedule until the starting time of job 4 is at time zero (see Fig 3.19). The new schedule is set to be R_c ={ 4→5→2→3 →1}. At this point, job 1 is tardy. Consider this solution as the current best solution,$S'_1 = 17$, $C'_1 = 22$, $S'_2 = 9$, $C'_2 = 14$, $S'_3 = 14$, $C'_3 = 17$, $S'_4 = 0$, $C'_4 = 6$, $S'_5 = 6$, $C'_5 = 9$, and $Z' = 131$.

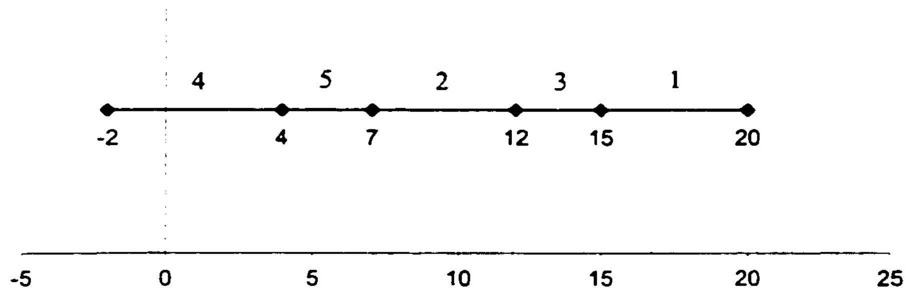


Figure 3.18 An infeasible solution built by algorithm I in example 3.2

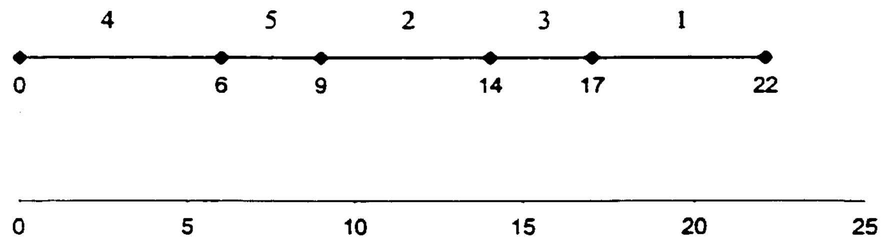


Figure 3.19 An initial schedule of example 3.2 (i.e. after right shifted in time)

Step 2.Candidate list of job pairs to be considered for interchange = { (1,2), (1,3), (1,4), (1,5),(2,3),(2,4),(2,5),(3,4),(3,5),(4,5) }.

Set *updated_cost = 131.*

Evaluation of all job pairs in the candidate list.

2a. Interchange the position of jobs 1and 2 in R_c. Job 2 is now tardy.

2b. Calculate the total cost $Z(R_{12}) = 129$.

2c. Reset jobs 1 and 2 back to their previous positions in R_c (i.e., R_c= $\{4 \to 5 \to 2 \to 3 \to 1\}$.

Step 2 is repeated for all job pairs in the candidate list.

Step 3. Select pair (1,2) as the best pair to be interchanged, since it provides the lowest cost compared to the other pairs in the list (see Fig.3.20).

Step 4. Since $Z(R_{12}) < Z'$, set $S'_1 = 17$, $C'_1 = 22$, $S'_2 = 9$, $C'_2 = 14$, $S'_3 = 14$, $C'_3 = 17$, $S'_4 = 0$, $C'_4 = 6$, $S'_5 = 6$, $C'_5 = 9$, *and* $Z' = 129$.

Since $Z(R_{12})$ < *updated_cost*, set $R_c = R_{12}$, *updated_cost = 129* and eliminate *(1,2)* from the candidate list and go to step 2.

Step 2. Evaluation of all job pairs R_c in the candidate list.

2a. Interchange the position of jobs 1, and 3 in R_c. Job 2 is still tardy.

2b. Calculate the total cost $Z(R_{13}) = 124$.

2c. Reset jobs 1 and 3 back to the previous positions in R_c.

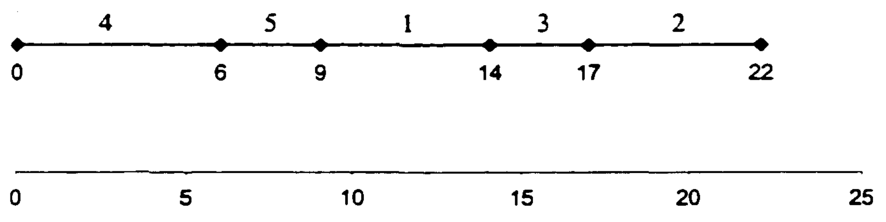


Figure 3.20 Schedule after interchanging between jobs 1 and 2 in example 3.2

Step 2 is repeated for all job pairs in the candidate list.

Step 3. Select pair (1,3) as the best pair to be interchanged, since it provides the lowest cost compared to the other pairs in the list (see Fig. 3.21).

Step 4. Since $Z(R_{13}) < Z'$, set $S'_1 = 12, C'_1 = 17, S'_2 = 17, C'_2 = 22, S'_3 = 9, C'_3 = 12, S'_4 = 0, C'_4 = 6, S'_5 = 6, C'_5 = 9,$ *and* $Z' = 124$.

Since $Z(R_{13}) <$ *updated_cost*, set $R_c = R_{13}$, *updated_cost* $= 124$ and eliminate $(1,3)$ from the candidate list and go to step 3.

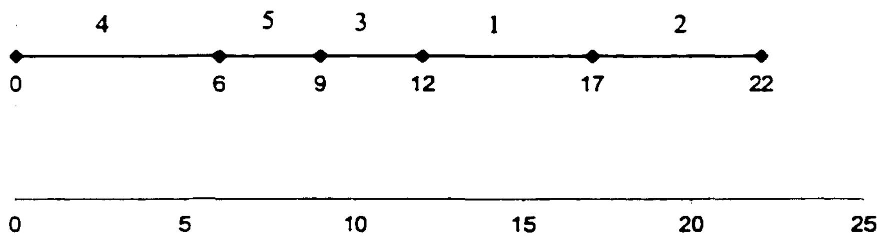


Figure 3.21 Schedule after interchanging between jobs 1 and 3 in example 3.2

Step 2. Evaluate all job pairs R_c in the candidate list.

Step 3. Select pair (4,5) as the best pair to be interchanged, since it provides the lowest cost compared to the other pairs in the list.

Step 4. Since $Z(R_{45}) >$ *updated_cost*, go to step 5.

Step 5. Set $I_c = 2$, and since $I_c < I_{max}$ (stopping criteria), go to step 6

Step 6. Re-apply algorithm I to search for a new improved sequence.

6b. For job 2, (tardy job), set its virtual due-date as its completion time ($D_2 = 22$).

6a. For jobs 1,3,4,5 (early jobs), leave their due-dates unchanged.

6c. Repeat algorithm I by using the due-dates obtained from 6a and 6b.

In algorithm I, the ideal solution is as shown in Fig. 3.22.

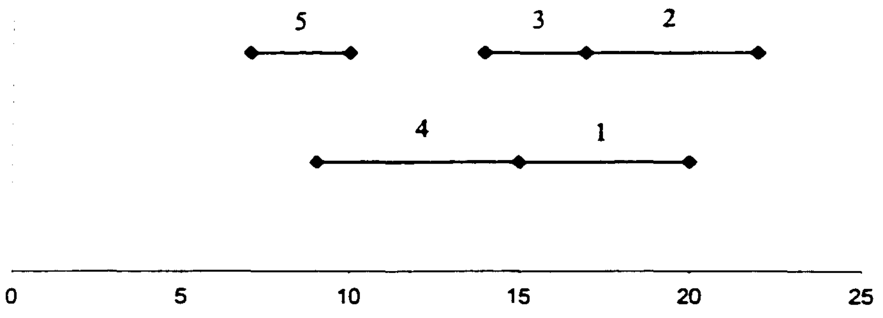


Figure 3.22 Ideal solution for example 3.1 where virtual due-date of job 2 is 22

The solution obtained after the application of algorithm I (see Fig. 3.23) is the same schedule as the one before applying algorithm I. This means the algorithm met the stopping criterion, so the algorithm moves to step 7.

Step 7. Output the best sequence = $\{4 \rightarrow 5 \rightarrow 3 \rightarrow 1 \rightarrow 2\}$ with $S'_1 = 12$, $C'_1 = 17$, $S'_2 = 17$, $C'_2 = 22$, $S'_3 = 9$, $C'_3 = 12$, $S'_4 = 0$, $C'_4 = 6$, $S'_5 = 6$, $C'_5 = 9$, and $Z' = 124$.

Algorithm IV is a slight modification of algorithm II. There are two differences between these two algorithms. First, the initial solution of algorithm IV is constructed using the dispatching rule of Ow and Morton [31]. Second, instead of discarding candidate pairs that create tardy jobs in step 8b of algorithm II, all candidate pairs

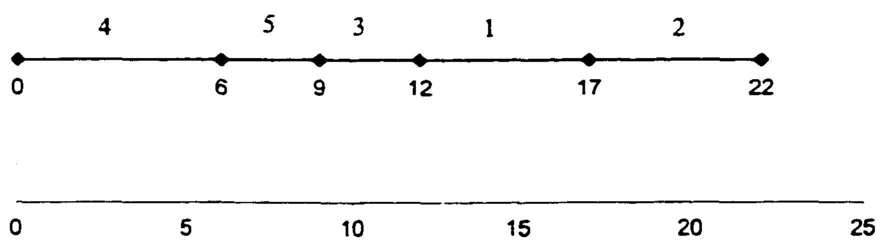


Figure 3.23 Final solution for example 3.2 by algorithm III

3.2.2.2 Algorithm IV development

including the ones that produce tardy jobs are considered in step 8b of algorithm IV. The Ow and Morton dispatching rule is presented in Appendix B.

Algorithm IV (Tabu search for early/tardy problem).

Algorithm IV is the same as algorithm II except for steps 0, and step 8b which are modified as follows

Step 0. Set $\sigma = J$, $\pi = \phi$. *ETIME* = 0, *TIME* = $\max_{i \in \sigma}\{D_i\}$.

Set the *tabu_size*, n_{max}, and l_{max}.

Set $C_i = D_i$ for all $i \in J$.

Set the initial sequence by the Ow and Mortan algorithm (see Appendix B).

Step 8b Schedule all jobs on R_{ij} as early as possible (i.e., set the start time of the first job in R_{ij} at time zero, and the starting time of the next job in R_{ij} is at the completion time of the immediate preceding job in R_{ij} and so on).

Algorithm IV also provides the same solution as algorithm III for the numerical example 3.2. From the structure of the algorithms, algorithm IV has a drawback in that the two major parameters, *k* (*Ow & Morton alg.*) and *tabu_size*, will need to be specified by scheduler. There are no specific formulas for specifying their initial values. Guidelines do, however, exist.

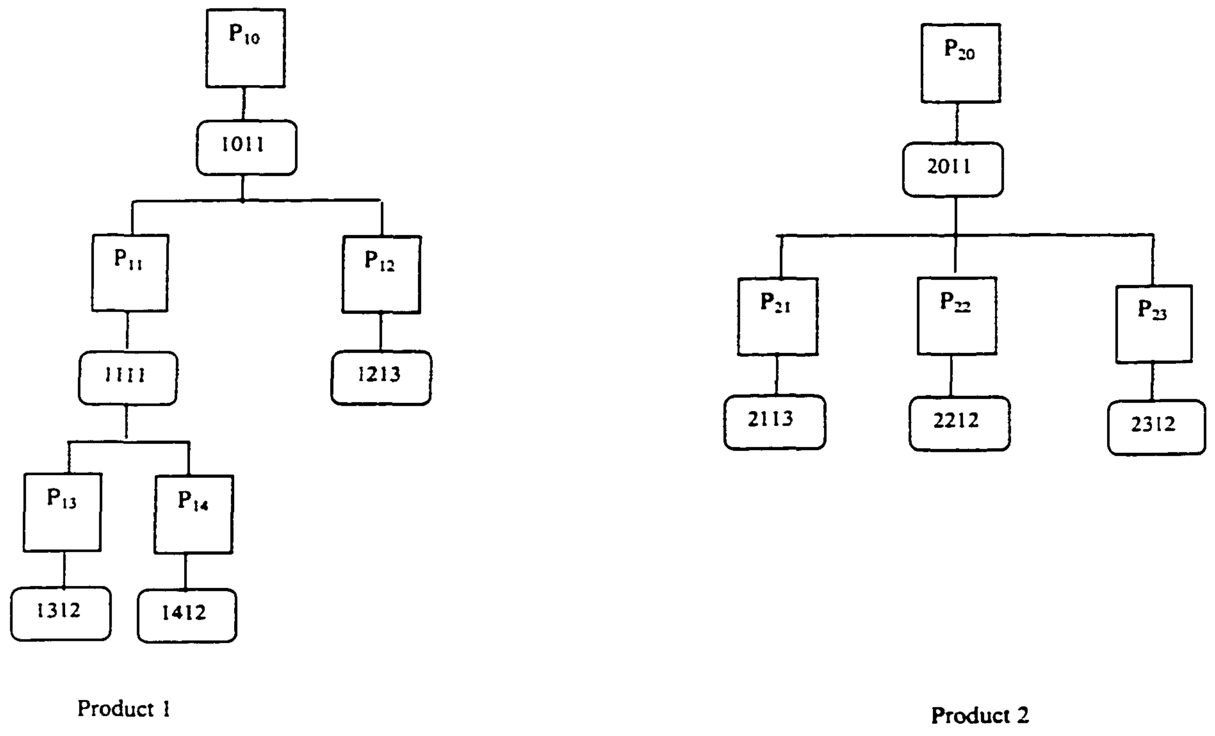
CHAPTER 4

MODEL DEVELOPMENT IN ASSEMBLY JOB SHOP PROBLEM

In this chapter, the solution methodologies for solving the problem of minimizing weighted earliness penalty, and the sum of weighted earliness and weighted tardiness penalties in assembly job shop problem along with their mathematical models are presented.

4.1 Assembly job shop problem with earliness cost minimization

The basic assumption for this problem is that job tardiness is not allowed. Only the earliness cost is considered. The problem is to schedule a set of products with due-dates to minimize the total weighted earliness cost. Each job (product) has a product structure consisting of components and subassemblies that require both machining and assembly operations. Each operation requires a specific machine from a set of *M* machines in the shop. Operations may have different processing times. Each product has its own weighted earliness penalty and sub-jobs of products may have different inventory holding cost. Inventory holding cost for each sub-job may also be different in each stage of its operation. In other word, earliness cost is incurred when the final operation of a product is completed before the product's due-date. Inventory holding cost is incurred, when a sub-job after an operation has to wait for the next operation. In this research, earliness penalty for a product is calculated based on the completion time of the final operation of the product while the inventory holding cost of a sub-job is equivalent to the earliness penalty of an operation, which is not the final operation. The problem was first modeled mathematically. An illustrative product structures is as shown in Figure 4.1. The mathematical formulation is as given in section 4.1.1.



ijkl	Operation required by subjob P_{ij}, where k represents operation and l is the machine used for operation k.
P_{ij}	Subjob j of final job i. A subjob is a component or a subassembly in a product tree or bill of materials.

Figure 4.1. Product structure for assembly products

4.1.1 Problem formulation.

The mathematical model of the assembly job shop problem with the objective of minimizing the weighted earliness penalty is as given below.

Objective function

$$Min \sum_{i=1}^{N}\left(E_{(i0k^{*}m)} * \left(D_i - C_{(i0k^{*}m)}\right) + \sum_{j=0}^{J_i}\sum_{k=1}^{K_{ij}}\left(E_{(ijkm)} * \left(S_{(ij(k+1)m')} - C_{(ijkm)}\right)\right)\right) \qquad (1)$$

Subject to

$C_{(ijkm)} - S_{(ijkm)} = t_{(ijkm)}$ *for* $\forall i, j, k$ *(Processing time constraints)* *(2)*

$C_{(i0k^{*}m)} \leq D_i$ *for* $\forall i, j, k$ *(No tardy job constraints)* *(3)*

$C_{(i'j'k'm)} - C_{(ijkm)} + \alpha(1 - X_{(ijkmi'j'k'm)}) \geq t_{(i'j'k'm)}$ *for* $\forall i, j, k, i', j', k'$ *(Disjunctive constraints)* *(4)*

$C_{(ijkm)} - C_{(i'j'k'm)} + \alpha X_{(ijkmi'j'k'm)} \geq t_{(ijkm)}$ *for* $\forall i, j, k, i', j', k'$ *(Disjunctive constraints)* *(5)*

$S_{(ij(k+1)m')} - C_{(ijkm)} \geq 0$ *for* $\forall i, j, k$ *(Precedence Constraints)* *(6)*

$S_{(ijkm)} \geq 0$ *for* $\forall i, j, k$ *(7)*

$X_{(ijkmi'j'k'm)} \in \{0, 1\}$, *integer,* *for* $\forall i, j, k, i', j', k'$ *(8)*

where N *= number of products,*

D_i *= due-date of product i,*

J_i *= number of sub-jobs in product i.*

K_{ij} *= number of operations in sub-job (ij),*

(ij(k+1)m') = the parent-operation of operation (ijkm),

$S_{(ijkm)}$ *= starting time of operation (ijkm),*

$C_{(iok^{*}m)}$ *= completion time of the final operation of product i,*

$C_{(ijkm)}$ *= completion time of operation (ijkm),*

$t_{(ijkm)}$ *= processing time of operation (ijkm),*

α *= large positive number,*

k^* = The last operation of an end product,

$E_{(i0k^*m)}$ = earliness cost of the final operation of product i (cost/unit time).

$E_{(ijkm)}$ = earliness cost of operation (ijkm) (cost/unit time).

$$X_{(ijkmi'j'k'm)} = \begin{cases} 1, & \text{if operation } (ijkm) \text{ precedes operation } (i'j'k'm) \text{ in machine } m, \\ 0, & \text{otherwise.} \end{cases}$$

Constraints *(2)* express that processing time of an operation is equal to the difference between its starting time and its completion time. Constraints *(3)* guarantee that there is no tardy job in the system. Constraints *(4)* and *(5)* ensure that no two operations can be processed simultaneously on the same machine. Constraints *(6)* are precedence constraints based on the product structure. Constraints *(7)* express that the starting time of operation *(ijkm)* must be positive. Integrality requirement on $X_{(ijkmij'k'm)}$ is described in constraint *(8)*.

Since this problem is NP problem [27], then the presented mathematical model is impractical for solving reasonable size problem. Therefore, a heuristic algorithm is developed for solving the problem. A heuristic, called algorithm V, is the extension of algorithm I, the algorithm based on the local optimality condition for solving single machine problem with earliness cost minimization. Algorithm V is described in section 4.1.2.

4.1.2 Heuristic for assembly job shop problem with the weighted earliness cost minimization

In solving the assembly job shop problem with weighted earliness cost minimization, the computational time required for obtaining optimal solution from mathematical programming or exact method procedures for practical size problems is excessive since the problem is NP [27]. Therefore, mathematical programming approach is not a practical way for solving the problem. The development of efficient heuristic that generates optimal or near-optimal solution is practical.

In this section, a heuristic algorithm is developed. The algorithm, called algorithm V, is the extension of algorithm I. Algorithm V starts with the construction of an ideal solution. An ideal solution is a solution in which the completion times of the final operations of all products are scheduled to occur or coincide with products' due-dates. The completion times of all preceding operations are made to coincide with the starting time of their parent-operations. As in Figure 4.1, assume that the due-date of product *1* is at the *500* th unit time and the processing time of operation *1011* is *100* unit times. The starting time and completion time of operation *1011* in the ideal solution are at the *400*th and the *500*th unit times, respectively. The completion times of the operations *1111* and *1213* are at the *400*th unit times, which is the starting time of their parent-operation, operation *1011*. Similarly, the completion times of operations *1312* and *1412* are at the starting time of operation *1111* in the ideal solution.

If the generated ideal solution is feasible, then the solution is optimal. However, in the real world, machine conflicts between operations are always present in ideal solution. Any solution with machine conflicts between operations is infeasible. To obtain a feasible solution, algorithm V solves the multiple machine problem as a series of single machine problems. Algorithm V solves the problem backward. It starts with the unscheduled operation with the latest completion time in the ideal solution, called the latest operation. If the operation has no machine conflict with other operations, this operation is scheduled on the machine as it was in the ideal solution. Otherwise algorithm V will identify a set of ready operations that have conflict with the latest operation. A ready operation is an unscheduled operation, which is either the final operation of product or an operation whose successor operation has already been scheduled. Machine conflicts are resolved based on the local optimality conditions presented in section 3.1.2. After the conflicts are eliminated, the previously conflicting operation is marked as scheduled operations. Next, the algorithm moves backwards to select the new latest unscheduled operation and the process is repeated until all schedule conflicts in the problem are eliminated. For illustration, let us consider the following example, called example 4.1.

Assume that products *1* and *2* are to be scheduled. The product structures of the products are as shown in Fig. 4.1. Due-dates of both products are at the *500*th unit time and the processing times and weighted earliness penalty of the operations are as shown in Table 4.1.

The ideal solution is constructed as shown in Fig. 4.2. There are three sets of machine conflict operations. The first set is on machine *1* and it involves operations *2011*, *1011*, and *1111*. The second set is on machine 2 and it consists of operations *2312, 2212, 1312, and 1412*. Operations *2113* and *1213* on machine 3 are members of the third set. Therefore, there are three decomposed single machine problems that need to be solved at this point.

Table 4.1 Parameters of product 1 and product 2 in example 4.1

Operations	Earliness cost of final job and inventory holding cost of sub-jobs (E_{ijkl}) (unit cost/ unit time)	Processing time (t_{ijkl}) (unit time)
1011	70	100
1111	40	40
1213	20	50
1312	20	40
1412	20	60
2011	50	140
2113	30	100
2212	30	50
2312	30	120

In algorithm V, the single machine problem containing the operation with the latest completion time is first solved. For this example, machine one has the operations with the latest completion times. The machine conflicts of operations in this set are eliminated by applying the local optimality properties presented in section 3.1.2. At this point, the new sequence of operations on machine *1* is *2011→1111→1011*. Then, the new ideal solution is then formed based on the sequence of operations at machine *1* (see Fig. 4.3). That is, the next ideal solution is constructed by keeping the sequence on machine 1 unchanged after the conflict resolution.

Now the latest machine conflict is on machine *2*. It involves operations *1312* and *1412*. Again, the machine conflict between these two operations is eliminated based on the local optimality properties. The new sequence of operations at machine *2* is *1412→1312* (see Fig. 4.4). The new ideal solution is then formed based on the sequence of operations on machine 2 and machine 1 (see Fig. 4.4). Finally, the latest conflict is, again, on machine *2* involving operations *2212* and *2312*. After eliminating the conflict, the new operation sequence for the last conflict is *2312→2212*. The current feasible solution is as shown in Fig. 4.5.

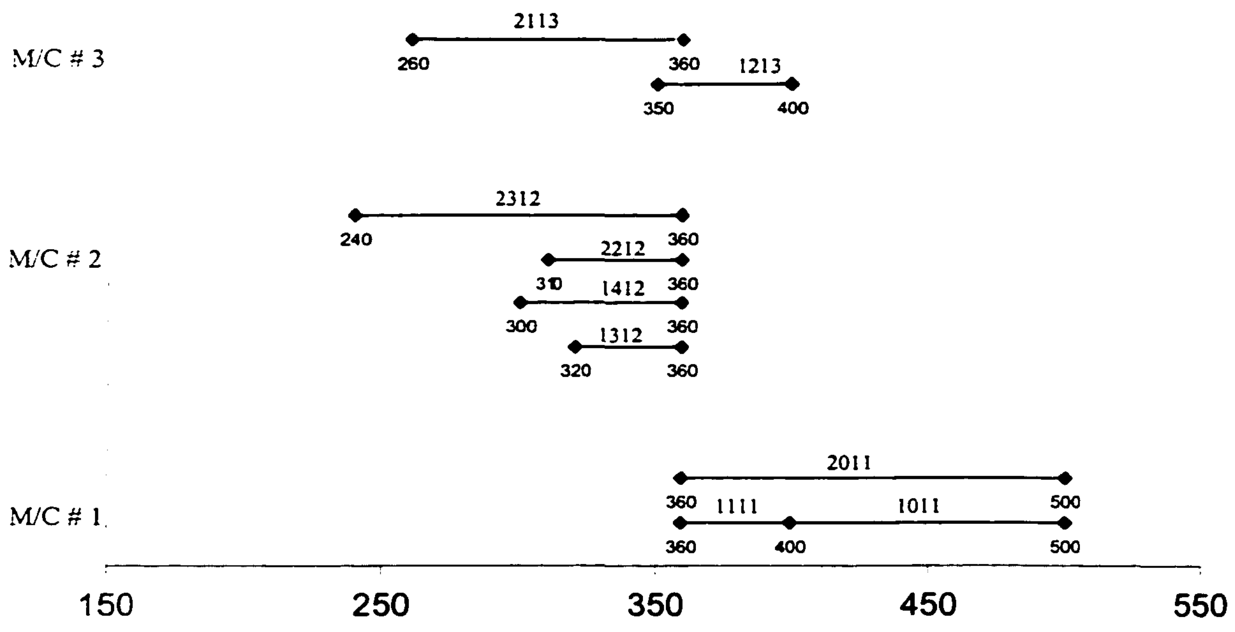


Figure 4.2 An ideal solution of the example 4.1

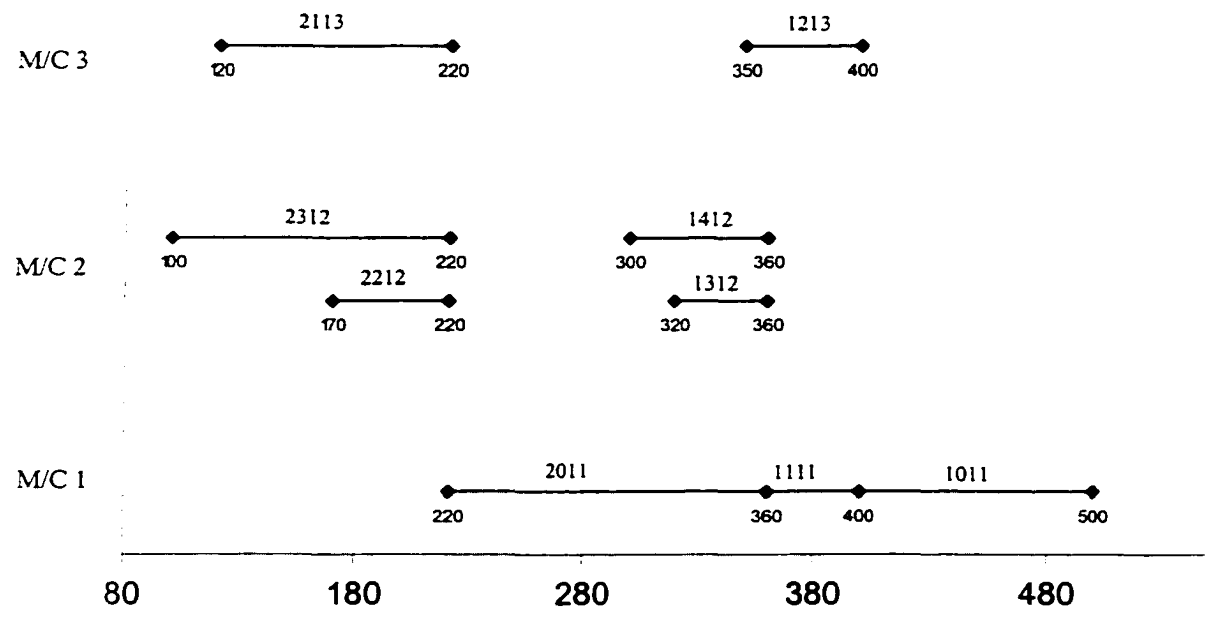


Figure 4.3 An ideal solution of example 4.1 after eliminating the latest conflict at machine 1

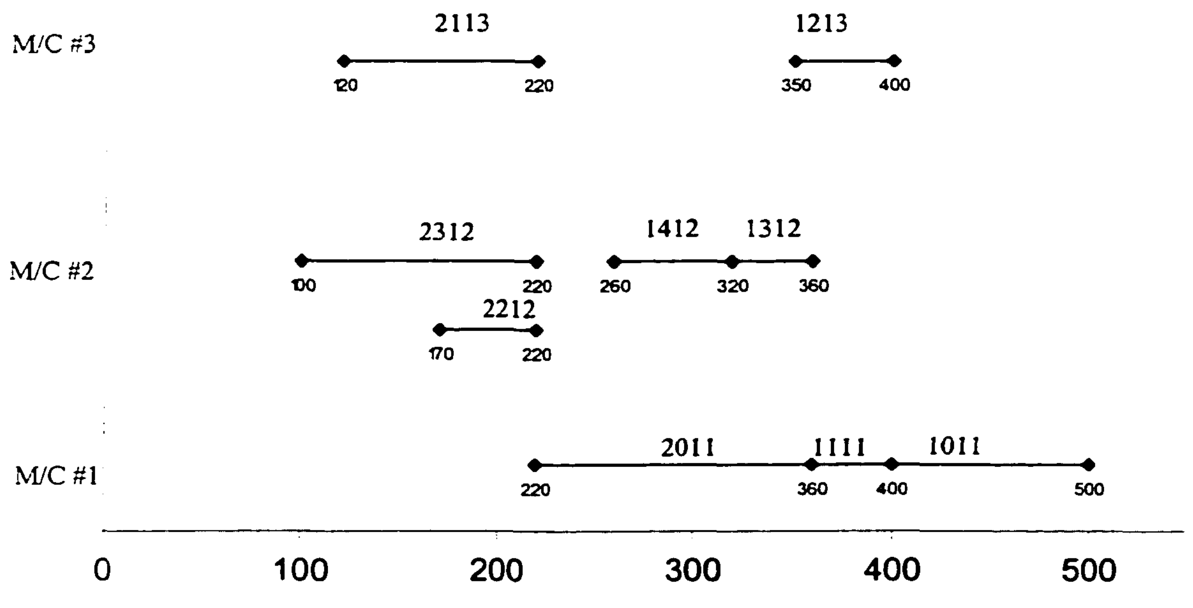


Figure 4.4 An ideal solution of example 4.1 after eliminating the latest conflict at machine 2

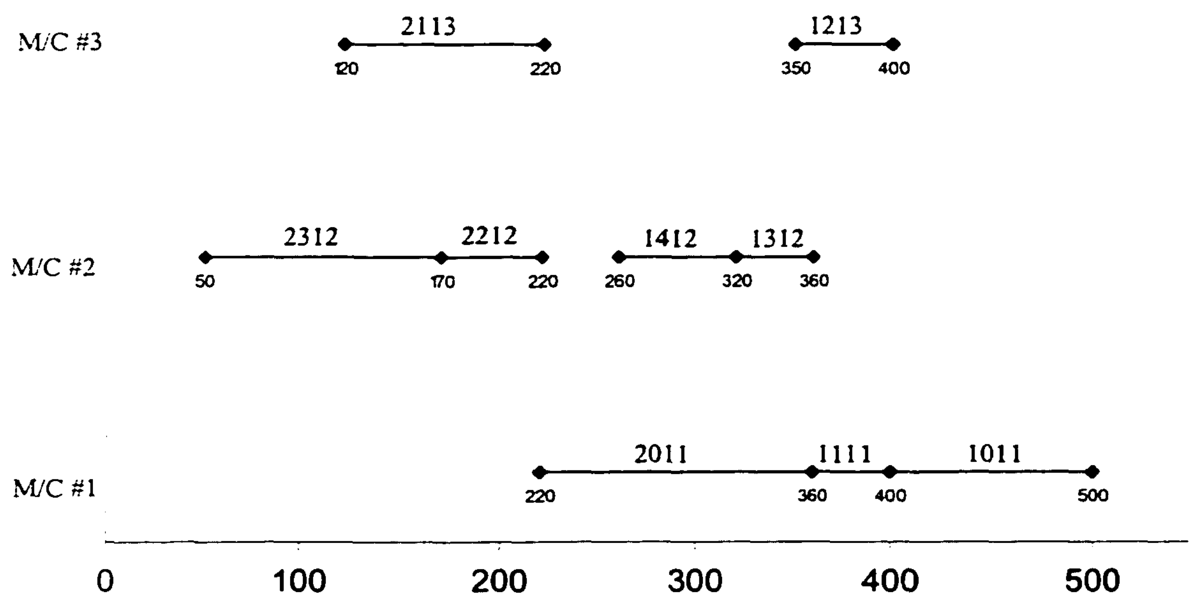


Figure 4.5 A feasible solution of the example 4.1 after eliminating all machine conflicts

At this point, the solution obtained may not be the best solution, since the algorithm solved each decomposed problem independently. The best sequence in a decomposed problem may not be the best sequence for the entire problem. To overcome this drawback, a search for improved solution based on the conflict free schedule of Figure 4.5 is necessary. Algorithm V searches for improved solution by shifting or moving the moveable operations from one sequence position to another on a machine without violating schedule feasibility constraints. A movable operation is an operation that can be moved to a new sequence position from the current schedule without violating any precedence constraint. For example, in the current solution of example 4.1 (i.e. Fig. 4.5), operation *2011* can be moved to two new positions, either starting at time *260* and finishing at time *400*, or starting at time *360* and finishing at time *500*. In other word, the current schedule *"2011→1111→1011"* on machine 1 can be changed to the schedule *"1111→2011→1011"* or *"1111→1011→2011"*. Assume that operation *2011* is selected to move to new position (i.e., the position of operation *1111*) to make new sequence

"1111→2011→1011". After placing operation *2011* into the new position, other operations (i.e. *1111,1213, ..., 2312*) that preceeded operation 2011 in the sequence may no longer be in the best sequence. From this point of view, the schedule may be improved by searching for the best sequence of this set of operations. This can be done by pegging the sequence of the placed operation and the operations previously scheduled after the placed operation (i.e. *2011→1011*), and setting all operations which were scheduled before operation *2011* in an ideal schedule (see Figure 4.6). Then any machine conflict between operations occurring on the schedule is eliminated by using the optimality conditions as previously described. This search procedure is performed for all movable operations in the current schedule. After all movable operations are evaluated, the algorithm selects the lowest cost solution. If the best solution from the search is better than the current best solution, set that solution as the current best solution and repeat the search procedure again. Otherwise the algorithm stops. The entire steps of algorithm V can be described as in the following section.

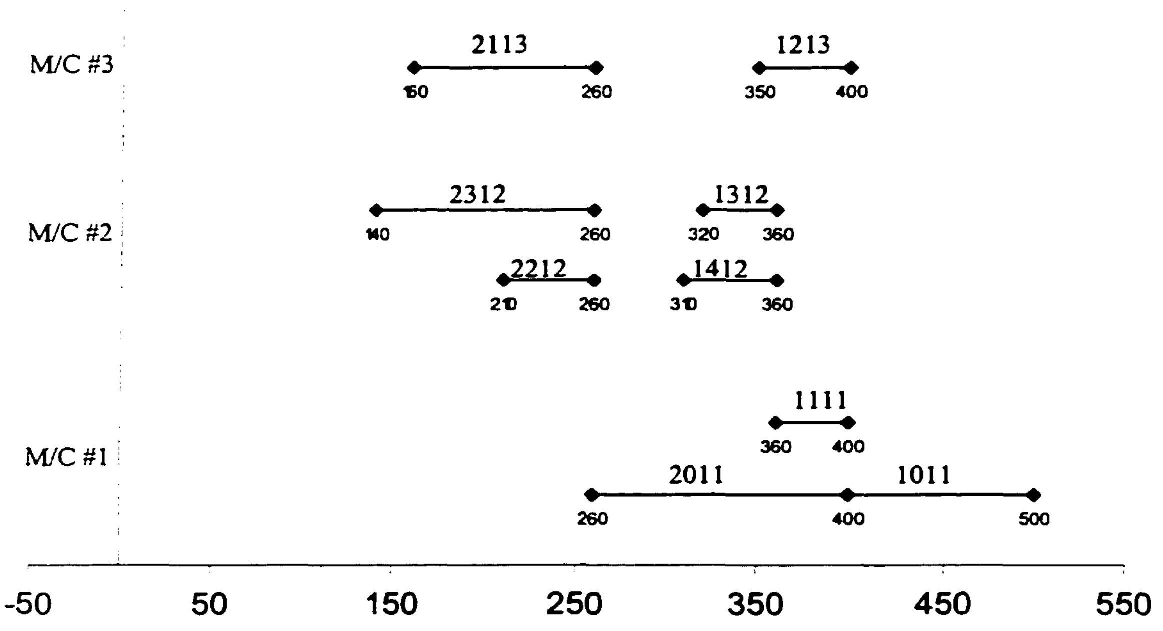


Figure 4.6 An ideal solution of example 4.1 after changing the schedule of operation 2011

4.1.3 Algorithm V development

Algorithm V employs a fundamental procedure, which is the method for eliminating machine conflicts in the single machine problem. This procedure is part of algorithm I presented in section 3.1.3.1. For ease of understanding, the elimination method is briefly described again in this section.

4.1.3.1 Machine conflict elimination method in single machine problem (MCE)

The following variables and parameters are used in the algorithm.

Input parameters

J = Set of operations.

$E_{(ijkl)}$ = Earliness cost of the operation *(ijkl)* (cost/unit time).

$t_{(ijkl)}$ = Processing time of the operation *(ijkl)*.

$Y_{(ijkl)} = \frac{E_{(ijkl)}}{t_{(ijkl)}}$.

D_i = Due-date of the product *i*.

$D_{(ijkl)}$ = Due-time of operation *(ijkl)*.

$O_{(ijkl)}$ = Operation *(ijkl)*.

System variables

$S_{(ijkl)}$ = Starting time of the operation *(ijkl)*.

$C_{(ijkl)}$ = Completion time of the operation *(ijkl)*.

σ = Set of unscheduled operations.

π = Set of scheduled operations.

$\delta_{(ijkl)}$ = Set of operations that conflict with operation *(ijkl)*based on the ideal schedule.

TIME = Latest available time on machine.

For any machine conflict that exists between operations, do the following procedure.

Step 0. Initialization

0a. Set $\sigma = J$, $\pi = \phi$, $TIME = \max_{O(ijkl) \in \sigma} \{D_{(ijkl)}\}$.

0b. For each $O_{(ijkl)} \in \sigma$, set $\delta_{(ijkl)} = \phi$, $Y_{(ijkl)} = \frac{E_{(ijkl)}}{t_{(ijkl)}}$.

Step1. If there are some final operations of products in J, set the due-times of the final operations to their end-products' due-date (*i.e.*, $D_{(i01l)} = D_i$). For each $O_{(ijkl)} \in J$, which is not the final operation of products, set its due-time to the starting time of its parent-operation (i.e., $D_{(ijkl)} = S_{(ij(k-1)l')}$, where $O_{(ij(k-1)l')}$ is the parent-operation of $O_{(ijkl)}$ in the product structure).

Step2. Let P be the set of operations $(ijkl) \in \sigma$ with $D_{(ijkl)} \geq TIME$ (i.e., $P = \{D_{(ijkl)} \geq TIME$, $O_{(ijkl)} \in \sigma\}$). Select $O_{(ijkl)^*}$ where $Y_{(ijkl)^*} = \max_{O(ijkl) \in P} \{Y_{(ijkl)}\}$. If $P = \phi$, then select $O_{(ijkl)^*}$, that satisfy $D_{(ijkl)^*} = \max_{O(ijkl) \in \sigma} \{D_{(ijkl)}\}$. Break ties arbitrarily.

Step 3. Set $C_{(ijkl)^*} = \min \{TIME, D_{(ijkl)^*}\}$, $S_{(ijkl)^*} = C_{(ijkl)^*} - t_{(ijkl)^*}$.

For each $O_{(ijkl)} \in \sigma - \{O_{(ijkl)^*}\}$, if that conflicts with $O_{(ijkl)^*}$, set $\delta_{(ijkl)^*} \leftarrow \delta_{(ijkl)^*} + \{O_{(ijkl)}\}$. If there is no any job conflicts with $O_{(ijkl)^*}$, go to step 7.

Step 4. If $Y_{(ijkl)} \leq Y_{(ijkl)^*}$, for all $O_{(ijkl)} \in \delta_{(ijkl)^*}$, go to step 7, otherwise go to step 5

In this step, it follows proposition 1, if $D_{(ijkl)^*} \geq D_{(ijkl)}$ for all $O_{(ijkl)} \in \delta_{(ijkl)^*}$. It follows the proposition 3, if $O_{(ijkl)^*}$ and some of $O_{(ijkl)}$ conflict to a scheduled operation. It follows the proposition 4, if only $O_{(ijkl)^*}$ conflicts to a scheduled operation and some of $O_{(ijkl)}$ conflict to $O_{(ijkl)^*}$. Therefore, if $Y_{(ijkl)} \leq Y_{(ijkl)^*}$, for all $O_{(ijkl)} \in \delta_{(ijkl)^*}$, is hold, $O_{(ijkl)^*}$ is the best operation to be scheduled in any environment. The propositions 1, 3, 4 are presented in section 3.1.2

Step 5. Select $O_{(ijkl)'} \in \delta_{(ijkl)^*}$ where $D_{(ijkl)'}$ is the latest due-time among operations with $Y_{(ijkl)'} \geq Y_{(ijkl)^*}$ in $\delta_{(ijkl)^*}$. Break ties arbitrarily.

Step 6. Find the relationship between $O_{(ijkl)^*}$ and $O_{(ijkl)'}$ based on local optimality conditions.

6a. Set $T = \max \{ TIME - D_{(ijkl)^*}, 0\}$.

6b. For $Y_{(ijkl)^*} \leq Y_{(ijkl)'}$, $D_{(ijkl)^*} > D_{(ijkl)'}$, and $T \geq 0$,

if $(D_{(ijkl)'} - D_{(ijkl)^*} + T) \leq \dfrac{t_{(ijkl)'}.E_{(ijkl)^*} - t_{(ijkl)^*}.E_{(ijkl)'}}{(E_{(ijkl)^*} + E_{(ijkl)'})}$ (follow propositons 2 for $T = 0$, and 5 for $T > 0$), set $\delta_{(ijkl)^*} \leftarrow \delta_{(ijkl)^*} - \{O_{(ijkl)}\}$ and go to step 4. Otherwise set $C_{(ijkl)^*} = D_{(ijkl)^*}$, $S_{(ijkl)^*} = C_{(ijkl)^*} - t_{(ijkl)^*}$ (i.e., set $O_{(ijkl)^*}$ back to ideal form), set $O_{(ijkl)^*} = O_{(ijkl)'}$,$\delta_{(ijkl)^*} = \phi$ (i.e., $O_{(ijkl)'}$ is the selected operation) and go to step 3.

Step 7. Schedule $O_{(ijkl)^*}$ on machine

7a. Set *TIME* as the starting time of the $O_{(ijkl)^*}$, which is the just scheduled operation (i.e., set $TIME = S_{(ijkl)^*}$).

7b. Delete $O_{(ijkl)^*}$ from the set of unscheduled operations and add $O_{(ijkl)^*}$ to the set of scheduled operations (i.e. $\sigma \leftarrow \sigma - \{O_{(ijkl)^*}\}$ and $\pi \leftarrow \pi + \{O_{(ijkl)^*}\}$).

7c. If all operations are scheduled ($|\sigma| = 0$), then stop, otherwise go to step 2.

A step by step demonstration of the *MCE* algorithm is illustrated in Appendix C.

4.1.3.2 Summary of algorithm V

In this section, a step by step description of algorithm V is presented. This algorithm includes the *MCE* method, which was previously presented in section 4.1.3.1 and a search method, which was briefly described in section 4.1.3. The followings are additional notations and definitions used in the presentation of algorithm V.

System variables

λ : Set of ready operations.

$\Omega_{(ijkl)}$: Set of ready operations, which conflict with $O_{(ijkl)}$.

M : Set of feasible solutions after applying the search method.

Definitions

Ready Operation: An unscheduled operation, which is either the final operation of a product or the operation whose parent-operation was already scheduled. Backward scheduling is used.

Movable operation: An operation that can be moved to a new sequence position from the current schedule without violating any precedence constraint.

Algorithm V

Step 0. Set $M = \phi$. For each operation, set it as an unscheduled operation.

Step 1. Construct an ideal solution.

1a. For each unscheduled operation, if it is the final operation of a product, set the completion time of the operation at the product's due-date *(i.e.,* $C_{(i01l)} = D_i$*)*, otherwise set the completion time of the operation as the starting time of its parent-operation (i.e., $C_{(ijkl)} = S_{(ij'(k-1)l')}$, where $O_{(ij'(k-1)l')}$ is the parent-operation of $O_{(ijkl)}$ in the product structure), $S_{(ijkl)} = C_{(ijkl)} - t_{(ijkl)}$, where the schedule is built in a backward manner.

Step 2. Determine machine conflict operation

2a. Let λ be the set of unscheduled operations, which are ready operations.

2b. For all $O_{(ijkl)} \in \lambda$, select the operation, $O_{(ijkl)^*}$, with the latest due-time (*i.e.,* $D_{(ijkl)^*} = Max\{D_{(ijkl)}\}, \ \forall O_{(ijkl)} \in \lambda$). Break ties arbitrarily.

2c. If the $O_{(ijkl)^*}$ has no conflict with both unscheduled and scheduled operations in ideal solution (note that $O_{(ijkl)^*}$ can conflict with some unscheduled operations in an ideal solution), schedule $O_{(ijkl)^*}$ on the machine as it was in the ideal solution, set $O_{(ijkl)^*}$ as a scheduled operation and go to step 4.

If $O_{(ijkl)^*}$ conflicts with scheduled operations but has no any conflict with unscheduled operations, schedule $O_{(ijkl)^*}$ on the machine at the starting time of the earliest scheduled operation (i.e., set $C_{(ijkl)^*}$ at $S_{(ijkl)^{**}}$ where $S_{(ijkl)^{**}} = Min\{S_{(ijkl)}\}, \forall O_{(ijkl)} \in \pi$ on machine l), and set $O_{(ijkl)^*}$ as a scheduled operation and go to step 4.

If $O_{(ijkl)^*}$ conflicts with unscheduled operations (no matter that $O_{(ijkl)^*}$ conflicts to scheduled operation or not), place all unscheduled $O_{(ijkl)} \in \lambda$, which conflict with $O_{(ijkl)^*}$ in $\Omega_{(ijkl)^*}$, also add operation $O_{(ijkl)^*}$ into $\Omega_{(ijkl)^*}$ (i.e., set $\Omega_{(ijkl)^*} = \Omega_{(ijkl)^*} + O_{(ijkl)^*}$) and go to step 3. In this step, all unscheduled operations which conflict to $O_{(ijkl)^*}$ and $O_{(ijkl)^*}$ itself must be carried to step 3 to eliminate machine conflicts by employing *MCE* method.

Step 3. Apply *MCE* method to eliminate machine conflicts of all operations in $\Omega_{(ijkl)^*}$, Select the latest due-date operation of the sequence obtained from *MCE* method to schedule and set it as a scheduled operation. Set the remaining operations in $\Omega_{(ijkl)^*}$ as unscheduled operations and set $\Omega_{(ijkl)^*} = \varnothing$, and go to step 4.

Step 4. If all operations in the problem are scheduled, then go to step 5, otherwise set all unscheduled operations in an ideal solution while keeping unchanged the schedule of scheduled operations and go back to step 2.

Step 5. Calculate total earliness cost of the solution obtained in Step 4.

Step 6. Set the current solution obtained from step 4 to be the current best solution, , called *R*. Also set $R'_1 \leftarrow R$.

Step 7. Search for improved solution by evaluating the changing of the sequence position of movable operations in the current schedule. For each operation, $O_{(ijkl)'}$,in *R*, repeat step 7a.

7a. Determine the operations which are scheduled before $O_{(ijkl)'}$ in *R* (*i.e.*, all operations whose $C_{(ijkl)} \leq S_{(ijkl)'}$) and can be moved to the position of $O_{(ijkl)'}$ (i.e., $D_{(ijkl)} \geq S_{(ijkl)'} + t_{(ijkl)}$). Let $P_{(ijkl)'}$ be the set of operations that can be moved to take the position of $O_{(ijkl)'}$ in *R*. For each operation $O_{(ijkl)^*} \in P_{(ijkl)'}$, repeat the following steps (7b-7g).

7b. Remove operation $O_{(ijkl)'}$ from *R* and set $O_{(ijkl)'}$ to be an unscheduled operation.

7c. Let *LTIME* be the latest available time on machine after removing $O_{(ijkl)'}$. (i.e. $LTIME = S_{(i'j'k'l)}$ where $O_{(i'j'k'l)}$ is the operation scheduled right after $O_{(ijkl)'}$ in R).

7d. Schedule $O_{(ijkl)''} \in P_{(ijkl)'}$ into the previous position of $O_{(ijkl)'}$ in R. If the due-time of $O_{(ijkl)''}$ is greater than *LTIME* (*i.e.*, $D_{(ijkl)''} \geq LTIME$), let $C_{(ijkl)''} = LTIME$, otherwise set $C_{(ijkl)''} = D_{(ijkl)}$. Set $S_{(ijkl)''} = C_{(ijkl)''} - t_{(ijkl)''}$ and let $O_{(ijkl)''}$ be a scheduled operation.

7e. Set all operations, $O_{(ijkl)}$, except $O_{(ijkl)''}$ which are previously scheduled before $O_{(ijkl)'}$ in R (i.e., $C_{(ijkl)} \leq S_{(ijkl)'}$) and operation $O_{(ijkl)'}$ as unscheduled operations.

7f. Repeat step1 – step5 to schedule the unscheduled operations and keep the solution in M.

7g. Reset $R \leftarrow R'_1$.

Step 8. Select the best solution, R' in M, resulting from the search method in step 7.

Step 9. If the total cost of R' is less than the total cost of R, set R' as the current best solution (i.e., $R = R'$), set $M = \phi$ and return to step 7, otherwise go to step 10.

Step 10. Output the best schedule determined.

A step by step illustration of algorithm V with example 4.1 is given in Appendix C.

4.2 Assembly job shop problem with the sum of weighted earliness and tardiness cost minimization

This problem is not significantly different from the problem of minimizing the weighted earliness penalty, which was presented in section 4.1. The only difference is that job tardiness is allowed in this problem. Thus, the tardiness cost of products must be included in the model. The problem is to schedule a set of products to minimize the sum of weighted earliness and tardiness penalties. Including the tardiness cost into the model increases the complexity of the problem. The problem is NP problem [6]. The optimal solutions can be obtained using exact procedures for only small size problems. Therefore, heuristic algorithm is a practical approach to solving reasonable size problems. In this research, both mathematical and heuristic approaches are presented.

The problem is first modeled mathematically in section 4.2.1 and the heuristic algorithm for finding solution to the problem is described in section 4.2.2

4.2.1 Problem formulation

The mathematical model of the assembly job shop problem with the objective of minimizing the sum of weighted earliness and tardiness penalties is as given below.

Objective function

$$Min \sum_{i=1}^{N} \left(\begin{array}{l} (E_{(i0k^* m)} * max(0, D_i - C_{(i0k^* m)})) + (W_i * max(0, C_{(i0k^* m)} - D_i)) \\ + \sum_{j=0}^{J_i} \sum_{k=1}^{K_{ij}} \left(E_{(ijkm)} * \left(S_{(ij'(k+1)m')} - C_{(ijkm)}\right)\right) \end{array} \right)$$

This objective function can be transformed to the following function

$$Min \sum_{i=1}^{N} \left((E_{(i0k^* m)} * A_i) + (W_i * B_i) + \sum_{j=0}^{J_i} \sum_{k=1}^{K_{ij}} \left(E_{(ijkm)} * \left(S_{(ij'(k+1)m')} - C_{(ijkm)}\right)\right) \right) \qquad (1)$$

Subject to

$C_{(ijkm)} - S_{(ijkm)} = t_{(ijkm)}$ *for* $\forall i, j, k$ *(Processing time constraints)* *(2)*

$C_{(i'j'k'm)} - C_{(ijkm)} + \alpha(1 - X_{(ijkm,i'j'k'm)}) \geq t_{(i'j'k'm)}$ *for* $\forall i, j, k, i', j', k'$ *(Disjunctive constraints)* *(3)*

$C_{(ijkm)} - C_{(i'j'k'm)} + \alpha X_{(ijkmi'j'k'm)} \geq t_{(ijkm)}$ *for* $\forall i, j, k, i', j', k'$ *(Disjunctive constraints)* *(4)*

$S_{(ij'(k+1)m')} - C_{(ijkm)} \geq 0$ *for* $\forall i, j, k$ *(Precedence Constraints)* *(5)*

$S_{(ijkm)} \geq 0$ *for* $\forall i, j, k$ *(6)*

$X_{(ijkmi'j'k'm)} \in \{0, 1\}$, *integer,* *for* $\forall i, j, k, i', j', k'$ *(7)*

$A_i \geq D_i - C_{(i0k^* m)}$, *for* $\forall i$ *(8)*

$B_i \geq C_{(i0k^* m)} - D_i$, *for* $\forall I$ *(9)*

$A_i \geq 0$ *for* $\forall i$ *(10)*

$B_i \geq 0$ *for* $\forall i$ *(11)*

where N = number of products,

D_i = due-date of product i,

J_i = number of sub-jobs in product i.

K_{ij} = number of operations in sub-job (ij),

$(ij'(k+1)m')$ = the parent-operation of operation (ijkm),

$S_{(ijkm)}$ = starting time of operation (ijkm),

$C_{(i0k^{*}m)}$ = completion time of the final operation of product i,

$C_{(ijkm)}$ = completion time of operation (ijkm),

A_i = the amount of time of the final operation of product i completed before products's due-date (i.e. $A_i = D_i - C_{(i10m)}$).

B_i = the amount of time of the final operation of product i completed after product's due-date (i.e. $B_i = C_{(i10m)} - D_i$).

$t_{(ijkm)}$ = processing time of operation (ijkm),

α = large positive number,

$E_{(i0k^{*}m)}$ = earliness cost of the final operation of product i (cost/unit time).

$E_{(ijkm)}$ = earliness cost of operation (ijkm) (cost/unit time).

$$X_{(ijkmi'j'k'm)} = \begin{cases} 1, & \text{if operation}\,(ijkm)\,\text{precedes operation}\,(i'j'k'm)\,\text{in machine}\ m, \\ 0, & \text{otherwise.} \end{cases}$$

Constraints *(2)* express that processing time of an operation is equal to the difference between its starting time and its completion time. Constraints *(3)* and *(4)* ensure that no two operations can be processed simultaneously on the same machine. Constraints *(5)* are precedence constraints based on the product structure. Constraints *(6)* express that starting time of operation *(ijkl)* must be positive. Integrality requirement on $X_{(ijkli'j'k'l)}$ is described in constraint *(7)*. Constraints *(8)*, *(9)* *(10)* and *(11)* linearize the nonlinear objective function of the problem.

4.2.2 Heuristic for assembly job shop problem with the sum of weighted earliness and tardiness cost minimization

In an assembly product, tardiness cost for an assembly product is incurred only when the end product is completed beyond its due-date. From this characteristic, we can consider the tardiness cost of the product on the final operation of the product. If the final operation of the product is completed beyond the product's due-date, the tardiness cost is incurred. The tardiness cost is not considered for the sub-jobs because they are not the final job. For example, in example 4.1, the tardiness costs are considered for the final operations of sub-jobs P_{10}, and P_{20}, with final operations *1011* and *2011,* respectively. If operation *1011* is completed after the due-date of product 1, then tardiness cost is incurred. It is the same for operation *2011* of product 2. For the other operations, inventory cost is the only penalty that can be incurred. Inventory cost is incurred when an operation is completed and it has to wait for the next operation to begin. In this research, inventory holding cost is considered as earliness cost of operations, which are not the final operations.

Based on the above characteristics of the problem, a possible solution strategy to take is to find a procedure that can identify the appropriate amount of tardiness for the last operation of each product and integrate the procedure with heuristic algorithm V, which was presented in section 4.1. In other word, the algorithm for minimizing earliness cost, algorithm V, should be employed to schedule the operations, which are not the final operations, after the appropriate amount of tardiness of each product is set or determined.

The integrated algorithm, called algorithm VI developed for solving the problem is the extension of algorithm V. In algorithm VI, algorithm V is first applied to solve the problem. Solutions obtained from Algorithm V may be either infeasible or feasible. Infeasibility in this case implies that some operations have to start before time zero to produce at products within their due dates. In the latter case, feasibility implies all products can be produced to meet their due dates. However, a schedule that meets the due dates of all jobs does not imply a schedule with minimum cost. It is possible to lower production cost by forcing some jobs to be tardy. Therefore, algorithm VI should be able to solve both problem cases. Since these two cases can be studied and solved independently, ones can separately study and develop procedure for solving each problem case. Later, these two developed methods are integrated with algorithm V to form algorithm VI, which is an algorithm for minimizing the sum of weighted earliness and tardiness penalties. These two problem cases are discussed in sections 4.2.2.1 and 4.2.2.2.

4.2.2.1 The case involving the realization of an infeasible solution after employing algorithm V

For this problem case, there are two possible ways to get feasible solution. In the first approach, the infeasible schedule is right shifted in time without violating any constraint until feasible solution is obtained. In the second approach, set new due-date, called virtual due-date, for the final operation of a product. For example, consider, the product structure shown in Figure 4.7. The processing times, the earliness costs, and the tardiness costs for each operation are presented in Table 4.2. The actual due-dates for product 1 is *60*, product 2 is *50*, and product 3 is *40*.

Table 4.2 Parameters of products 1, 2 and 3 in example 4.2

Operation	Processing time (unit time)	Earliness Cost (cost/time)	Tardiness Cost (cost/time)	Operation	Processing time (unit time)	Earliness Cost (cost/time)	Tardiness Cost (cost/time)
1011	11	20	25	2222	5	6	-
1111	7	12	-	2213	3	2	-
1312	5	4	-	3031	10	20	40
1412	5	4	-	3022	8	12	-
1213	3	4	-	3011	6	10	-
2021	10	22	35	3112	5	7	-
2012	7	11	-	3313	3	2	-
2133	4	7	-	3413	2	1	-
2122	4	5	-	3221	6	6	-
2111	2	2	-	3212	2	1	-

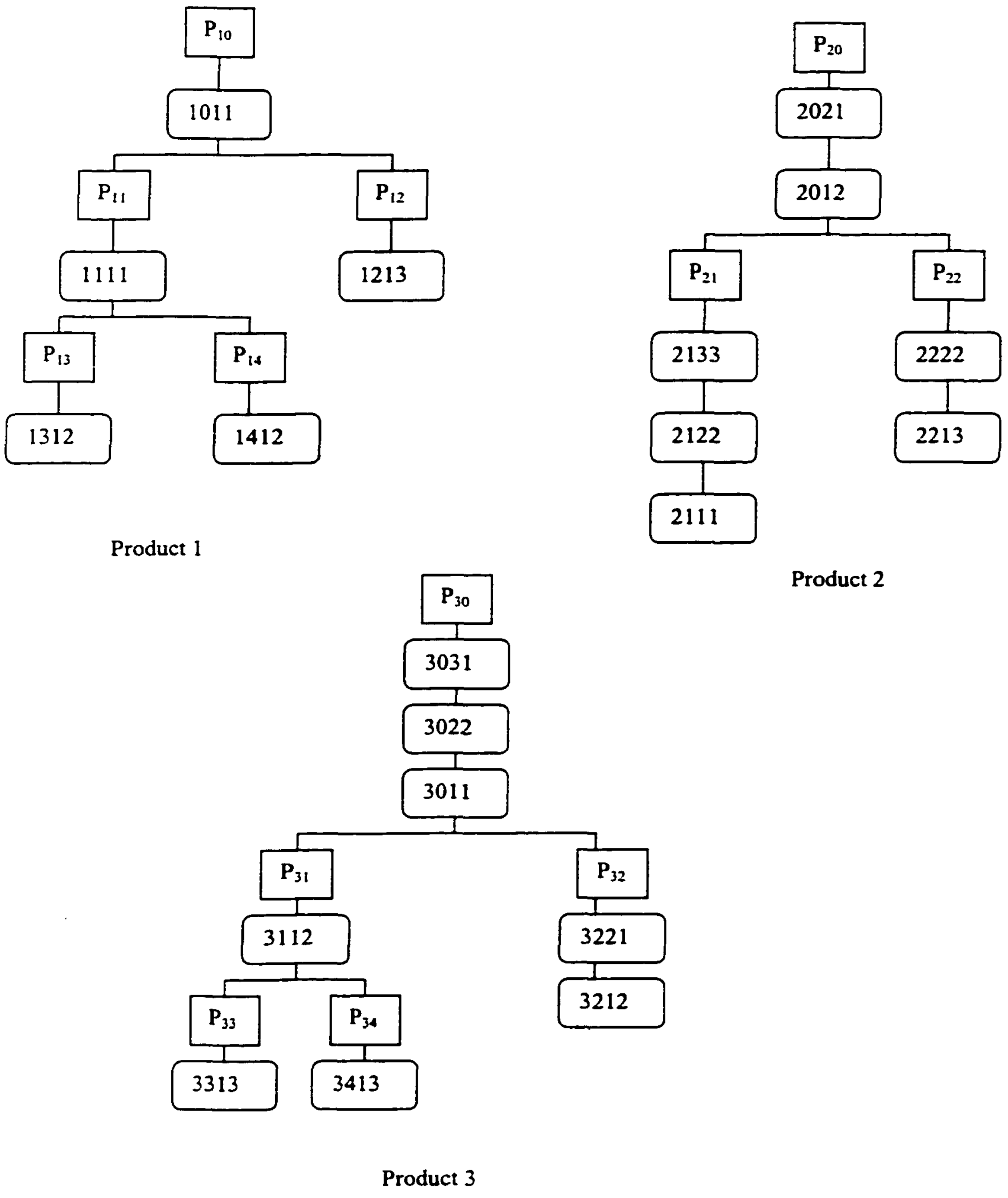


Figure 4.7. Product structure for assembly products in examples 4.2-4.3

For simplicity, we will refer to the due-dates as *"60-50-40"*. The ideal schedule and the schedule after employing algorithm V are as shown in Figure 4.8-4.9, respectively. The solution given in Figure 4.9 is infeasible, since there are some operations that start in negative time. The negative start time also implies that at least one product must be tardy to obtain feasible solution. Now, we can set the virtual due-date of each product by setting it as the actual due-date plus the absolute value of the earliest non-positive start time of any operation in the infeasible solution. The earliest non-positive start time is *–8* for operation *3413* in Fig. 4.9. By this concept, three sets of due-dates can be obtained, which are *"68-50-40"*, *"60-58-40"*, and *"60-50-48"*. For each due-date set, algorithm V is applied to solve the problem. The due-date set, which provides the best feasible solution is selected. If a feasible solution cannot be obtained, the method is repeated until feasible solution is achieved. These two approaches contain some weaknesses. The rightward shift method depends pretty much on the initial solution (infeasible solution) which was obtained from algorithm V, but more than one product can be tardy simultaneously. This will not happen in the virtual due-date method. The virtual due-date method searches for one tardy product at a time. It is quite possible that a local optimal solution is obtained in an early stage of the process. The integration of these two approaches can help to reduce the weaknesses of both. This can be done by applying the rightward shift method after the virtual due-date method, if the virtual due-date still provides an infeasible solution. For example, with the virtual due-date of *"60-50-48"*, the ideal solution is as shown in Figure 4.10 and the schedule after employing algorithm V is the same as the schedule in Figure 4.9. Although this solution is infeasible, it will not be discarded. The rightward shift method is applied to get a feasible solution (see Fig.4.11). This solution is then compared with solutions obtained from the cases involving the virtual due-date sets *"68-50-40"* and *"60-58-40"*. This integration method helps to search for solution more broadly. This reduces the problem of overlooking potential solutions if the virtual due-date method is used alone.

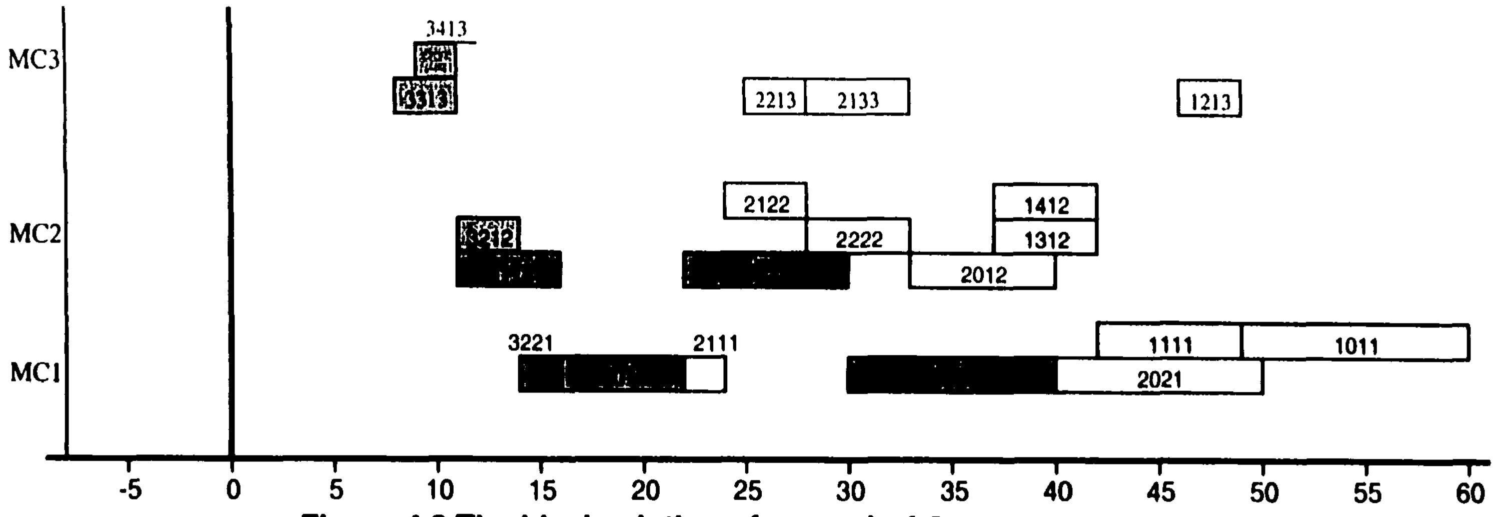


Figure 4.8 The ideal solution of example 4.2

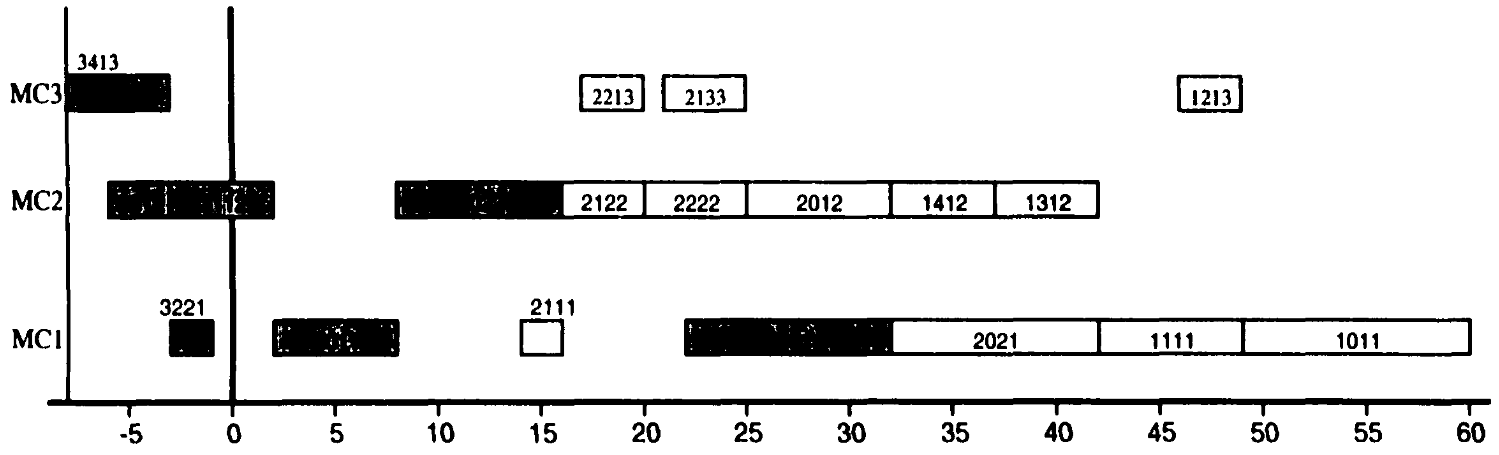


Figure 4.9 The infeasible solution after employing algorithm V in example 4.2

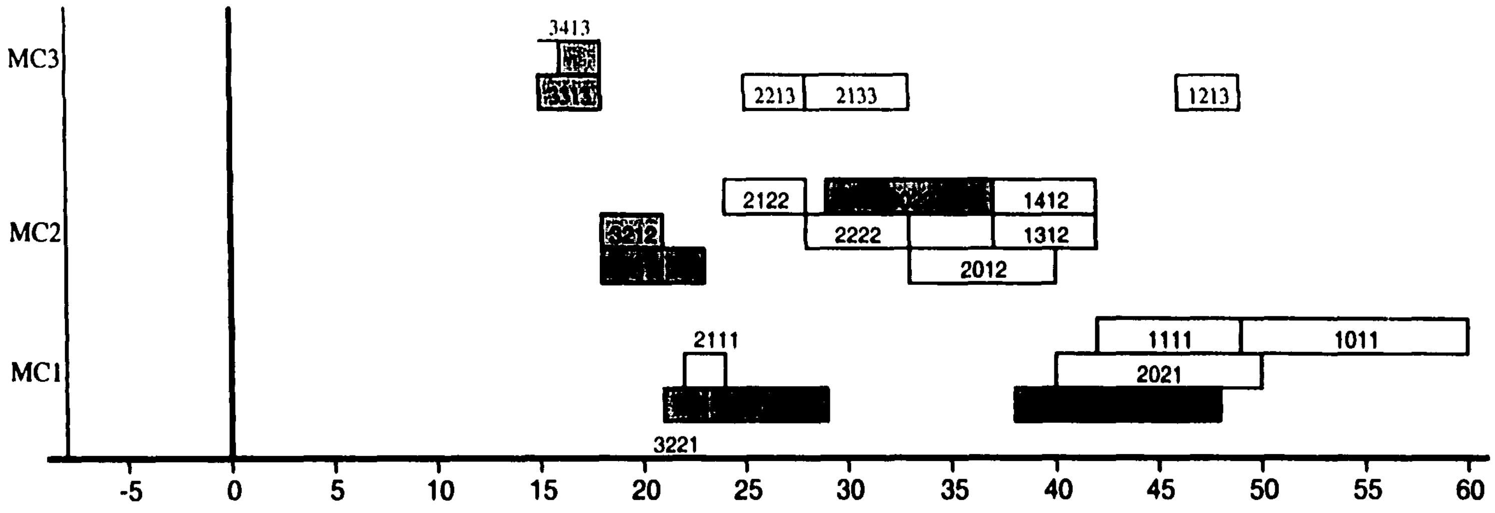


Figure 4.10 Ideal solution, for example 4.2, with virtual due-date "60-50-48"

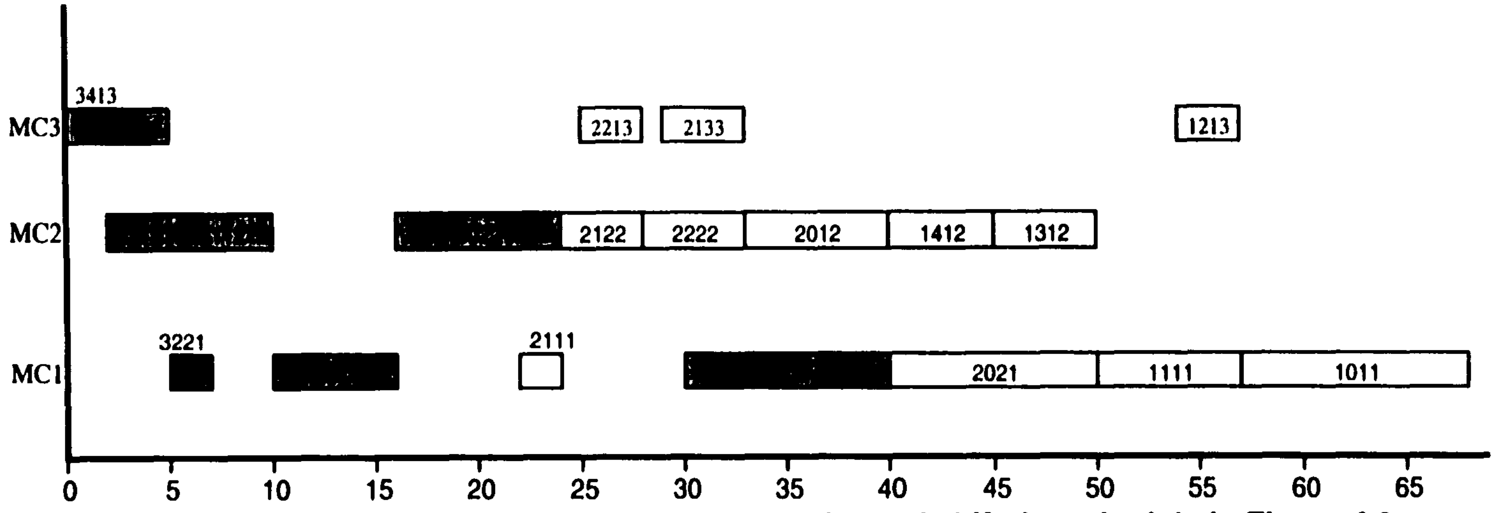


Figure 4.11 A feasible solution after doing rightward shift the schedule in Figure 4.9.

4.2.2.2 The case involving the realization of feasible solution after employing algorihm V

In this problem case, the schedule obtained from algorithm V is feasible. It implies that all products can be completed before or on their due-dates. But, this schedule may not be the best solution. If one forces some products to be tardy, the total cost could be reduced. This situation occurs, when the due-dates of products are very tight and the earliness costs of products and sub-jobs are high. Allowing some products to be tardy may help to relax the tightness of due-dates and consequently reduce overall cost. To deal with this situation, one needs to find the critical product from the initial schedule and the amount of tardiness necessary for the critical product. The critical product is the product that causes the tightness of the schedule. In other word, the critical product is the product that generates the highest earliness penalty of the schedule. To identify the critical product, in this research, products are removed from the schedule one at a time. After removing a product, the operations of the rest of the products are right shifted in time without violating any constraints. In this respect, the earliness cost of the existing products should be reduced. For the removed product, it can be assumed that it is tardy in the same amount of rightward shift of operations of unremoved products. If the amount of earliness cost saved by the rightward shift of unremoved products is larger than the tardiness cost of the removed product, then the removed product is a candidate product to be tardy.

To obtain a solution, the virtual due-date of the critical product is obtained by adding the amount of rightward shift to the original due-date. Based on the new virtual due-date, algorithm V is employed to solve the problem. As an illustration, consider a problem situation that involves products as Table 4.3 (Example 4.3), whose product structures are as shown in Figure 4.7. The processing time, earliness cost, and tardiness cost of each operation are as presented in Table 4.3. The actual due-dates of product 1 is *70*, product 2 is *80* and product 3 is *80*. The due-date can be represented as *"70-80-80"*.

Table 4.3 Parameters of products 1, 2 and 3 of example 4.3

Operation	Processing time (unit time)	Earliness Cost (cost/time)	Tardiness Cost (cost/time)	Operation	Processing time (unit time)	Earliness Cost (cost/time)	Tardiness Cost (cost/time)
1011	10	20	30	2222	7	10	-
1111	8	12	-	2213	3	2	-
1312	3	4	-	3031	10	20	30
1412	3	4	-	3022	9	12	-
1213	5	9	-	3011	6	10	-
2021	10	22	30	3112	4	7	-
2012	9	11	-	3313	3	1	-
2133	6	7	-	3413	3	2	-
2122	5	5	-	3221	6	10	-
2111	4	2	-	3212	4	4	-

The ideal solution and an initial feasible solution obtained from algorithm V are as shown in Figures 4.12 and 4.13, respectively. Then remove one product at a time from the initial schedule. In Figure 4.14, product 3 is removed from Figure 4.13, and then rightward shift is performed for all the operations of the unremoved products (i.e., product 1 and 2). There are two stages in doing rightward shift for products 1 and 2 without violating any constraints. Operations *1011* and *1111* can be rightward shifted *10* time units (see Fig.4.15). Operations *1312* and *1412* can be rightward shifted *2* time units without violating any constraints (see Fig. 4.16). Therefore, we have two possible stages (i.e., shift 2 and 10 unit times) of shifting the existing schedule of unremoved products. This also means that the possible amount of tardiness of the removed product (product 3) is either *2* or *10.* This concept is reasonable. Assume that one wants to shift the existing schedule of products 1 and 2 by *10* unit times for saving some earliness costs, then product 3 must be tardy, at least, by *10* unit times. At this point, one needs to identify the tardiness amount to be used (i.e., either *2* or *10*). The right tardiness amount can be determined by calculating the cost savings. If one shifts an existing and schedule by *10* unit times, earliness cost of *145* unit cost can be saved (see Fig. 4.15) and the tardiness cost of product 3 (i.e., *10* unit times tardiness) is *300* unit cost. Therefore, forcing product 3

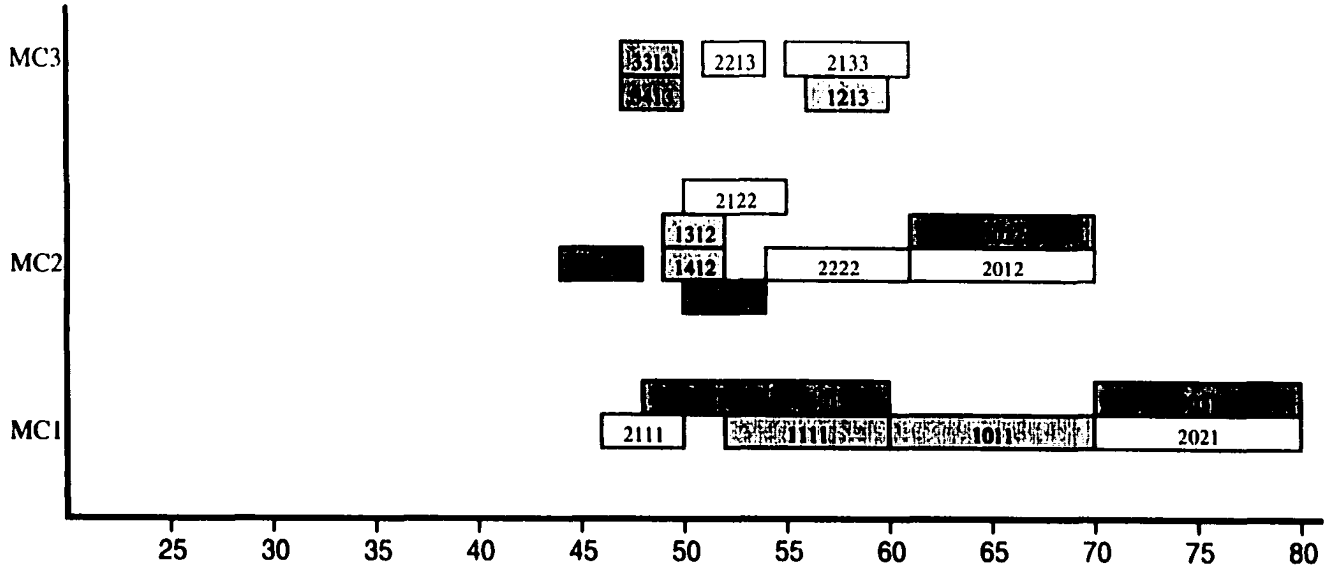


Figure 4.12 The ideal solution of example 4.3

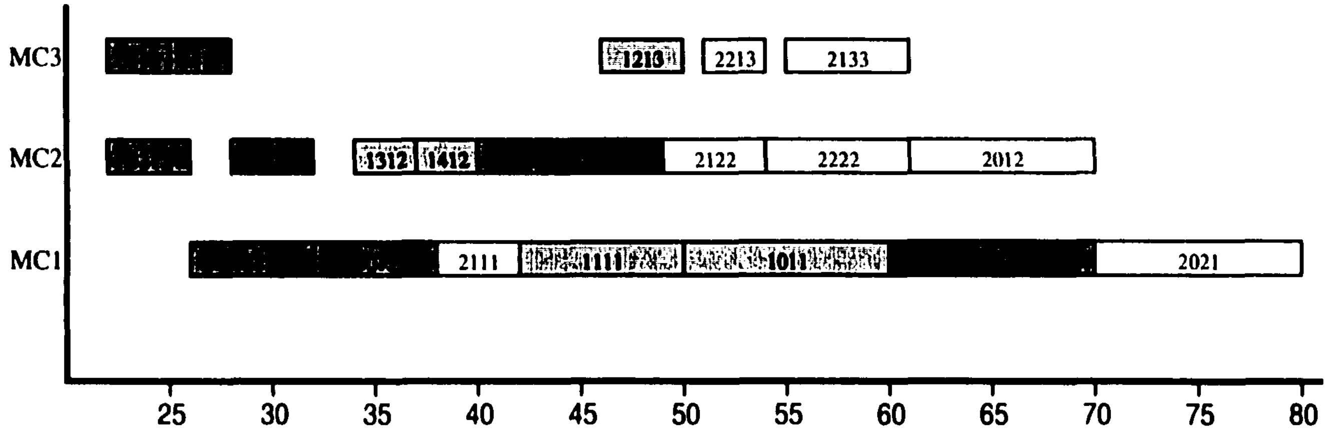


Figure 4.13 An initial feasible solution after employing algorithm V in example 4.3

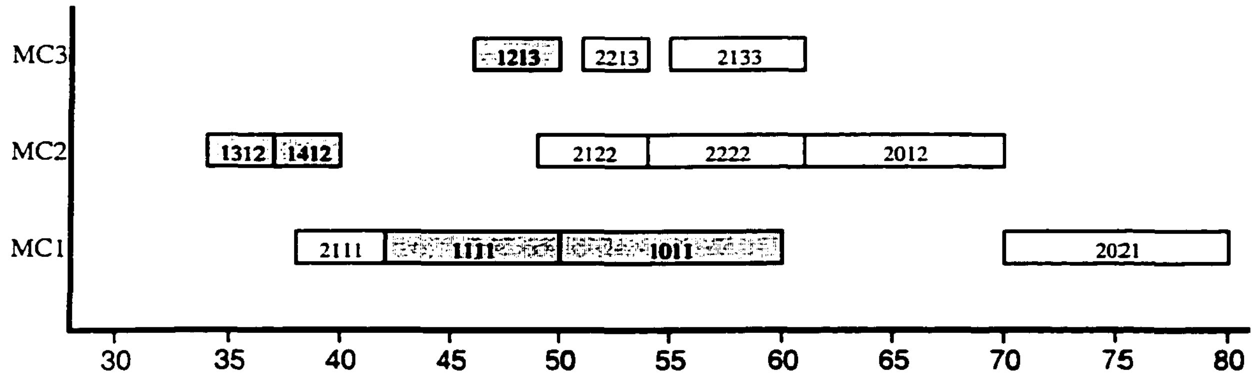


Figure 4.14 Schedule after removing product 3

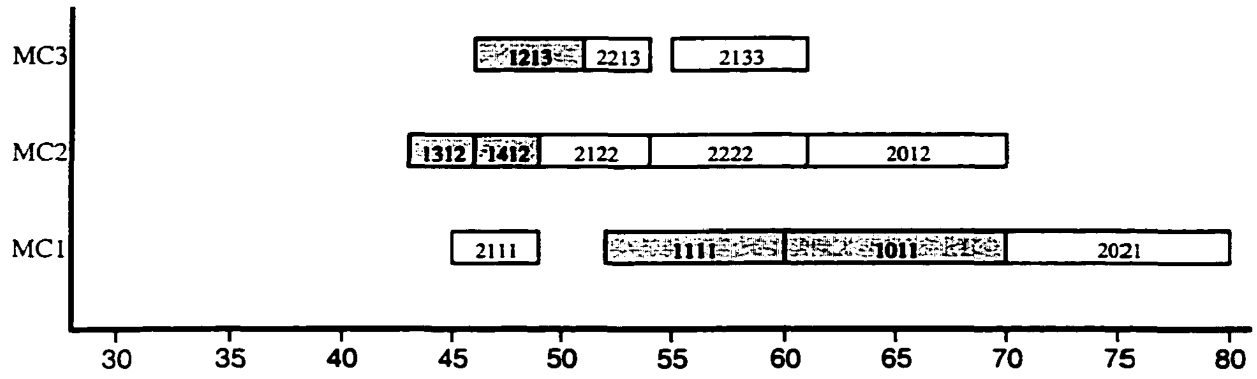


Figure 4.15 Schedule after removing product 3 and doing rightward shift 10 unit times for operations 1011 and 1111

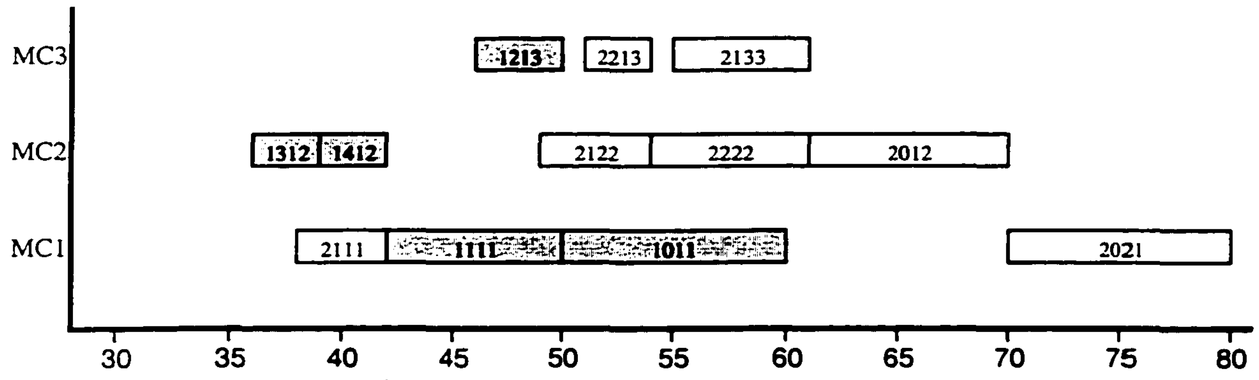


Figure 4.16 Schedule after removing product 3 and doing rightward shift 2 unit times for operations 1312 and 1412

to be tardy *10* unit times does not reduce cost, since the tardiness cost is larger than the earliness cost that can be saved. This process is also applied to force product 3 to be tardy for *2* unit times. It turns out that the tardiness cost is also larger than the earliness cost that can be saved. Therefore, product 3 is not the critical product. Forcing product 3 to be a tardy job does not help to reduce the overall cost. Similarly, the whole process is repeated by removing products 1 and 2. In example 4.3, in all possible cases of removing products 1 and 2 from the initial schedule, it is found that it is only by removing product 2 from the initial schedule (see Fig. 4.17) and carrying out rightward shift of existing schedule by *10* unit times (see Fig. 4.18) can reduction in the overall cost be realized. The earliness cost can be reduced by *502* unit costs and the tardiness cost of product 2 (i.e., *10* unit time tardiness) is *300* unit costs. Therefore, product 2 is a critical product. Forcing product 2 to be tardy by 10 unit times could help to reduce the overall cost. Now, the virtual due-date of product 2 can be set as *90*. With the due-date *"70-90-80"*, then algorithm V is employed to schedule all operations. The ideal solution and feasible solution for the due-date *"70-90-80"* are as shown in Figures 4.19 and 4.20, respectively. If the new solution is not better than the initial solution, then the algorithm stops. Otherwise, the new solution is set as the current best solution and the whole process is repeated again until no more improvement in solution is obtained.

4.2.3 Algorithm VI development

In this section, the step by step description of algorithm VI is presented. Algorithm VI is an integration of algorithm V and the methods previously mentioned in sections 4.2.2.1 and 4.2.2.2. The followings are additional notations and definitions used in the presentation of algorithm VI.

System variables

M: Set of feasible solutions.

B: Set of all operations.

P_i : Set of possible tardiness amount for product i.

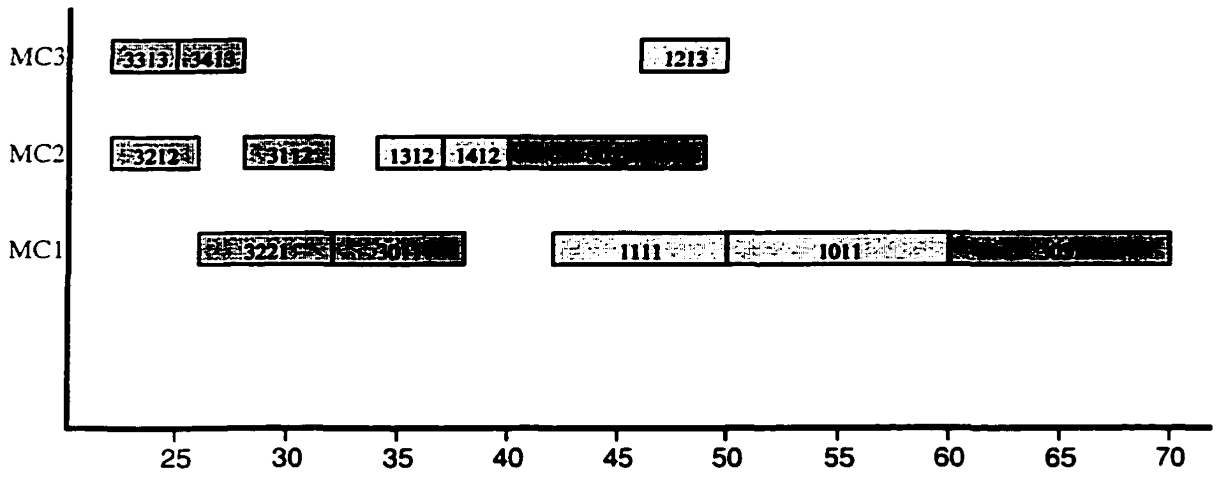


Figure 4.17 Schedule after removing product 2

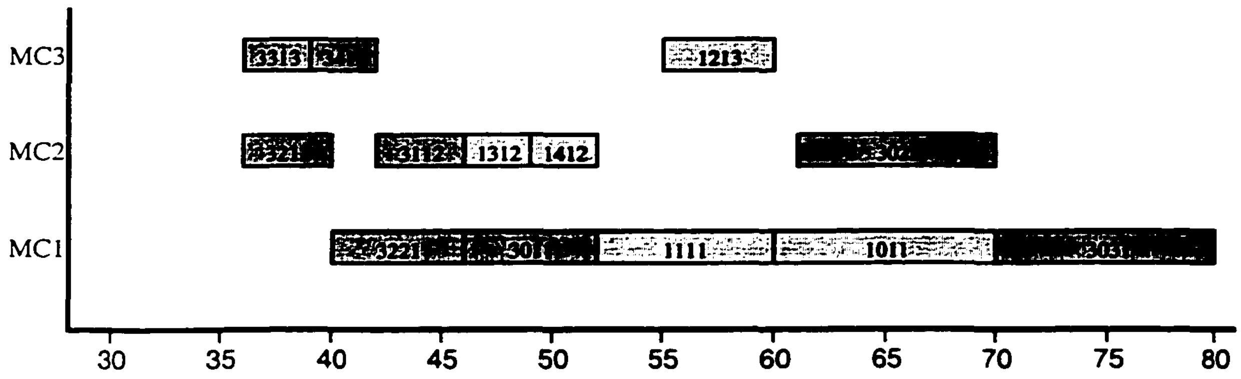


Figure 4.18 Schedule after removing product 2 and doing rightward shift of 10 unit times

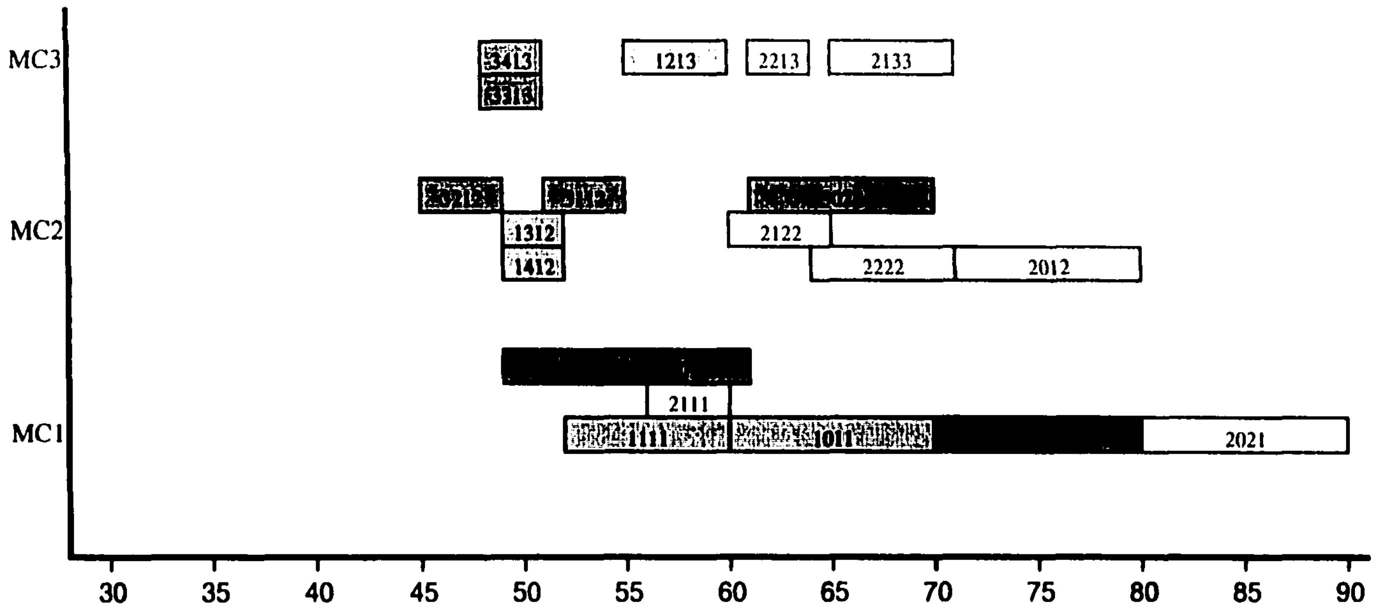


Figure 4.19 Ideal solution, for example 4.3, with virtual due-date " 70-90-80"

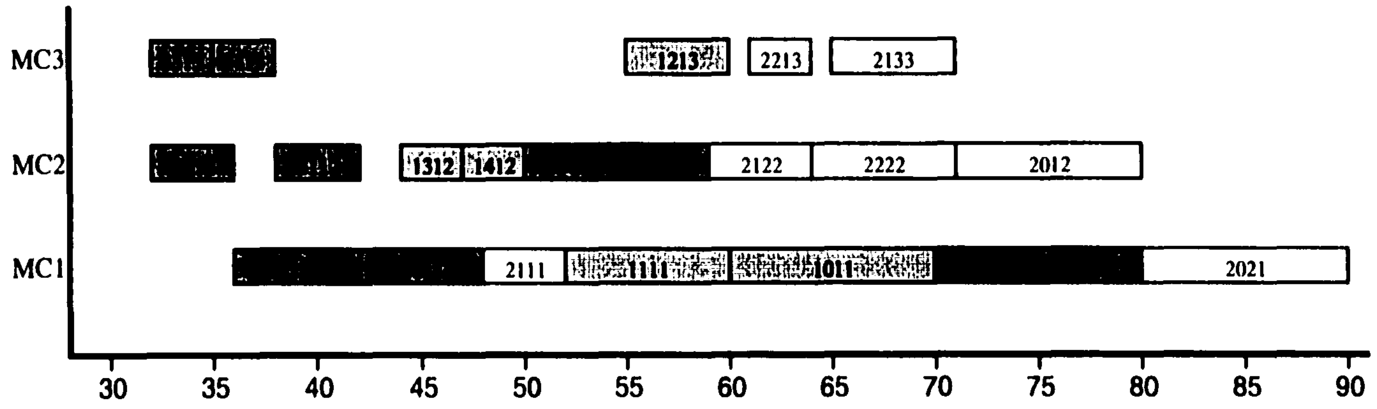


Figure 4.20 A feasible solution of the problem with virtual due-date "70-90-80"

Algorithm VI

Step 0. Set $M = \phi$ and $P_i = \phi$ for all i.

Step 1. Employ algorithm V to construct an initial schedule, called schedule R.

Step 2. Check feasibility of R. If R is a feasible solution (i.e., $S_{(ijkl)} \geq 0$, $\forall O_{(ijkl)}$), go to step 7. Otherwise go to step 3.

Step 3. While maintaining precedence relationship between operations, do rightward shift in R until feasible solution is obtained (i.e., $S_{(ijkl)} \geq 0$, $\forall O_{(ijkl)}$). Keep this solution as a feasible solution in M, where M is a set of feasible schedules. Set schedule back to R.

Step 4. Determine the earliest starting time of all operations B in R (i.e., $S_{(ijkl)^*} \leq \min_{O(ijkl) \in B} \{S_{(ijkl)}\}$), where B is the set of all operations in R. In this step, $S_{(ijkl)^*} \leq 0$.

Let G be the appropriate amount of tardiness. Set $G = |S_{(ijkl)^*}|$.

Step 5. For each product i, set the virtual due-date of the product equal to the actual due-date plus the appropriate amount of tardiness (i.e., virtual due-date of product $i = D_i + G$). and repeat the following steps (5a-5c).

5a. Based on the virtual due-date of product i and the actual due-dates of the other products, employ algorithm V to schedule the operations.

5b. If a feasible solution (i.e., $S_{(ijkl)} \geq 0$, for $\forall O_{(ijkl)}$) is obtained, place this solution in M. Otherwise do rightward shift operations until feasible solution is achieved (i.e.,$S_{(ijkl)} \geq 0$, for $\forall O_{(ijkl)}$) and place this solution as a feasible solution in M,

5c. Reset the schedule back to R.

Step 6. Select the best solution in M, called R', and set $R = R'$.

Step 7. Set $M = \phi$. For each product i in R, do the following steps (7a-7d).

7a. Remove all operations of product i from R.

7b. Let R^* be the schedule after removing product i from R. For each operation $O_{(ijkl)'}$ in R^*, do the steps (7b(1)-7b(2)).

7b(1). Shift $O_{(ijkl)'}$ to the right without violating precedence constraint.

Let $Q_{(ijkl)'}$ be the amount that $O_{(ijkl)'}$ can be shifted to the right in time. If $Q_{(ijkl)'} \geq 0$, place $Q_{(ijkl)'}$ in P_i where P_i is the set of possible tardiness amount of product *i*.

7b(2). Set the schedule back to R^*.

7c. Let $P_i(a)$ be a member of P_i. For each $P_i(a)$, do the following steps.

7c(1). Set virtual due-date of product *i* equal to the due-date of product *i* in *R* plus $P_i(a)$ (i.e., $D_{(i \text{ in } R)} + P_i(a)$).

7c(2). Based on the new virtual due-date of product *i* from 7c(1) and the due-dates of the other products in *R*, employ algorithm V to construct a new schedule for all the operations. Place the solution in set ***M***.

7d. Reset the schedule back to *R*.

Step 8. Select the best solution in *M*, called *R'*. If *R'* is better than *R*, set $R = R'$, set $M = \phi$ and return to step 7, otherwise stop.

A step by step presentation of algorithm V with example 4.2 is illustrated in Appendix C.

CHAPTER 5
HEURISTIC COMPARISON

In this chapter, the six developed heuristic algorithms are tested and compared to optimal solutions obtained by using exact solution procedure for some sample test problems. In the single machine problem, algorithm I and algorithm II are also compared with each other for the case of weighted earliness cost minimization. Comparisons between algorithms III, IV and the Ow & Morton dispatching algorithm [31] are also performed for the case of minimizing the sum of weighted earliness and weighted tardiness cost.

5.1 Heuristic comparison in single machine problem

5.1.1 Single machine problem with earliness cost minimization

Algorithms I and II were applied to 75 test problems. The problem sizes varied from 10 to 30 jobs. The processing times, the due-dates, and the weighted earliness penalty per unit time for each job were randomly generated. The results from both algorithms were also compared with the optimal solutions for the problem cases involving 10 jobs, and 15 jobs. The optimal solutions were obtained by applying the LINDO commercial software to solve the equivalent mathematical models on a PC with Pentium II processor and running at 233 MHz. In this research, if the optimal solution cannot be obtained within 2 hours by LINDO, the problem is aborted. Aborted test problems had no optimal solution to report and were instead marked as being nonapplicable (N/A) on the comparison table.

Of the 75 test problems examined, 19 were solved to optimality within two hours using mathematical programming approach (see Table 5.1-5.2). Algorithm II also found the optimal solutions for all the 19 problems, while heuristic algorithm I found the optimal solutions for 18 problems. For the remainder of the problems, exact solution procedure was no longer used because of the increasing problem sizes. However, the two algorithms were used in solving the problem. The results obtained are shown in Tables 5.3-5.5. From the results given, it was found that algorithm I outperformed algorithm II. The quality of the solutions from algorithm I

Table 5.1 Comparison of solutions for problems with 10 jobs

Pro. #	Opt. Sol. (Unit Cost) (A)	Alg. I Sol. (unit cost) (B)	Alg. II Sol. (unit cost) (C)	%dev of B from A	%dev of C from A	%dev of C from B	CPU time (A) (sec.)	CPU time (B) (sec.)	CPU time (C) (sec.)
1	81	81	81	0	0	0	6	<1	<1
2	286	293	286	2.44	0	-2.44	4	<1	<1
3	112	112	112	0	0	0	3	<1	<1
4	83	83	83	0	0	0	13	<1	<1
5	97	97	97	0	0	0	11	<1	<1
6	299	299	299	0	0	0	43	<1	<1
7	64	64	64	0	0	0	4	<1	<1
8	186	186	186	0	0	0	8	<1	<1
9	421	421	421	0	0	0	139	<1	<1
10	271	271	271	0	0	0	8	<1	<1
11	405	405	405	0	0	0	53	<1	<1
12	121	121	121	0	0	0	4	<1	<1
13	229	229	229	0	0	0	6	<1	<1
14	225	225	225	0	0	0	13	<1	<1
15	208	208	208	0	0	0	39	<1	<1
Average of % deviation				0.16	0	-0.16			
Standard deviation of % deviation				0.63	0	0.63			

Table 5.2 Comparison of solutions for problems with 15 jobs

Pro. #	Opt. Sol. (Unit Cost) (A)	Alg. I Sol. (unit cost) (B)	Alg. II Sol. (unit cost) (C)	%dev of B from A	%dev of C from A	%dev of C from B	CPU Time (A) (sec.)	CPU time (B) (sec.)	CPU time (C) (sec.)
1	N/A	1219	1251	N/A	N/A	2.63	> 2 hr.	<1	<1
2	N/A	762	762	N/A	N/A	0.00	> 2 hr.	<1	<1
3	135	135	135	0	0	0.00	105	<1	<1
4	406	406	406	0	0	0.00	2hr. 1min.	<1	<1
5	N/A	852	880	N/A	N/A	3.29	> 2hr.	<1	<1
6	N/A	581	581	N/A	N/A	0.00	> 2hr.	<1	<1
7	N/A	1041	1041	N/A	N/A	0.00	> 2hr.	<1	<1
8	226	226	226	0	0	0.00	1090	<1	<1
9	153	153	153	0	0	0.00	364	<1	<1
10	N/A	197	202	N/A	N/A	2.54	> 2hr.	<1	<1
11	N/A	540	545	N/A	N/A	0.93	> 2hr.	<1	<1
12	N/A	1033	1033	N/A	N/A	0.00	> 2hr.	<1	<1
13	N/A	218	225	N/A	N/A	3.21	> 2hr.	<1	<1
14	N/A	286	283	N/A	N/A	-1.05	> 2hr.	<1	<1
15	N/A	313	313	N/A	N/A	0.00	> 2hr.	<1	<1
Average of % deviation						0.77			
Standard deviation of % deviation						1.40			

- N/A : the optimal solution could not be obtained within 2 hours.

Table 5.3 Comparison of solutions for problems with 20 jobs

Pro. #	Alg. I Sol. (unit cost)	Alg. II Sol. (unit cost)	% deviation of Alg II from Alg I	Algorithm I CPU time (sec.)	AlgorithmII CPU time (sec.)
1	917	925	0.87	<1	<1
2	762	762	0.00	<1	<1
3	135	135	0.00	<1	<1
4	406	406	0.00	<1	<1
5	852	880	3.29	<1	<1
6	581	581	0.00	<1	<1
7	1041	1041	0.00	<1	<1
8	226	226	0.00	<1	<1
9	153	153	0.00	<1	<1
10	197	202	2.54	<1	<1
11	540	545	0.93	<1	<1
12	1033	1033	0.00	<1	<1
13	218	225	3.21	<1	<1
14	286	283	-1.05	<1	<1
15	313	313	0.00	<1	<1
Average of % deviation			0.65		
Standard deviation of % deviation			1.30		

- Algorithm I is better than algorithm II by 0.65 % on the average with standard deviation of 1.30.

Table 5.4 Comparison of solutions for problems with 25 jobs

Pro. #	Alg. I Sol. (unit cost)	Alg. II Sol. (unit cost)	% deviation of Alg II from Alg I	Algorithm I CPU time (sec.)	AlgorithmII CPU time (sec.)
1	929	917	-1.29	<2	<2
2	1278	1278	0.00	<2	<2
3	756	752	-0.53	<2	<2
4	1112	1078	-3.06	<2	<2
5	512	532	3.91	<2	<2
6	980	1120	14.29	<2	<2
7	2630	2759	4.90	<2	<2
8	2284	2369	3.72	<2	<2
9	2033	2230	9.69	<2	<2
10	1745	1675	-4.01	<2	<2
11	1207	1207	0.00	<2	<2
12	1477	1547	4.74	<2	<2
13	1484	1499	1.01	<2	<2
14	1479	1461	-1.22	<2	<2
15	2144	2106	-1.77	<2	<2
Average of % deviation			2.03		
Standard deviation of % deviation			4.80		

- Algorithm I is better than the algorithm II by 2.03% on the average with standard deviation of 4.8.

Table 5.5 Comparison of solutions for problems with 30 jobs

Pro. #	Alg. I Sol. (unit cost)	Alg. II Sol. (unit cost)	% deviation of Alg II from Alg I	Algorithm I CPU time (sec.)	AlgorithmII CPU time (sec.)
1	1750	1737	-0.74	<2	<2
2	3005	3001	-0.13	<2	<2
3	707	707	0.00	<2	<2
4	626	626	0.00	<2	<2
5	1282	1282	0.00	<2	<2
6	1158	1264	9.15	<2	<2
7	985	1044	5.99	<2	<2
8	1241	1237	-0.32	<2	<2
9	454	454	0.00	<2	<2
10	1371	1557	13.57	<2	<2
11	1870	1868	-0.11	<2	<2
12	5171	5167	-0.08	<2	<2
13	2182	2386	9.35	<2	<2
14	2892	2997	3.63	<2	<2
15	1284	1284	0.00	<2	<2
Average of % deviation			2.69		
Standard deviation of %dev.			4.45		

- algorithm I is better than the algorithm II by 2.69% on the average with standard deviation of 4.45.

are better than those obtained by algorithm II by approximately 1.09%. In all cases, it took less than 2 seconds of computational time to solve the problems by each of the two algorithms. The sensitivity analysis of the problem is presented in Appendix D.

5.1.2 Single machine problem with sum of the weighted earliness and weighted tardiness cost minimization

Algorithms III and algorithm IV were employed to solve 75 test problems. The problem sizes varied from 10 to 30 jobs. The processing times, the due-dates, the earliness, and the tardiness penalties for each job are randomly generated. The results from both algorithms are compared with the optimal solutions obtained for the cases involving 10 and 15 job problems. The optimal solutions are obtained by applying the LINDO commercial software to solve the equivalent mathematical models on a PC with Pentium II processor and running at 233 MHz. In this research, if the optimal solutions can not be obtained within 2 hours by LINDO, the model is aborted and no solution is reported for comparison. Aborted cases are denoted as N/A on the comparison tables. The solutions from both algorithms are also compared with the Ow & Morton's dispatch algorithm. The results of the comparisons are given in Tables 5.6 through 5.12. The sensitivity analysis of the problem is presented in Appendix D.

Table 5.6 Comparison of solutions for the 10 job problems with optimal solutions

Pro. #	Optimal Sol. (Unit Cost) (A)	Alg. III Sol. (unit cost) (B)	Alg. IV Sol. (unit cost) (C)	%dev of B from A	%dev. of C from A	%dev of B fromC	CPU time (A) (sec.)	CPU time (B) (sec.)	CPU Time (C) (sec.)
1	178	178	183	0	2.8	-2.73	952	<1	<1
2	80	80	80	0	0	0.00	39	<1	<1
3	76	77	77	1.3	1.3	0.00	45	<1	<1
4	171	177	171	3.5	0	3.51	50	<1	<1
5	249	249	249	0	0	0.00	408	<1	<1
6	77	77	77	0	0	0.00	65	<1	<1
7	247	260	260	5.3	5.3	0.00	574	<1	<1
8	142	153	153	7.7	7.7	0.00	346	<1	<1
9	240	243	240	1.2	0	1.25	237	<1	<1
10	178	178	178	0	0	0.00	444	<1	<1
11	162	167	165	3	1.8	1.21	468	<1	<1
12	89	89	89	0	0	0.00	247	<1	<1
13	234	234	234	0	0	0.00	234	<1	<1
14	375	375	375	0	0	0.00	773	<1	<1
15	N/A	669	669	N/A	N/A	0.00	> 2hr.	<1	<1
Average of % deviation				1.57	1.35	0.22			
Standard deviation of % deviation				2.44	2.40	1.26			

N/A: The optimal solution can not be obtained within 2 hours.
- Algorithm III obtains on average deviation of 1.57% from the optimal with standard deviation of 2.44.
- Algorithm IV obtains on average deviation of 1.35% from the optimal with standard deviation of 2.40.
- Algorithm IV is better than the algorithm III by 0.22% on the average with standard deviation of 1.26.

Table 5.7 Comparison of solutions for the 10 job problems with solutions from Ow & Morton algorithm

Pro. #	O&M Alg. Sol. (Unit Cost) (D)	Alg. III Sol. (unit cost) (B)	Alg. IV Sol. (unit cost) (C)	%dev. of D from B	%dev. of D from C	%dev. of B from C	CPU time (D) (sec.)	CPU time (B) (sec.)	CPU Time (C) (sec.)
1	216	178	183	21.35	18.03	-2.73	<1	<1	<1
2	110	80	80	37.50	37.50	0.00	<1	<1	<1
3	81	77	77	5.19	5.19	0.00	<1	<1	<1
4	184	177	171	3.95	7.60	3.51	<1	<1	<1
5	274	249	249	10.04	10.04	0.00	<1	<1	<1
6	95	77	77	23.38	23.38	0.00	<1	<1	<1
7	265	260	260	1.92	1.92	0.00	<1	<1	<1
8	168	153	153	9.80	9.80	0.00	<1	<1	<1
9	309	243	240	27.16	28.75	1.25	<1	<1	<1
10	244	178	178	37.08	37.08	0.00	<1	<1	<1
11	192	167	165	14.97	16.36	1.21	<1	<1	<1
12	107	89	89	20.22	20.22	0.00	<1	<1	<1
13	238	234	234	1.71	1.71	0.00	<1	<1	<1
14	400	375	375	6.67	6.67	0.00	<1	<1	<1
15	823	669	669	23.02	23.02	0.00	<1	<1	<1
Average of % deviation				16.26	16.49	0.22			
Standard deviation of % deviation				11.97	11.78	1.26			

- Algorithm III is better than the O & M algorithm by 16.29% on the average with standard deviation of 11.97.
- Algorithm IV is better than the O & M algorithm by 16.49% on the average with standard deviation of 11.78.

Table 5.8 Comparison of solutions for the 15 job problems with optimal solutions

Pro. #	Optimal Sol. (Unit Cost) (A)	Alg. III Sol. (unit cost) (B)	Alg. IV Sol. (unit cost) (C)	%dev of B from A	%dev. of C from A	%dev of B fromC	CPU time (A) (sec.)	CPU time (B) (sec.)	CPU time (C) (sec.)
1	202	202	202	0	0	0.00	32 min.	<1	<1
2	131	131	131	0	0	0.00	17 min.	<1	<1
3	N/A	261	274	N/A	N/A	-4.74	> 2 hr.	<1	<1
4	N/A	228	219	N/A	N/A	4.11	> 2 hr.	<1	<1
5	N/A	275	275	N/A	N/A	0.00	> 2 hr.	<1	<1
6	N/A	258	258	N/A	N/A	0.00	> 2 hr.	<1	<1
7	242	242	242	0	0	0.00	1hr.46min.	<1	<1
8	N/A	799	799	N/A	N/A	0.00	> 2 hr.	<1	<1
9	N/A	1067	1067	N/A	N/A	0.00	> 2 hr.	<1	<1
10	N/A	529	582	N/A	N/A	-9.11	> 2hr.	<1	<1
11	N/A	705	705	N/A	N/A	0.00	> 2hr.	<1	<1
12	N/A	549	549	N/A	N/A	0.00	> 2hr.	<1	<1
13	N/A	584	584	N/A	N/A	0.00	> 2hr.	<1	<1
14	N/A	699	699	N/A	N/A	0.00	> 2hr.	<1	<1
15	N/A	1131	1130	N/A	N/A	0.09	> 2hr.	<1	<1
Average of % deviation				0	0	-0.64			
Standard deviation of % deviation				0	0	2.88			

N/A : the optimal solution can not be obtained within 2 hours.
- Algorithm III is better than the algorithm IV by 0.64% on the average with standard deviation of 2.88.

Table 5.9 Comparison of solutions for the 15 job problems with solutions from Ow & Morton algorithm

Pro. #	O&M alg. Sol. (Unit Cost) (D)	Alg. III Sol. (unit cost) (B)	Alg. IV Sol. (unit cost) (C)	%dev. of D from B	%dev. of D from C	%dev. of B from C	CPU time (D) (sec.)	CPU time (B) (sec.)	CPU time (C) (sec.)
1	226	202	202	11.88	11.88	0.00	<1	<1	<1
2	151	131	131	15.27	15.27	0.00	<1	<1	<1
3	280	261	274	7.28	2.19	-4.74	<1	<1	<1
4	261	228	219	14.47	19.18	4.11	<1	<1	<1
5	527	275	275	91.64	91.64	0.00	<1	<1	<1
6	462	258	258	79.07	79.07	0.00	<1	<1	<1
7	358	242	242	47.93	47.93	0.00	<1	<1	<1
8	868	799	799	8.64	8.64	0.00	<1	<1	<1
9	1422	1067	1067	33.27	33.27	0.00	<1	<1	<1
10	959	529	582	81.29	64.78	-9.11	<1	<1	<1
11	811	705	705	15.04	15.04	0.00	<1	<1	<1
12	570	549	549	3.83	3.83	0.00	<1	<1	<1
13	657	584	584	12.50	12.50	0.00	<1	<1	<1
14	788	699	699	12.73	12.73	0.00	<1	<1	<1
15	1180	1131	1130	4.33	4.42	0.09	<1	<1	<1
Average of % deviation				29.28	28.16	-0.64			
Standard deviation of % deviation				30.58	28.98	2.88			

-Algorithm III is better than the O& M algorithm by 29.28% on the average with standard deviation of 30.58.
-Algorithm IV is better than the O&M algorithm by 28.16% on the average. with standard deviation of 28.98.

Table 5.10 Comparison of solutions for the 20 job problems

Pro. #	O&M alg. Sol. (Unit Cost) (D)	Alg. III Sol. (unit cost) (B)	Alg. IV Sol. (unit cost) (C)	%dev. of D from B	%dev. of D from C	%dev. of B from C	CPU time (D) (sec.)	CPU time (B) (sec.)	CPU time (C) (sec.)
1	583	430	430	35.58	35.58	0.00	< 1	< 1	< 1
2	794	665	665	19.40	19.40	0.00	< 1	< 1	< 1
3	334	221	221	51.13	51.13	0.00	< 1	< 1	< 1
4	742	659	617	12.59	20.26	6.81	< 1	< 1	< 1
5	1128	990	868	13.94	29.95	14.06	< 1	< 1	< 1
6	1430	1025	1069	39.51	33.77	-4.12	< 1	< 1	< 1
7	1088	1037	860	4.92	26.51	20.58	< 1	< 1	< 1
8	1340	920	1069	45.65	25.35	-13.94	< 1	< 1	< 1
9	426	372	363	14.52	17.36	2.48	< 1	< 1	< 1
10	1443	1011	903	42.73	59.80	11.96	< 1	< 1	< 1
11	426	499	426	-14.63	0.00	17.14	< 1	< 1	< 1
12	546	504	515	8.33	6.02	-2.14	< 1	< 1	< 1
13	766	697	687	9.90	11.50	1.46	< 1	< 1	< 1
14	780	636	633	22.64	23.22	0.47	< 1	< 1	< 1
15	712	595	588	19.66	21.09	1.19	< 1	< 1	< 1
Average of % deviation				21.73	25.40	3.73			
Standard deviation of % deviation				17.96	15.60	8.95			

- Algorithm III is better than the O&M algorithm by 21.73% on the average with standard deviation of 17.96.
- Algorithm IV is better than the O&M algorithm by 25.40% on the average with standard deviation of 15.60.
- Algorithm IV is better than algorithm III by 3.73% on the average with standard deviation of 8.95.

Table 5.11 Comparison of solutions for the 25 job problems

Pro. #	O&M alg. Sol. (Unit Cost) (D)	Alg. III Sol. (unit cost) (B)	Alg. IV Sol. (unit cost) (C)	%dev. of D from B	%dev. of D from C	%dev. of B from C	CPU time (D) (sec.)	CPU time (B) (sec.)	CPU time (C) (sec.)
1	1206	1033	1023	16.75	17.89	0.98	< 1	<2	<2
2	568	469	446	21.11	27.35	5.16	< 1	<2	<2
3	827	562	604	47.15	36.92	-6.95	< 1	<2	<2
4	555	436	414	27.29	34.06	5.31	< 1	<2	<2
5	520	305	315	70.49	65.08	-3.17	< 1	<2	<2
6	322	218	218	47.71	47.71	0.00	< 1	<2	<2
7	1493	1121	1121	33.18	33.18	0.00	< 1	<2	<2
8	748	708	683	5.65	9.52	3.66	< 1	<2	<2
9	695	558	558	24.55	24.55	0.00	< 1	<2	<2
10	699	621	545	12.56	28.26	13.94	< 1	<2	<2
11	649	632	600	2.69	8.17	5.33	< 1	<2	<2
12	931	611	606	52.37	53.63	0.83	< 1	<2	<2
13	996	885	885	12.54	12.54	0.00	< 1	<2	<2
14	427	378	371	12.96	15.09	1.89	< 1	<2	<2
15	1336	1192	1132	12.08	18.02	5.30	< 1	<2	<2
Average of % deviation				26.61	28.80	2.15			
Standard deviation of % deviation				19.71	16.71	4.73			

- Algorithm III is better than the O & M algorithm by 26.61% on the average with standard deviation of 19.71.
- Algorithm IV is better than the O & M algorithm by 28.80% on the average with standard deviation of 16.71.
- Algorithm IV is better than the algorithm III by 2.15% on the average with standard deviation of 4.73.

Table 5.12 Comparison of solutions for the 30 job problems

Pro. #	O&M alg. Sol. (Unit Cost) (D)	Alg. III Sol. (unit cost) (B)	Alg. IV Sol. (unit cost) (C)	%dev. of D from B	%dev. of D from C	%dev. of B from C	CPU time (D) (sec.)	CPU time (B) (sec.)	CPU time (C) (sec.)
1	1159	1062	894	9.13	29.64	18.79	< 1	<2	<2
2	951	777	732	22.39	29.92	6.15	< 1	<2	<2
3	1395	810	852	72.22	63.73	-4.93	< 1	<2	<2
4	1268	1084	1120	16.97	13.21	-3.21	< 1	<2	<2
5	1057	941	851	12.33	24.21	10.58	< 1	<2	<2
6	1068	908	891	17.62	19.87	1.91	< 1	<2	<2
7	1032	901	883	14.54	16.87	2.04	< 1	<2	<2
8	1229	1095	1077	12.24	14.11	1.67	< 1	<2	<2
9	1088	654	672	66.36	61.90	-2.68	< 1	<2	<2
10	427	398	431	7.29	-0.93	-7.66	< 1	<2	<2
11	1598	1338	1330	19.43	20.15	0.60	< 1	<2	<2
12	1409	935	894	50.70	57.61	4.59	< 1	<2	<2
13	1300	1055	1031	23.22	26.09	2.33	< 1	<2	<2
14	1357	1110	1021	22.25	32.91	8.72	< 1	<2	<2
15	1552	1249	1306	24.26	18.84	-4.36	< 1	<2	<2
Average of % deviation				26.06	28.54	2.30			
Standard deviation of % deviation				20.27	18.77	6.85			

- Algorithm III is better than the O & M algorithm by 26.06% on the average with standard deviation 20.27.
- Algorithm IV is better than the O & M algorithm by 28.54% on the average with standard deviation 18.77.
- Algorithm IV is better than the algorithm III by 2.30% on the average with standard deviation 6.85.

The results show that algorithm IV, the tabu search, is the best algorithm for solving the earliness/tardiness problems compared to algorithm III and the Ow & Morton algorithm. It outperforms algorithm III and Ow & Morton dispatch algorithm by about 1.55% and 25.48 %, respectively, on average. Algorithm IV is relatively better than algorithm III when the size of the problem is increased. Both algorithms are much better than the Ow & Morton algorithm at all levels. The computational times for both algorithms are also much faster than those obtained by solving the corresponding mathematical models by the LINDO software.

In the Ow & Morton algorithm, inserted idle time is not allowed on the machine and this is different from that of algorithms III and IV. Thus, algorithms III and IV can be compared to the Ow & Morton algorithm to some degrees.

Although algorithm IV performs better than algorithm III, it has one key drawback. The algorithm (i.e., algorithm IV) depends on two major parameters, *k* (in Ow & Morton algorithm) and the *tabu size* while the algorithm III does not depend on any prespecified parameters.

5.2 Heuristic comparison in assembly job shop problem

5.2.1 Assembly job shop problem with earliness cost minimization

Algorithm V was employed to solve 50 test problems. The problem sizes are varied from 2-product and 3-machine (2P / 3M) to 5-product and 5 machine (5P / 5M). The number of operations in the problems is varied from 12 to 42 operations. Each product has its own product structure. The due-dates and tardiness penalty of products, the processing times, and the earliness penalty for operations are randomly generated. The results from algorithm V are compared to the optimal solutions obtained by applying the LINDO commercial software to solve the equivalent mathematical models on PC with Pentium III processor and running at 550 MHz. The results of the comparisons are given in Table 5.13.

Table 5.13 Comparison of solutions from algorithm V and optimal solutions

No.	Product/ Machine	# Oper.	Opt. Sol (unit cost)	Alg V (unit cost)	%diff Alg V from Opt	CPU time Opt. (sec)	CPU time Alg. V (sec)
1	2P / 3M	12	264	264	0.00	2	<1
2	2P / 3M	12	160	160	0.00	2	<1
3	2P / 3M	12	56	56	0.00	1	<1
4	2P / 3M	12	153	153	0.00	1	<1
5	2P / 3M	12	270	270	0.00	2	<1
6	2P / 3M	12	235	235	0.00	2	<1
7	3P / 3M	22	311	311	0.00	97	<1
8	3P / 3M	22	457	457	0.00	119	<1
9	3P / 3M	22	297	298	0.34	95	<1
10	3P / 3M	22	432	432	0.00	72	<1
11	3P / 3M	22	322	322	0.00	156	<1
12	3P / 3M	22	590	591	0.17	1112	<1
13	3P / 3M	22	432	432	0.00	239	<1
14	3P / 3M	22	374	374	0.00	415	<1
15	3P / 3M	22	159	159	0.00	23	<1
16	3P / 3M	22	245	245	0.00	208	<1
17	3P / 3M	22	415	421	1.45	376	<1
18	3P / 3M	22	315	321	1.90	203	<1
19	3P / 3M	22	355	355	0.00	136	<1
20	3P / 3M	22	267	267	0.00	41	<1
21	3P / 3M	22	500	500	0.00	10	<1
22	3P / 3M	20	509	509	0.00	77	<1
23	3P / 3M	20	499	499	0.00	34	<1
24	3P / 3M	20	193	193	0.00	32	<1
25	3P / 3M	20	214	214	0.00	6	<1
26	3P / 3M	20	602	610	1.33	51	<1

Table 5.13 (continued)

No.	Product/ Machine	# Oper.	Opt. Sol (unit cost)	Alg V (unit cost)	%diff Alg V from Opt	CPU time Opt. (sec)	CPU time Alg. V (sec)
27	3P / 3M	20	301	301	0.00	6	<1
28	3P / 3M	20	579	579	0.00	201	<1
29	3P / 3M	31	1019	1019	0.00	11702	<1
30	3P / 3M	31	619	619	0.00	9659	<1
31	4P / 4M	27	444	471	6.08	785	<1
32	4P / 4M	27	461	471	2.17	96	<1
33	4P / 4M	30	572	572	0.00	484	<1
34	4P / 4M	30	601	607	1.00	1011	<1
35	4P / 4M	30	581	584	0.52	1638	<1
36	4P / 4M	30	705	705	0.00	2934	<1
37	4P / 4M	27	542	556	2.58	1047	<1
38	4P / 4M	29	491	529	7.74	3232	<1
39	4P / 4M	29	350	350	0.00	742	<1
40	4P / 4M	29	530	530	0.00	466	<1
41	4P / 4M	29	334	334	0.00	171	<1
42	4P / 4M	29	500	500	0.00	938	<1
43	4P / 4M	29	770	794	3.12	4618	<1
44	4P / 4M	29	608	614	0.99	393	<1
45	4P / 4M	29	601	601	0.00	1433	<1
46	4P / 4M	29	632	632	0.00	662	<1
47	4P / 4M	29	737	746	1.22	899	<1
48	5P / 5M	37	453	453	0.00	1314	<1
49	5P / 5M	42	637	637	0.00	11720	<1
50	5P / 5M	37	720	742	3.06	11700	<1
Average of %difference					0.67		
Standard deviation of % difference					1.54		

Of the 50 test problems examined (see Table 5.13), algorithm V found the optimal solutions for 34 problems. The largest deviation from optimal is about *7.74%*. In all the test problems, algorithm V obtained an average deviation solutions of *0.67%* from the optimal with a standard deviation of *1.54*. The computational requirements for solving the problems by algorithm V are less than 1 second in all test problems. They are much less than the computational times required by the optimal solution procedure.

5.2.2 Assembly job shop problem with the sum of weighted earliness and weighted tardiness cost minimization

To test the efficiency of algorithm VI, more than 50 sample problems were generated and tested. But it was not possible to find optimal solutions for all problems by LINDO software because of the storage and computational load required. LINDO was out of memory for some problems, especially with the large size problems (4-product and 4-machine, 5-product and 5-machine). Of the more than 50 sample problems solved, 50 of them were solved optimally by the LINDO software on a PC with Pentium III processor and running at 550 MHz. Algorithm VI was applied to solve these 50 test problems. The problem sizes are varied from 2-product and 3-machine (2P / 3M) to 4-product and 4-machine (4P / 4M). The number of operations is varied from 12 to 30 operations. Each product has its own product structure. The due-dates and tardiness penalties of the products are randomly generated. The processing times, and the earliness penalties of operations are also randomly generated. The results of the comparisons for the 50 test problems solved to optimality by LINDO are shown in Table 5.14.

Table 5.14 Comparison of solutions from algorithm VI and optimal solutions

No.	Product/ Machine	# Oper.	Opt. Sol (unit cost)	Alg VI (unit cost)	%diff Alg VI from Opt	CPU time Opt. (sec)	CPU time Alg 6 (sec)	Comment[a] (Tardiness Type)
1	2P/3M	12	275	275	0.00	7	<1	1
2	2P/3M	12	458	458	0.00	6	<1	1
3	2P/3M	12	513	518	0.97	3	<1	1
4	2P/3M	12	284	284	0.00	5	<1	1
5	2P/3M	12	530	530	0.00	3	<1	1
6	2P/3M	12	347	347	0.00	7	<1	2
7	2P/3M	12	487	487	0.00	10	<1	2
8	2P/3M	12	631	631	0.00	5	<1	2
9	2P/3M	12	556	556	0.00	5	<1	2
10	2P/3M	12	680	680	0.00	5	<1	2
11	3P/3M	20	300	300	0.00	40	<1	1
12	3P/3M	20	479	479	0.00	232	<1	1&2
13	3P/3M	20	980	980	0.00	181	<1	1
14	3P/3M	20	798	798	0.00	185	<1	1
15	3P/3M	20	299	299	0.00	684	<1	1&2
16	3P/3M	20	886	886	0.00	1059	<1	1
17	3P/3M	20	625	629	0.64	1280	<1	2
18	3P/3M	20	682	682	0.00	631	<1	2

Table 6.14 (continued)

No.	Product/ Machine	# Oper.	Opt. Sol (unit cost)	Alg VI (unit cost)	%diff Alg VI from Opt	CPU time Opt. (sec)	CPU time Alg 6 (sec)	Comment [a] (Tardiness Type)
19	3P/3M	20	682	752	10.26	917	<1	2
20	3P/3M	20	746	770	3.22	298	<1	2
21	3P/3M	20	846	846	0.00	279	<1	2
22	3P/3M	20	958	958	0.00	1232	<1	2
23	3P/3M	20	1025	1025	0.00	551	<1	2
24	3P/3M	22	587	587	0.00	3959	<1	2
25	3P/3M	22	306	306	0.00	724	<1	2
26	3P/3M	22	357	357	0.00	381	<1	1&2
27	3P/3M	22	311	318	2.25	264	<1	1&2
28	3P/3M	22	440	440	0.00	1213	<1	1&2
29	3P/3M	22	754	819	8.62	1986	<1	1
30	3P/3M	22	533	533	0.00	374	<1	1
31	3P/3M	22	419	419	0.00	965	<1	1&2
32	3P/3M	22	397	406	2.27	412	<1	1
33	3P/3M	22	615	643	4.55	536	<1	1
34	3P/3M	22	865	1003	15.95	435	<1	1
35	3P/3M	22	648	707	9.10	329	<1	1
36	3P/3M	22	766	766	0.00	1322	<1	2
37	3P/3M	22	807	820	1.61	2344	<1	2
38	3P/3M	22	744	756	1.61	368	<1	2
39	3P/3M	22	1320	1320	0.00	938	<1	2
40	3P/3M	22	1250	1310	4.80	900	<1	2
41	3P/3M	22	1396	1483	6.23	610	<1	2
42	4P/4M	27	595	673	13.11	2069	<1	2
43	4P/4M	27	542	549	1.29	673	<1	2
44	4P/4M	27	995	1120	12.56	5844	<1	2
45	4P/4M	27	1112	1192	7.19	2875	<1	2
46	4P/4M	27	797	813	2.01	1922	<1	2
47	4P/4M	30	707	712	0.71	5281	<1	1
48	4P/4M	30	918	947	3.16	5204	<1	1
49	4P/4M	30	1097	1214	10.67	4891	<1	2
50	4P/4M	30	648	672	3.70	5029	<1	2
	Average of %difference =				2.53			
	Standard deviation =				4.12			

[a] Tardiness type 1: To obtain feasible solution, some products must be tardy.
[a] Tardiness type 2: Feasible solution can be obtained without any tardiness, but forcing some products to be tardy reduces the total cost

Of the 50 test problems examined (see Table 5.14), algorithm VI found the optimal solutions for 27 problems. The largest deviation from optimum is about *15.95%*. In all test problems, algorithm VI obtained an average deviation solutions of *2.53%* from the optimum and with a standard deviation of *4.12*. The computational requirements for solving the problems by algorithm VI are less than 1 second in all problems tested. These times are much less than the computational times of the optimal solution procedure.

CHAPTER 6
SUMMARY AND CONCLUSION

In this research, the following four scheduling problems have been studied: (1) single machine problem with earliness cost minimization, (2) single machine problem with the sum of the weighted earliness and weighted tardiness cost minimization, (3) assembly job shop problem with earliness cost minimization, and (4) assembly job shop problem with the sum of weighted earliness and weighted tardiness cost minimization. Four mathematical models based on these four scheduling problems were developed in an effort to obtain optimal solutions. Six heuristic algorithms were developed to solve the problems. Algorithms I and II were developed to solve the single machine problem with earliness cost minimization. Algorithms III and IV were developed to solve the single machine problem with the sum of the weighted earliness and weighted tardiness cost minimization. Algorithm V was developed to solve the assembly job shop problem with earliness cost minimization and Algorithm VI was developed to solve the assembly job shop problem with the sum of weighted earliness and weighted tardiness cost minimization. The performances of the heuristic algorithms were demonstrated on some sample test problems. Quality of solutions and CPU time of solutions were the performance measures of interest.

6.1 Summary of the research

We have identified several properties of optimal solutions for the single machine scheduling problem with the objective of minimizing the weighted earliness penalty. Algorithm I was developed based on these properties while algorithm II is based on the tabu search concept with short term memory search. Both algorithms I and II were applied to 75 test problems. The problem sizes are varied from 10 to 30 jobs. The results from both algorithms were also compared with the optimal solutions for the problem cases involving 10 jobs and 15 jobs. The results from both algorithms I and II indicate that these two algorithms are able to produce solutions close to optimal in small size problems. For the large problems, the quality of the solutions from algorithm I based on optimality conditions are relatively better than those obtained by algorithm II based on tabu search concept by approximately 1.09%. The computational time to solve the problems by these two heuristic algorithms is less than 2 second in all cases.

Algorithms III and IV are respectively the extension of heuristic algorithms I and II. Algorithm III is a combination of the features of algorithm I and the pairwise interchange method while algorithm IV is based on the tabu search concept. The only difference between algorithms II and IV is that job tardiness is allowed in algorithm IV. Algorithms III and IV were applied to 75 test problems of single machine problem with the sum of the weighted earliness and weighted tardiness cost minimization. The results from these two algorithms were compared with the optimal solutions for the problem cases involving 10 and 15 jobs. The solutions from both algorithms were also compared with the Ow & Morton [31] dispatching algorithm. In Ow & Morton [31] algorithm, inserted idle time is not allowed on the machine and this is different from that of algorithms III and IV. Thus, algorithms III and IV can be compared to the Ow & Morton algorithm to some degrees. For small size problems, the results indicate that algorithm III obtained an average deviation solutions of 1.38% from optimal while algorithm IV based on tabu search obtained an average deviation of 1.18% from the optimal. For all problems tested, the results show that algorithm IV, the tabu search, is the best algorithm for solving the earliness/tardiness problems compared to algorithm III and the Ow & Morton algorithm. It outperforms algorithm III and Ow & Morton [31] dispatch algorithm by about 1.55% and 25.48%, respectively, on average. The computational time to solve the problems by these two heuristic algorithms is less than 2 seconds in all cases.

Algorithm V is extended from algorithm I. It is applied to solve multiple machine problems with earliness cost minimization. In algorithm V, a multiple machine problem is decomposed into a set of single machine problems. Each decomposed single machine problem is solved by algorithm I. Decomposed single machine problems are related to one another by the precedence relationships between operations in the product structures. Algorithm V was applied to 50 test problems. The test problems are varied from 2-product and 3-machine (2P / 3M) problems to 5-product and 5 -machine (5P / 5M) problems. Each product has its own product structure. The number of operations is varied from 12 to 42 operations. The solutions from algorithm V were compared to the optimal solutions. The results show that the largest deviation is about *7.74%* from the optimal. This deviation was registered for a 4-product and 4-machine problem. But in all tested problems, algorithm V obtained an average deviation solutions of *0.67%* from the optimal. The computational requirements for solving the problems are less than 1 second in all tested problems. They are much less than the computational times of the optimal solution procedure.

Algorithm VI is the extension of algorithm V. It is a combination of the features of algorithm V and a method that can identify the appropriate amount of tardiness allocation for each product. Algorithm VI was applied to 50 test problems consisting of multiple machines and multiple jobs based on the minimization of the sum of weighted earliness and weighted tardiness cost. The test problems varied from 2-product and 3-machine (2P / 3M) problems to 4-product and 4-machine (4P / 4M) problems. Each product has its own product structure. The number of operations involved varied from 12 to 30 operations. The solutions from algorithm VI were compared to the optimal solutions. The results show that the largest deviation is about *15.95%* from the optimal and was obtained in a 3-product and 3-machine problem. But in all problems tested, algorithm VI obtained an average deviation solutions of *2.53%* from the optimal. The computational requirements for solving the problems are less than 1 second in all test problems. They are much less than the computational times of the optimal solution procedure.

6.2 Conclusion

In this research, a lack of heuristic algorithms in open literature for scheduling jobs of practical sizes in assembly job shop with the sum of weighted earliness and weighted tardiness penalties prompted the developments of six heuristic algorithms. The development of the heuristic algorithms starts with the development of heuristics for the single machine problem with earliness cost minimization (i.e., Alg I and Alg II) and single machine problem with the sum of weighted earliness and weighted tardiness costs minimization (i.e, Alg. III and Alg. IV). Finally, the heuristics of single machine problems were extended to solve the assembly job shop problem with the earliness cost minimization (i.e., Alg. V) and assembly job shop problem with the sum of weighted earliness and weighted tardiness penalties (i.e., Alg. VI).

The effectiveness of the heuristic algorithms for single machine problem (Alg. I, II, III, and IV) was demonstrated by the quality of solutions they produced on test problems. For problems of sizes 10-15 jobs, the solutions obtained were on average within 4% of optimal solution. It was also shown that the heuristics can solve large problems within very short computation times (i.e., less than 2 seconds in all cases). For large size problems, the optimal solutions could not be obtained to assess adequately their effectiveness. Algorithms I and II were compared with each other, while algorithm III was compared to algorithm IV. It was found that algorithm I was relatively better than algorithm II, and algorithm IV was relatively better than algorithm III.

In the case of assembly job shop problem, the effectiveness of algorithm V and VI was demonstrated by the fact that they produced on average solution within 1% and 3% of the optimum solution respectively. It was also shown that the heuristics can solve large problems within very short computational times (i.e., less than 2 seconds in all cases).

One of the most important aspects of the developed heuristic algorithms is that they are general enough to be used in any environment where job scheduling is required. Algorithm V and VI can also be applied to traditional job shop problems without product assembly considerations. These methodologies can be easily implemented. They provide a very systematic way for scheduling. They can generate good solutions within a reasonable time.

6.3 Research contributions

The contributions of this research in the area of scheduling with cost consideration are significant. It introduces six effective heuristic algorithms for scheduling problem with cost consideration. The first two heuristics deal with single machine with earliness penalty minimization. The third and fourth heuristics were developed to solve single machine problem with the sum of weighted earliness and tardiness penalties minimization. These four algorithms are very easy to apply in any environment. The first and third heuristics also contain one significant benefit. They do not need any prespecified parameters, which are always required in many algorithms in literature such as in Ow and Morton dispatching algorithm [31].

The contributions of the last two heuristics (i.e., algorithm V and VI) are very significant in the area of scheduling, since they deal with cost minimization in an assembly job shop. As previously mentioned in section 1.2, there have been very few reported research that focused on assembly job shop. With a few published papers on assembly job shop, most of them deal with such regular measures as mean flow time and completion time. To our knowledge, no published paper deals exactly with the minimization of the weighted earliness and weighted tardiness penalties in assembly job shop problem. It can be claimed that both algorithm V and VI are the very first algorithms dealing with assembly job shop with earliness and tardiness cost consideration. Both algorithm V and VI are also proofed to be effective heuristics and general enough to apply in industries.

These six developed heuristics are useful in real world industries, since they deal exactly with the earliness and tardiness cost. The earliness and tardiness criterion is considered as one of important measure in Just in time (JIT) production system, which are widely applied in many industries. As mentioned earlier, the scheduling problem dealing with earliness and tardiness criterion is an NP-complete problem even in the single machine case [6]. The optimal solution is prohibited to obtain in large size problems, which are normal problem size in industries. But the developed heuristics in this research can be applied to solve even large size problem in a reasonable computational time and the performance of these six heuristics are good compared to optimal solution in small size problems.

6.4 Possible extension

The heuristic algorithms modeled in this research allowed idle time in the generated schedule. The cost of machine staying idle was not considered in this research. This condition may not be true in industries where machine cost is extremely high and the machine idle time is prohibited. Generating the best schedule without idle time is different from the schedule which is generated by the heuristic algorithms developed in this research. Further research might address this problem by considering the cost of machine staying idle in additional to earliness and tardiness costs.

The case of having identical sub-assemblies in different products and consolidating these subassemblies together for scheduling was not considered in this study. In this study, each sub-assembly was considered as a unique product. If the problem involves setup cost, the consideration of scheduling identical sub-assemblies as a large batch might be necessary. Scheduling identical sub-assemblies in large batch reduces setup cost and this in turn can reduce the overall costs. The developed heuristics can be further improved by considering this condition.

Finally, in this study, machine break down was not considered. Work delay based on machine failures is a normal situation in industry. The consideration of machine failure would increase the complexity in scheduling. To implement this extension, stochastic modeling may be required for developing the heuristic procedures.

APPENDIX A

PROOF OF THE PROPOSITIONS

In this section, propositions presented in section 3.1.2 are proved. These propositions are developed for finding the optimal ordering of any two jobs for minimizing weighted earliness penalty. We consider the cases where two jobs are not possible to complete at their due-dates due to conflict (see Fig A.1). From Fig. A.1, if we would like to obtain optimal non-conflict ordering of two jobs, at least, a job must be leftward shifted. Rightward shift can not be employed, since it causes tardiness of jobs, a violation of constraints. There is no exact rule to select jobs to be leftward shifted jobs. It depends on the conditions and parameters of the jobs. Intuitively, we would like to schedule a job with the largest earliness penalty (E_i) closer to its due-date. However, such simple rule may not always guarantee optimal ordering. Jobs with small earliness penalty and small processing time (t_i) may have higher priority to schedule closer to their due-dates than jobs with high earliness penalty and high processing time. Thus, the ratio of earliness penalty and processing time (E_i / t_i) may be more suitable for scheduling consideration than simply using only the earliness penalty. Based on the *weighted longest processing time rule*, *WLPT* in [31], jobs with larger Y_i (where $Y_i = E_i / t_i$) must schedule closer to their due-dates than jobs with smaller Y_i value. In *WLPT* rule [31], if the *WLPT* sequence (i.e. $Y_1 \leq Y_2 \leq Y_3 \leq \ldots\ldots\ldots \leq Y_n$, where Y_n is the nth job on the sequence) results in a schedule that does not have any tardy jobs, then this sequence is optimal. The *WLPT* rule can not be directly applied to the work reported in this research, since the *WLPT* sequence may not yield a schedule without any tardy jobs. For example, job 1 has 5 units of processing time, due-date at 5, and 10 unit cost/unit time for earliness penalty. Job 2 has 5 units of processing times, due-date at 10, and 5 unit cost/unit time for earliness penalty. If the *WLPT* rule is employed, the schedule is $2 \rightarrow 1$. These sequence causes job 1 to be tardy and this is unacceptable in this research. To deal with this problem, five propositions are explored for optimal ordering between two conflict jobs based on E_i / t_i, due-dates of

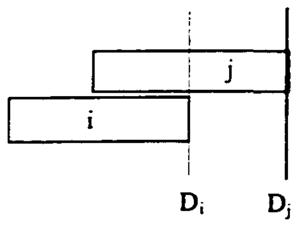


Figure A.1 Jobs *i* and *j* overlap each other, if each completes on its due date

both jobs, and some other conditions. These five propositions are mathematically derived as discussed below.

Proposition 1. For the case where jobs *i* and *j* are not possible to complete exactly on their due-dates due to conflict, if $\frac{E_i}{t_i} \leq \frac{E_j}{t_j}$ and $D_i < D_j$, then the optimal non-conflict ordering between jobs *i* and *j* is that job *i* precedes job *j* ($i \rightarrow j$) as shown in Fig. A.2.

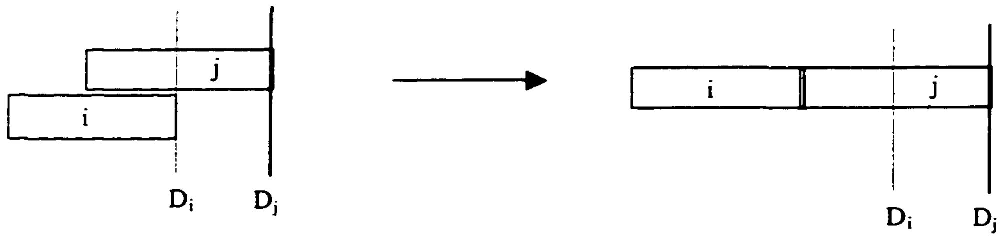


Figure A.2 Illustration of the proposition 1

Proof. Suppose that the cost of $j \rightarrow i$ (Fig. A.3a) is less than $i \rightarrow j$ (Fig. A.3b).

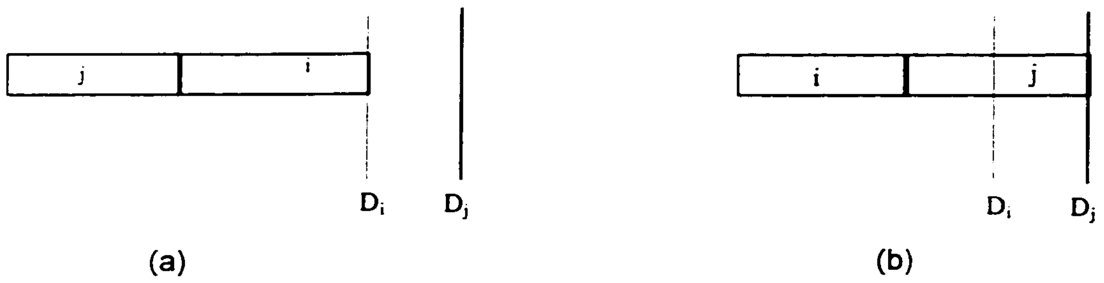


Figure A.3 Proof of proposition 1

$j \rightarrow i$		$i \rightarrow j$
$(D_j - (D_i - t_i))E_j$	$<$	$(D_i - (D_j - t_j))E_i$
$(D_j - D_i + t_i)E_j$	$<$	$(D_i - D_j + t_j)E_i$
$(D_j - D_i)E_j + t_i E_j$	$<$	$(D_i - D_j)E_i + t_j E_i$

Since $(D_j - D_i)$ is positive and $(D_i - D_j)$ is negative, and $t_iE_j > t_j E_i$, then the relation $(D_j - D_i)E_j + t_iE_j < (D_i - D_j)E_i + t_jE_i$ is a contradiction. Therefore, proposition 1 is true.

Proposition 2. For the case where $\frac{E_i}{t_i} > \frac{E_j}{t_j}$, $D_i < D_j$, and both jobs i and j are not possible to complete exactly on their due-dates due to conflict, if $(D_i - D_j) \leq \frac{t_iE_j - t_jE_i}{(E_i + E_j)}$, then the optimal non-conflict ordering between jobs i and j is that job i precedes job j $(i \rightarrow j)$, otherwise $j \rightarrow i$ (see Fig A.4).

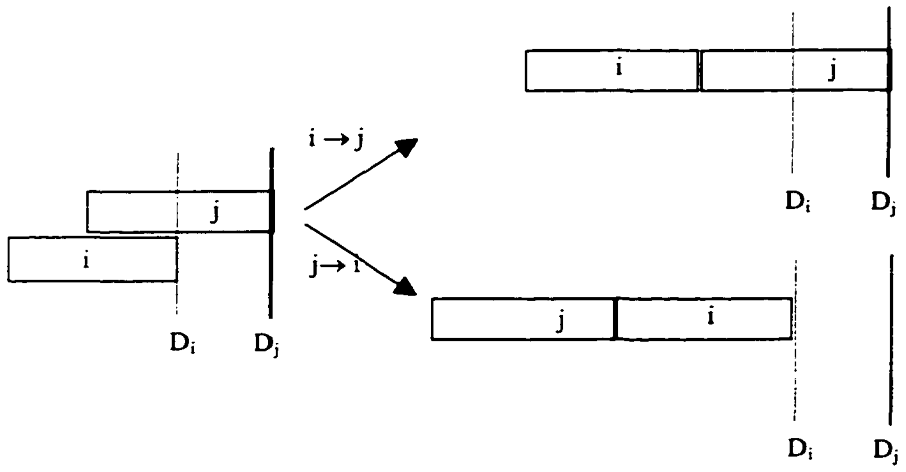


Figure A.4 Illustration of the proposition 2

Proof. For the case that cost of $i \to j$ (Fig. A.5(a)) is less than $j \to i$ (Fig. A.5(b)),

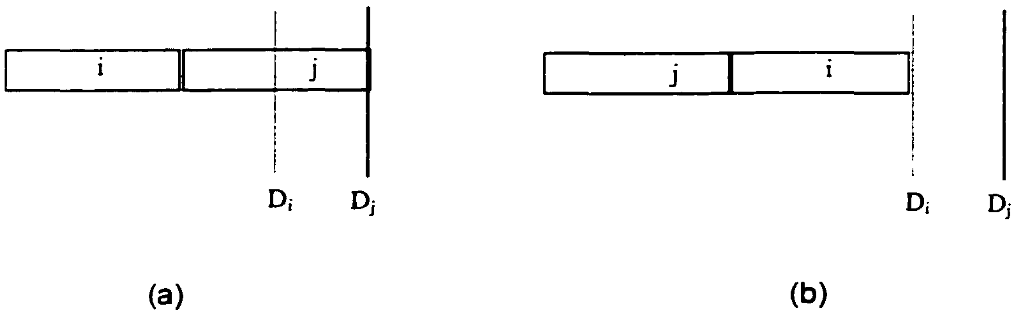


Figure A.5 Proof of proposition 2.

$i \to j$		$j \to i$
$(D_i - (D_j - t_j))E_i$	$\le$	$(D_j - (D_i - t_i))E_j$
$(D_i - D_j + t_j)E_i$	$\le$	$(D_j - D_i + t_i)E_j$
$(D_i - D_j)E_i + t_j E_i$	$\le$	$(D_j - D_i)E_j + t_i E_j$
$(D_i - D_j)E_i + (D_i - D_j)E_j$	$\le$	$t_i E_j - t_j E_i$
$(D_i - D_j)$	$\le$	$\dfrac{t_i E_j - t_j E_i}{(E_i + E_j)}$

On the other hand, if $(D_i - D_j) > (t_i E_j - t_j E_i) / (E_i + E_j)$, then $j \rightarrow i$. The proposition 2 is proved.

Proposition 3. For the case where jobs *i* and *j* are not possible to complete exactly on their due-dates due to conflicts between jobs *i*, *j* and *k* where *k* is already scheduled, If $\frac{E_i}{t_i} \le \frac{E_j}{t_j}$, then the optimal non-conflict ordering between jobs *i* and *j* is that job *i* precedes job *j*.

For example, in Fig. A.6, suppose a third job (i.e. job *k*) is already scheduled for processing between the time period from *B* to *B** and the due-dates of jobs *i* and *j* fall within this time period (i.e. $B \le D_i$, $D_j \le B^*$). Thus jobs *i* and *j* can not be processed between *B* to *B**. If $\frac{E_i}{t_i} \le \frac{E_j}{t_j}$, then the optimal non-conflict ordering is that $i \rightarrow j$ and the completion time of job *j* is at time *B*.

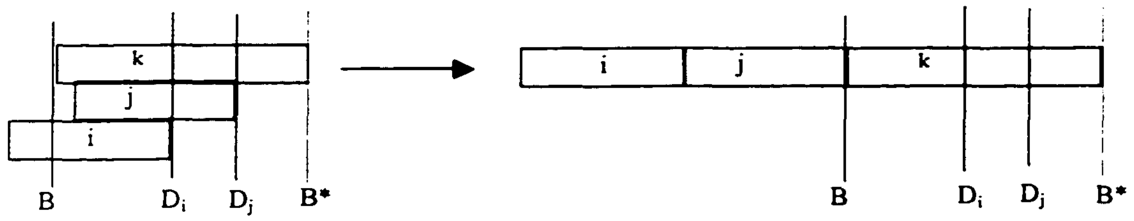


Figure A.6 Illustration of the proposition 3

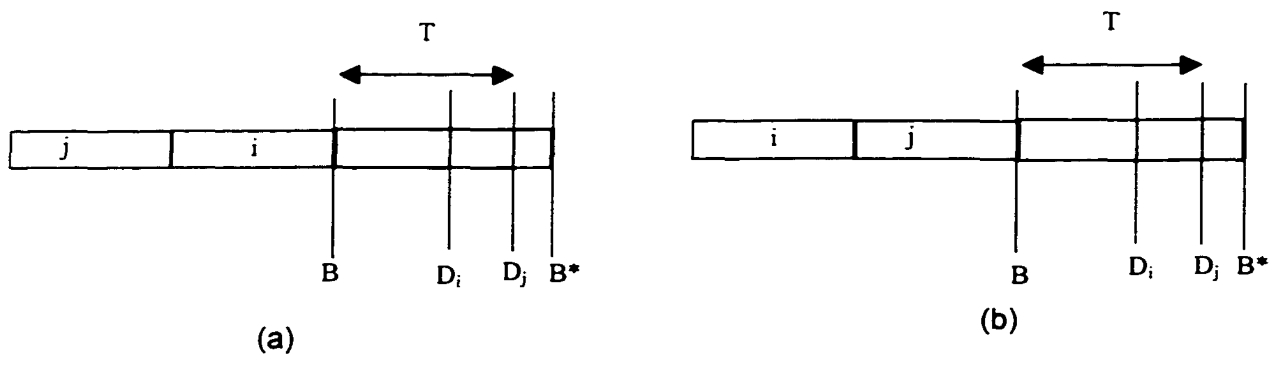


Figure A.7 Proof of proposition 3.

Proof. Suppose that the cost of $j \to i$ (Fig.A7(a)) is less than $i \to j$ (Fig.A7(b))

$$\begin{array}{lcl} j \to i & & i \to j \\ (D_i - (D_j - T))E_i + (D_j - (D_j - T - t_i))E_j & < & (D_i - (D_j - T - t_j))E_i + (D_j - (D_j - T))E_j \\ (D_i - D_j)E_i + TE_i + TE_j + t_i E_j & < & (D_i - D_j)E_i + TE_i + t_j E_i + TE_j \\ t_i E_j & < & t_j E_i \\ (E_j / t_j) & < & (E_i / t_i) \end{array}$$

Since the relation contradicts the stated condition $\frac{E_i}{t_i} \le \frac{E_j}{t_j}$, so proposition 3 is proved.

Proposition 4. For the case where job *k* is already scheduled, and job *j* is not possible to complete exactly on its due-date due to conflicts between jobs *j* and *i*, and jobs *j* and *k*, and job *i* is not possible to complete exactly on its due-date due to conflict between jobs *i* and *j* (see Fig. A.8(a)), If $\frac{E_i}{t_i} \le \frac{E_j}{t_j}$, then the optimal non-conflict ordering between jobs *i* and *j* is that job *i* precedes job *j* *(i → j)*.

For example, in Fig A.8, suppose a third job (i.e. job *k*) is already scheduled for processing between the time period from *B* to *B** and the due-date of job *j* falls within this time period (i.e. $B \le D_j \le B^*$). Thus job *j* can not be processed from time *B* to D_j.

Similarly, job *i* has conflict with job *j*, but not with job *k*. If $\frac{E_i}{t_i} \le \frac{E_j}{t_j}$, then the optimal non-conflict ordering is that $i \rightarrow j$ and the completion time of job *j* is at time *B*.

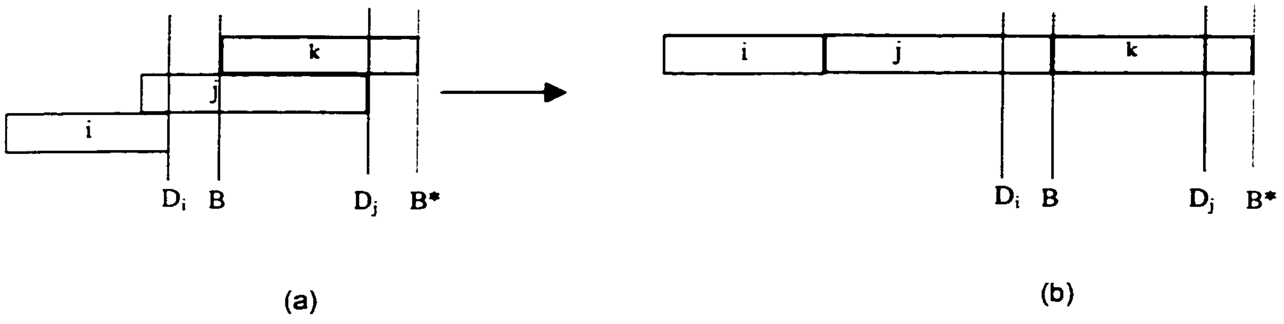


Figure A.8 Illustration of the proposition 4

Proof same as proposition 3.

Proposition 5. For the case where job *k* is already scheduled, and job *j* is not possible to complete exactly on its due-date due to conflicts between jobs *j* and *i*, and jobs *j* and *k*, and job *i* is not possible to complete exactly on its due-date due to conflict between jobs *i* and *j* (see Fig. A.9 (a)), if $\frac{E_i}{t_i} > \frac{E_j}{t_j}$ and $(D_i - D_j + T) \le \frac{t_iE_j - t_jE_i}{(E_i + E_j)}$ where *T* is the lenght of time that job *j* can not be processed until its due-date due to the conflict between jobs *j* and *k*, then the optimal non-conflict ordering between jobs *i* and *j* is that job *i* precedes job *j* ($i \rightarrow j$). On the other hand, if $\frac{E_i}{t_i} > \frac{E_j}{t_j}$ and $(D_i - D_j + T) > \frac{t_iE_j - t_jE_i}{(E_i + E_j)}$, then the optimal non-conflict ordering between jobs *i* and *j* is that job *j* precedes job *i* ($j \to i$). This proposition can be shown as in Fig A.9.

In Figure A.9, suppose that job *k* is already scheduled for processing between the time period from *B* to *B** and the due-date of job *j* is in this time period (i.e. $B \le D_j \le B^*$). Thus job *j* can not be processed from *B* to D_j. Similarly, job *i* has conflict with job *j*, but not with job *k*. In this case, $T = D_j - B$. If $\frac{E_i}{t_i} > \frac{E_j}{t_j}$ and $(D_i - D_j + T) \le \frac{t_i E_j - t_j E_i}{(E_i + E_j)}$, then $i \to j$ as in Fig. A.9(b). If $\frac{E_i}{t_i} > \frac{E_j}{t_j}$ and $(D_i - D_j + T) > \frac{t_i E_j - t_j E_i}{(E_i + E_j)}$, then $j \to i$ as in Fig. A.9(c).

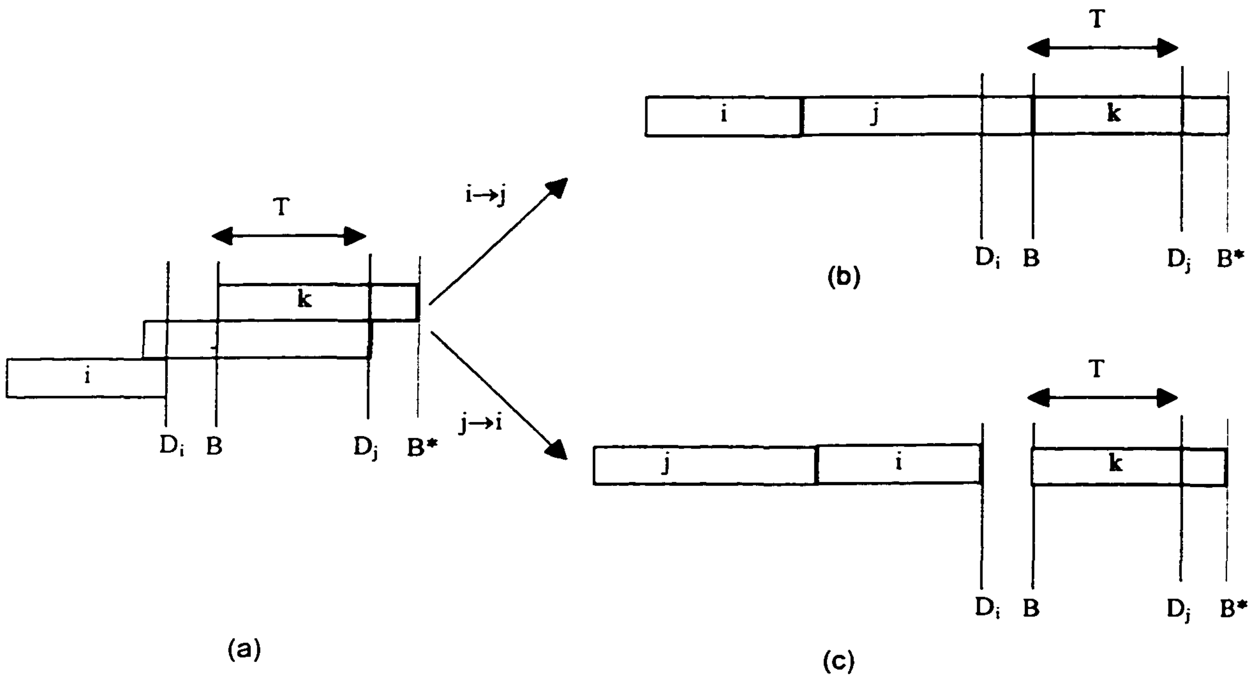


Figure A.9 Illustration of the proposition 5

Proof. For the case that cost of $i \to j$ (Fig. A.9(b)) is less than $j \to i$(Fig. A.9(c)),

$$\begin{array}{lcl}
i \to j & & j \to i \\
(D_j - (D_j - T))E_j + (D_i - (D_j - T - t_j)E_i & \leq & (D_j - (D_i - t_i))\,E_j \\
TE_j + (D_i - D_j)E_i + TE_i + t_j E_i & \leq & (D_j - D_i)E_j + t_i E_j \\
(E_j + E_i)T + (D_i - D_j)E_i - (D_j - D_i)E_j & \leq & t_i E_j - t_j E_i \\
(E_j + E_i)T + (D_i - D_j)(E_j + E_i) & \leq & t_i E_j - t_j E_i \\
(D_i - D_j + T) & \leq & \dfrac{t_i E_j - t_j E_i}{(E_i + E_j)}
\end{array}$$

This proves the proposition. On the other hand, if $(D_i - D_j + T) >$ $(t_i E_j - t_j E_i) / (E_i + E_j)$, then $j \to i$. The proposition 5 is proved.

APPENDIX B

THE OW & MORTON ALGORITHM

This algorithm was presented by Ow and Morton [31] in 1989. In this algorithm, the priorities of unscheduled jobs are determined when the machine becomes available. The highest priority job is selected to schedule next. The priority rule is based on the slack time of unscheduled jobs at the moment that the machine becomes available, and the value of a parameter *k*. The value of parameter *k* is assigned by the scheduler. It is an average number of jobs that the scheduler would like to see when a sequence decision is to be made. The steps of the algorithms are as presented below.

Algorithm

Step 1. Set $ETIME = 0$, $\pi = \varnothing$, $\sigma = J$, and set k (parameter).

Step 2. For all $i \in \sigma$, calculate the priority of job i, $P_i(s_i)$, at time $ETIME$:

$$P_i(s_i) = \begin{cases} W_i & \text{if } s_i < 0 \\ W_i \exp\left(\frac{E_i + W_i}{E_i}(s_i/\bar{t})\right) & \text{if } 0 \le s_i \le \left(\frac{W_i}{E_i + W_i}\right)k\bar{t} \\ E_i^{-2}\left(W_i - \frac{(E_i + W_i)}{k\bar{t}}s_i\right)^3 & \text{if } \left(\frac{W_i}{E_i + W_i}\right)k\bar{t} \le s_i \le k\bar{t} \\ -E_i & \text{otherwise} \end{cases}$$

Step 3. Schedule the highest priority job i;

Starting time of job i = $ETIME$;

Completion time of job i = starting time of job i + t_i.

Step 4. Set $ETIME$ = completion time of job i, $\pi \leftarrow \pi + \{i\}$, $\sigma = \sigma - \{i\}$.

Step 5. If $|\sigma| = 0$, stop, otherwise, go to step 2.

Where W_i = the tardy cost rate (costs/unit time) of job i,

E_i = the early cost rate (costs/unit time) of job i,

$\bar{t}$ = the average processing time,

N = the number of jobs,

k = the selected parameter ($1 \le k \le N$),

D_i = due-date of job i,

t_i = processing time of job i,

$ETIME$ = the earliest available time of the machine,

s_i = the slack time of job i at time $ETIME$ $(s_i = D_i - ETIME - t_i)$,

$P_i(s_i)$ = the priority of job i with slack time s_i.

π = the set of secheduled jobs,

σ = the set of unscheduled jobs,

J = the set of all jobs.

APPENDIX C

ALGORITHM ILLUSTRATION

C.1 Algorithm I illustration

To illustrate the steps of the algorithm I, consider example 3.1 in Chapter 3 , whose parameters are described as in Table C.1. The steps of employing algorithm I to example 3.1 is illustrated as follows:

Table C.1 Parameters of example 3.1

Job	1	2	3	4	5	6	7	8
t_i	3	4	9	10	10	5	7	2
D_i	80	80	75	66	64	60	50	43
E_i	6	4	9	2	3	7	5	1
Y_i	2	1	1	0.2	0.3	1.4	0.71	0.5

Step 0. Initialization

0a. Set $k=0$, $\sigma = \{1,2,3,4,5,6,7,8\}$, $TIME = 80$.

0b. For each $i \in \sigma$, set $\delta_i = \phi$, $Y_i = \frac{E_i}{t_i}$.

Step 1. Construct an ideal solution

1a. $C_1 = 80$, $S_1 = 77$, $C_2 = 80$, $S_2 = 76$, $C_3 = 75$, $S_3 = 66$, $C_4 = 66$, $S_4 = 56$, $C_5 = 64$, $S_5 = 54$, $C_6 = 60$, $S_6 = 55$, $C_7 = 50$, $S_7 = 43$, $C_8 = 43$, $S_8 = 41$ (see Fig C.1).

1b. There are conflicts between jobs. Go to step 2.

Step 2. Select $i^* = 1$ $(D_1 = TIME)$.

Step 3. Set $C_1 = 80$, $S_1 = 77$, and $\delta_1 = \{2\}$ (since $D_2 > S_1$).

Step 4. Since $Y_1 > Y_2$, go to Step 7 .

Step 7a. No imbedded idle period.

Step 7b. $TIME = 77$. Schedule job 1 at $C_1 = 80$, $S_1 = 77$ (see Fig C.2).

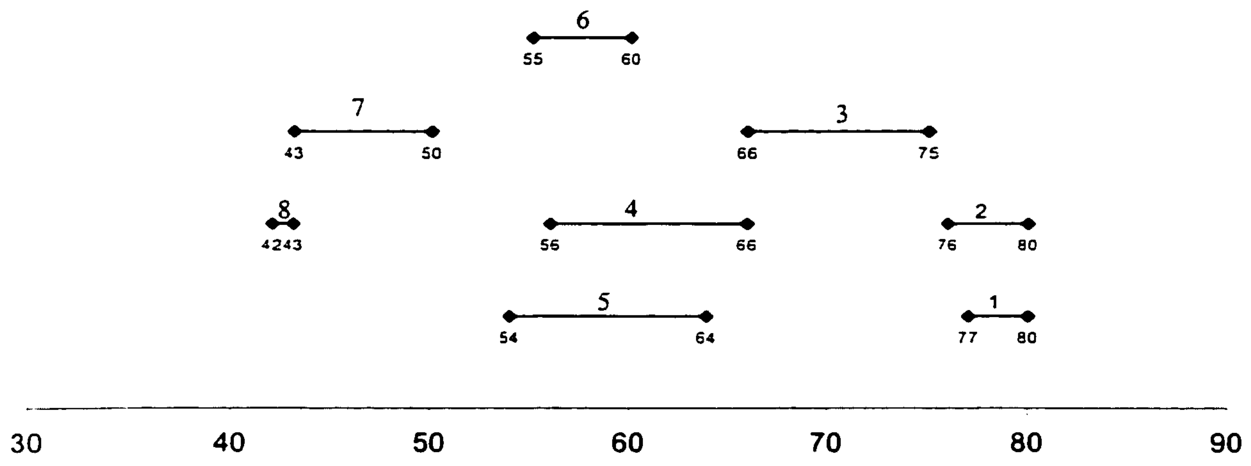


Figure C.1 Ideal solution for example 3.1

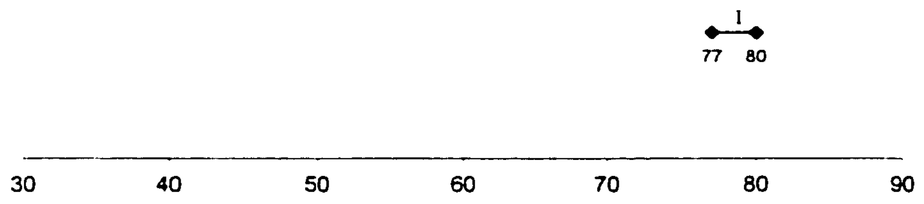


Figure C.2 Scheduled job 1 of example 3.1

Step 7c. $\sigma = \{2,3,4,5,6,7,8\}$, $\pi = \{1\}$

Step 7d. $|\sigma| \neq 0$ go to step 2.

Step 2. Select $i^* = 2$ $(D_2 > TIME)$.

Step 3. Set $C_2 = 77$, $S_2 = 73$, and $\delta_2 = \{3\}$

Step 4. Since $Y_2 = Y_3$, go to Step 7 .

Step 7a. No imbedded idle period.

Step 7b. $TIME = 73$. Schedule job i^* at $C_2 = 77$, $S_2 = 73$ (see Fig. C.3).

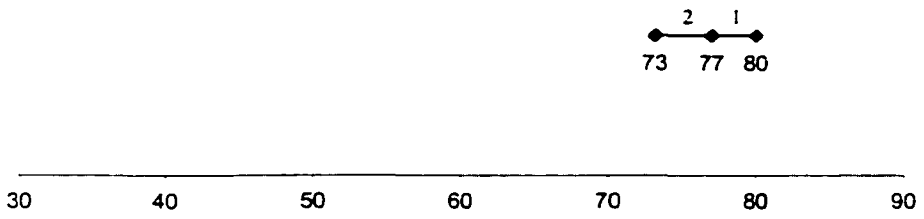


Figure C.3 Schedule job 2 of example 3.1

Step 7c. $\sigma = \{3,4,5,6,7,8\}$, $\pi = \{1,2\}$

Step 7d. $|\sigma| \neq 0$ go to step 2.

Step 2. Select $i^* = 3$ ($D_3 >$ TIME).

Step 3. Set $C_3 = 73$, $S_3 = 64$, and $\delta_3 = \{4\}$

Step 4. Since $Y_3 > Y_4$, go to Step 7 .

Step 7a. No imbedded idle period.

Step 7b. $TIME = 64$. Schedule job 3 at $C_3 = 73$, $S_3 = 64$ (see Fig. C.4).

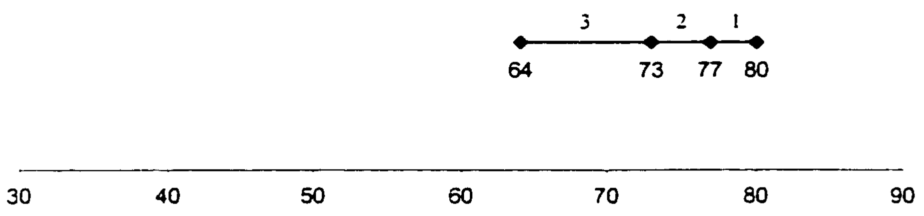


Figure C.4 Schedule job 3 of example 3.1

Step 7c. $\sigma = \{4,5,6,7,8\}$, $\pi = \{1,2,3\}$

Step 7d. $|\sigma| \neq 0$ go to step 2.

Step 2. Select $i^* = 5$ ($D_5 = TIME$, and $Y_5 > Y_4$).

Step 3. Set $C_5 = 64$, $S_5 = 54$, and $\delta_5 = \{4,6\}$

Step 4. Since $Y_6 > Y_5$, $Y_6 > Y_4$ go to Step 5.

Step 5. Select $j^* = 6$.

Step 6a. Calculate $T_5 = 0$.

Step 6b. Since $(D_6 - D_5 + T_5) = -4 > \dfrac{t_6E_5 - t_5E_6}{(E_5 + E_6)} = -5.5$, then job 6 is selected to be job i^* instead of job 5. Set C_5 back to be its due-date ($C_5 = 64$, $S_5 = 54$). Set $i^* = 6$, and go to step 3.

Step 3. Set $C_6 = 60$, $S_6 = 55$, and $\delta_6 = \{4,5\}$

Step 4. Since $Y_6 > Y_5$ and $Y_6 > Y_4$ go to Step 7 .

Step 7a.There is imbedded idle period. Set $k = 1$, $\Delta_1 = (60, 64)$.

Step 7b. $TIME = 55$. Schedule job 6 at $C_6 = 60$, $S_6 = 55$ (see Fig. C.5).

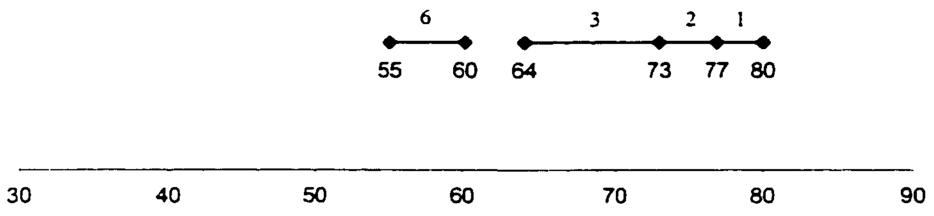


Figure C.5 Schedule job 6 of example 3.1

Step 7c. $\sigma = \{4,5,7,8\}$, $\pi = \{1,2,3,6\}$

Step 7d. $|\sigma| \neq 0$ go to step 2.

This procedure is repeated for the rest of the jobs. Based on the algorithm, jobs *5, 7, 8, 4* are scheduled respectively as shown in Fig. C.6.

After job 4 is scheduled, the problem contains $|\sigma| = 0$. Then, the algorithm moves to step 8 for calculating the total weighted earliness cost.

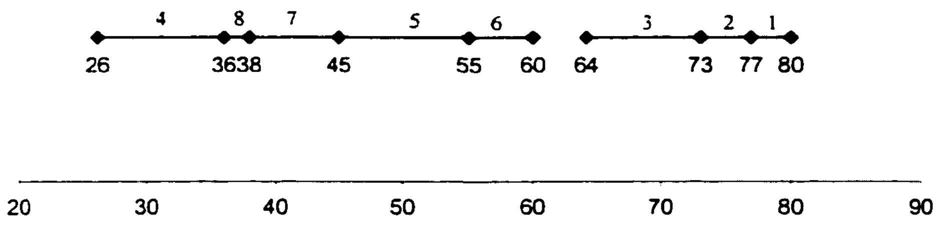


Figure C.6 Job schedule for example 3.1

Step 8. Calculate total weighted earliness cost , $Z = 147$ unit cost.

Step 9. Keep the solution from step 8 as the current best solution by setting

$S'_1 = 77$, $C'_1 = 80$, $S'_2 = 73$, $C'_2 = 77$, $S'_3 = 64$, $C'_3 = 73$, $S'_4 = 26$, $C'_4 = 36$, $S'_5 = 45$, $C'_5 = 55$, $S'_6 = 55$, $C'_6 = 60$, $S'_7 = 38$, $C'_7 = 45$, $S'_8 = 36$, $C'_8 = 38$, $K' = 1$, $\Delta'_1 = (60, 64)$, $Z' = 147$.

Step 10. There is an imbedded idle time, $\Delta'_1 = (60, 64)$, then go to step 11.

Step 11a. Set β_1 as the set of jobs that are scheduled before Δ'_1 and are able to fill in Δ'_1 , $\beta_1 = \{5\}$.

Step 11b. Assign job 5 to fill $\Delta'_1 = (60, 64)$. $C_5 = 64$, $S_5 = 54$. Since $S_5 < 60$, thus jobs 6,7,8 and 4 must be leftward shifted.

Step 11c. Set jobs 6,7,8 and 4 in an ideal form, set $\sigma = \{4,6,7,8\}$ and set *TIME* . $= S_5$ (i.e. 54). The schedule is as shown in Fig. C.7.

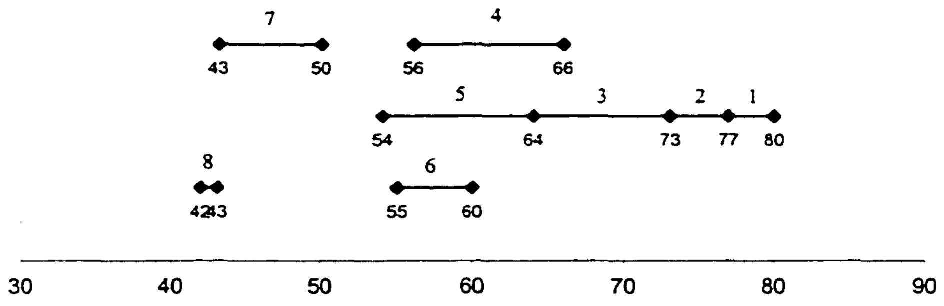


Figure C.7 Assign job 5 to fill in the imbedded idle time period $\Delta'_1 = (60, 64)$

Step 2. Select $i^* = 6$ ($D_6 >$ *TIME* and $Y_6 > Y_4$).

Step 3. Set $C_6 = 54$, $S_6 = 49$, and $\delta_6 = \{4, 7\}$

Step 4. Since $Y_6 > Y_4$, and $Y_6 > Y_7$, go to Step 7 .

Step 7a. No imbedded idle period.

Step 7b. *TIME* = *49*. Schedule job *6* at $C_6 = 54$, $S_6 = 49$ (see Fig. C.8).

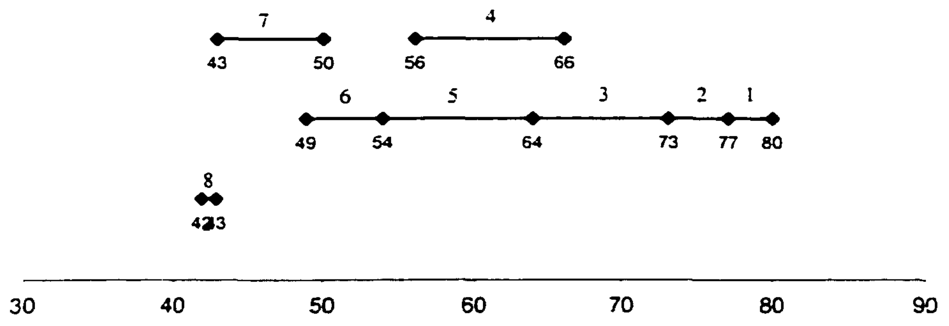


Figure C.8 Schedule job 6 of example 3.1 after assigning job 5 to fill Δ'_1 = *(60, 64)*

Step 7-C. $\sigma = \{4,7,8\}$, $\pi = \{1,2,3,5,6\}$

Step 7-D. $|\sigma| \neq 0$ go to step 2.

Step 2. Select $i^* = 7$ ($D_7 = TIME$, and $Y_7 > Y_4$).

Step 3. Set $C_7 = 49$, $S_7 = 42$, and $\delta_3 = \{8,4\}$

Step 4. Since $Y_7 > Y_4$, $Y_7 > Y_8$, go to Step 7 .

Step 7a. No imbedded idle period.

Step 7b. $TIME = 42$. Schedule job *7* at $C_7 = 49$, $S_7 = 42$

Step 7c. $\sigma = \{4,8\}$, $\pi = \{1,2,3,5,6,7\}$

Step 7d. $|\sigma| \neq 0$ go to step 2.

Step 2. Select $i^* = 8$ ($D_8 = TIME$, and $Y_8 > Y_4$).

Step 3. Set $C_8 = 42$, $S_8 = 40$, and $\delta_8 = \{4\}$

Step 4. Since $Y_8 > Y_4$, go to Step 7 .

Step 7a. No imbedded idle period.

Step 7b. $TIME = 40$. Schedule job *8* at $C_8 = 42$, $S_8 = 40$

Step 7c. $\sigma = \{4\}$, $\pi = \{1,2,3,5,6,7,8\}$

Step 7d. $|\sigma| \neq 0$ go to step 2.

Step 2. Select $i^* = 4$ ($D_4 > TIME$).

Step 3. Set $C_4 = 40$, $S_4 = 30$, and $\delta_4 = \phi$

Step 4. Go to Step 7 .

Step 7a. No imbedded idle period.

Step 7b. *TIME = 30*. Schedule job *4* at $C_4 = 40$, $S_4 = 30$ (see Fig. C.9).

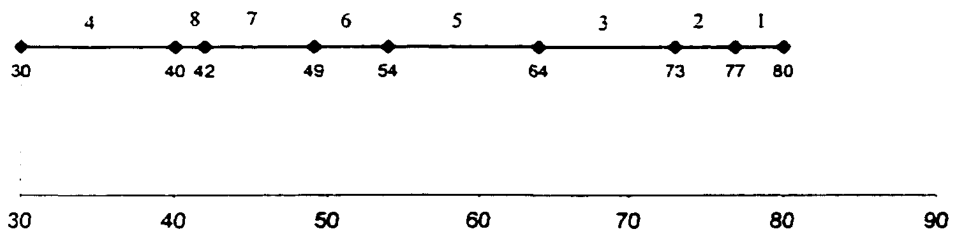


Figure C.9 The best job schedule obtained from Algorithm I

Step 7c. $\sigma = \phi$, $\pi = \{1,2,3,4,5,6,7,8\}$

Step 7d. $|\sigma| = 0$ go to step 8.

Step 8. Calculate total weighted earliness cost , $Z = 130$ unit cost. This is better than the $Z = 147$ previously obtained.

Step 9. Keep the solution from step 8 as the current best solution by setting $S'_1 = 77$, $C'_1 = 80$, $S'_2 = 73$, $C'_2 = 77$, $S'_3 = 64$, $C'_3 = 73$, $S'_4 = 30$, $C'_4 = 40$,$S'_5 = 54$, $C'_5 = 64$, $S'_6 = 49$, $C'_6 = 54$, $S'_7 = 42$, $C'_7 = 49$, $S'_8 = 40$, $C'_8 = 42$, $K' = 0$, $Z' = 130$.

Step 10. Since there is no imbedded idle period, go to step 14.

Step 14. Schedule all job $i \in J$ where $S_i = S_{i^*}$, and $C_i = C_{i^*}$., and stop.

C.2 MCE algorithm illustration

To illustrate the steps of the *MCE* algorithm (see pages 67-69), consider example 4.1 in Chapter 4 (page 61), whose product structures and parameters are described as in Figure 4.1(page 58) and Table 4.1(page 62), respectively. The ideal solution of example 4.1 is as shown in Figure 4.2 (page 63). Lets consider the latest conflict set consisting of operations *1011, 1111*, and *2011.* This conflict set can be eliminated by MCE algorithm as follows:

Step 0. Initialization

0a. Set $\sigma = \{O_{(1011)}, O_{(1111)}, O_{(2011)}\}$, $\pi = \phi$, $TIME = 500$.

0b. Set $\delta_{(1011)} = \phi$, $\delta_{(1111)} = \phi$, $\delta_{(2011)} = \phi$, $Y_{(1011)} = 0.7$, $Y_{(1111)} = 1$, $Y_{(2011)} = 0.36$.

Step1. $D_{(1011)} = 500$, $D_{(2011)} = 500$. $D_{(1111)} = 400$.

Step2. $P = \{O_{(2011)}, O_{(1011)}\}$. $O_{(ijkl)^*} = 1011$.

Step 3. Set $C_{1011} = 500$, $S_{1011} = 400$. The $O_{(2011)}$ conflicts to $O_{(1011)}$, then $\delta_{(1011)}=\{O_{(2011)}\}$.

Step 4. $Y_{2011} < Y_{1011}$, go to step 7.

Step 7. Schedule $O_{(1011)}$ on machine

7a. Set $C_{1011} = 500$ and $S_{1011} = 400$ (see Fig C.10), $TIME = 400$.

7b. $\sigma = \{O_{(2011)}, O_{(1111)}\}$ and $\pi = \{O_{(1011)}\}$.

7c. Since $|\sigma| \neq 0$, then go to step 2.

Step2. $P = \{O_{(2011)}, O_{(1111)}\}$. $O_{(ijkl)^*} = 1111$.

Step 3. Set $C_{(1111)} = 400$, $S_{(1111)} = 360$ .The $O_{(2011)}$ conflicts to $O_{(1111)}$, then $\delta_{(1111)}=\{O_{(2011)}\}$.

Step 4. $Y_{2011} < Y_{1011}$, go to step 7.

Step 7. Schedule $O_{(1111)}$ on machine

7a. Set $C_{1111} = 400$ and $S_{1111} = 360$ (see Fig C.11), $TIME = 360$.

7b. $\sigma = \{O_{(2011)}\}$ and $\pi = \{O_{(1011)}, O_{(1111)}\}$.

7c. Since $|\sigma| \neq 0$, then go to step 2.

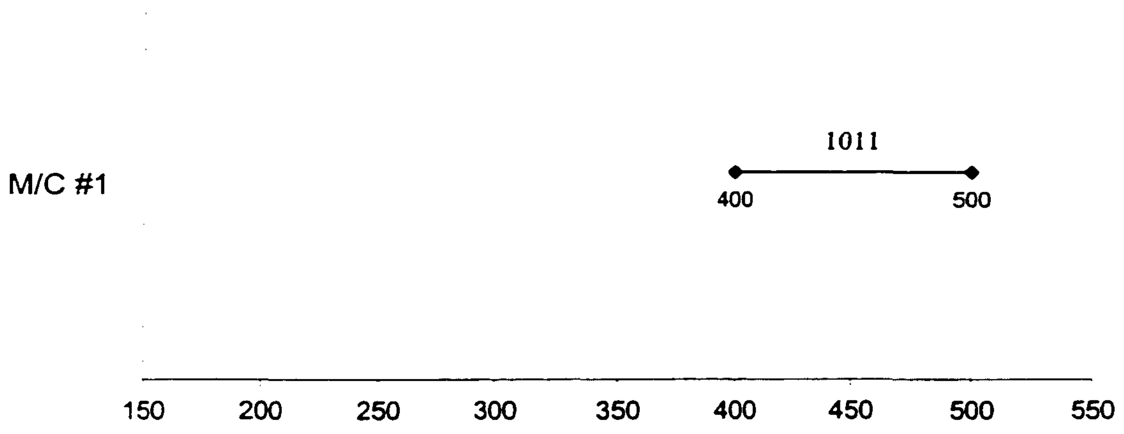


Figure C.10 Schedule operation 1011 in example 4.1

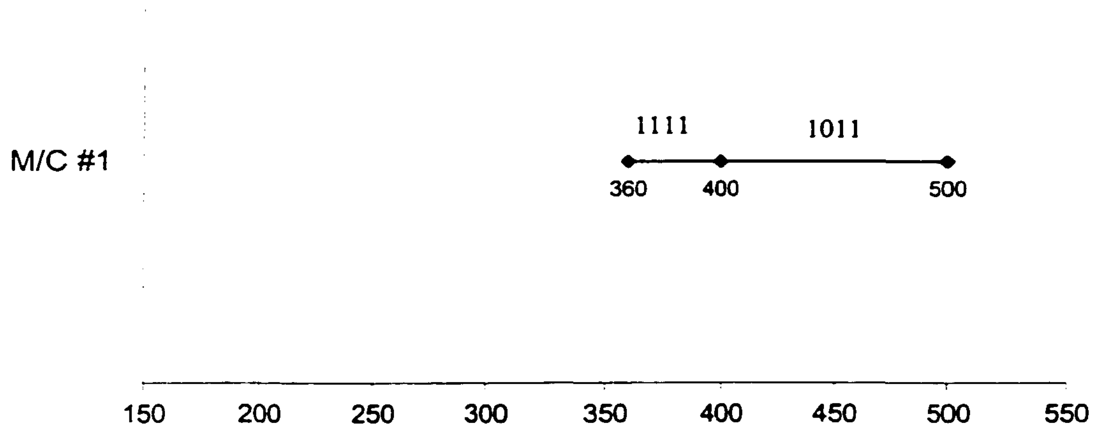


Figure C.11 Schedule operation 1111 in example 4.1

Step 2. $P = \{O_{(2011)}\}$. $O_{(ijkl)^*} = 2011$.

Step 3. Set $C_{(2011)} = 360$, $S_{(2011)} = 220$. There is no any job that conflicts with $O_{(2022)}$.
Go to step 7.

Step 7. Schedule $O_{(2011)}$ on machine

7a. Set $C_{2011} = 360$ and $S_{2011} = 220$ (see Fig C.12), $TIME = 220$.

7b. $\sigma = \phi$ and $\pi = \{O_{(1011)}, O_{(1111)}, O_{(2011)}\}$.

7c. Since $|\sigma| = 0$, then *MCE* algorithm stops.

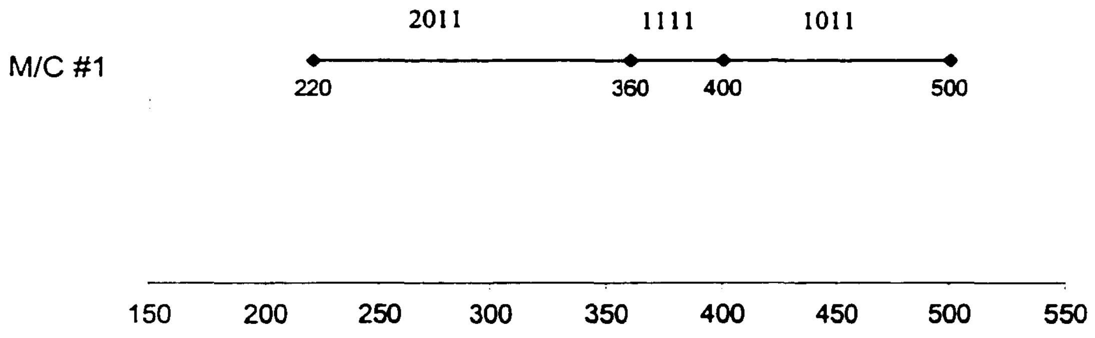


Figure C.12 Schedule operation 2011 in example 4.1

C.3 Algorithm V illustration

To illustrate the steps of algorithm V (pages 69-72), consider example 4.1 in Chapter 4 (page 61), whose product structures and parameters are described as in Figure 4.1 (page 58) and Table 4.1 (page 62), respectively.

Step 0. Set $M = \phi$. For each operation, set it as an unscheduled operation.

Step 1. Construct an ideal solution.

1a. $S_{(1011)} = 400$, $C_{(1011)} = 500$, $S_{(1111)} = 360$, $C_{(1111)} = 400$, $S_{(1312)} = 320$, $C_{(1312)} = 360$, $S_{(1412)} = 300$, $C_{(1412)} = 360$, $S_{(1213)} = 350$, $C_{(1213)} = 400$, $S_{(2011)} = 360$, $C_{(2011)} = 500$, $S_{(2113)} = 260$, $C_{(2113)} = 360$, $S_{(2212)} = 310$, $C_{(2212)} = 360$, $S_{(2312)} = 240$, $C_{(2312)} = 360$.

Step 2. Determine machine conflict operation

2a. Let λ be the set of unscheduled operations, which are ready operations.

$\lambda = \{ O_{(1011)}, O_{(2011)} \}$.

2b. Select the operation $O_{(ijkl)^*} = O_{(2011)}$.

2c. $O_{(1011)}$ conflicts with $O_{(2011)}$. Set $\Omega_{(2011)} = \{O_{(1011}, O_{(2011)}\}$ and go to step 3.

Step 3. Apply *MCE* method to eliminate machine conflicts of all operations in $\Omega_{(2011)}$, The sequence obtained from *MCE* method is $2011 \rightarrow 1011$. Select $O_{(1011)}$ to schedule (i.e., $S_{(1011)} = 400$, $C_{(1011)} = 500$). Set $O_{(1011)}$ as scheduled operation. Go to step 4.

Step 4. Since some of operations in the problem are not yet scheduled, then set all unscheduled operation in an ideal solution while keeping unchanged the schedule of operations $O_{(1011)}$ (see Fig. 4.2 page 63)and go to step 2.

Step 2. Step 2. Determine machine conflict operation

2a. Let λ be the set of unscheduled operations, which are ready operations.

$\lambda = \{ O_{(1111)}, O_{(2011)} \}$.

2b. Select the operation $O_{(ijkl)^*} = O_{(2011)}$.

2c. $O_{(1111)}$ conflicts with $O_{(2011)}$. Set $\Omega_{(2011)} = \{O_{(1111)}, O_{(2011)}\}$ and go to step 3.

Step 3. Apply *MCE* method to eliminate machine conflicts of all operations in $\Omega_{(2011)}$, The sequence obtained from *MCE* method is $2011 \rightarrow 1111$. Select $O_{(1111)}$ to schedule (i.e., $S_{(1111)} = 360$, $C_{(1111)} = 400$). Set $O_{(1111)}$ as scheduled operation. Go to step 4.

Step 4. Since some operations in the problem are not yet scheduled, then set all unscheduled operation in an ideal solution while keeping the schedule of operations $O_{(1011)}$ and $O_{(1111)}$ fixed as determined (see Fig. 4.2 page 63) and go to step 2.

Step 2. Step 2. Determine machine conflict operation

2a. Let λ be the set of unscheduled operations, which are ready operations. $\lambda = \{ O_{(2011)}, O_{(1312)}, O_{(1412)} \}$.

2b. Select the operation $O_{(ijkl)^*} = O_{(2011)}$.

2c. $O_{(2011)}$ conflicts with $O_{(1011)}$ and $O_{(1111)}$ (see Fig. 4.2, page 63). But $O_{(1011)}$ and $O_{(1111)}$ are scheduled operation in machine 1. Then, schedule $O_{(2011)}$ on machine 1 at the starting time of $O_{(1111)}$ (i.e., $C_{(2011)} = 360$ and $S_{(2011)} = 220$). Set $O_{(2011)}$ as a scheduled operation. Go to step 4.

Step 4. Since some operations in the problem are not yet scheduled, then set all unscheduled operation as an ideal solution while keep the schedule of operations $O_{(1011)}$, $O_{(1111)}$, and $O_{(2011)}$ fixed (see Fig. 4.3 page 64) and go to step 2.

Step 2 to step 4 of algorithm V is repeated until all unscheduled operation are scheduled as shown in Figure 4.5 (page 65), then the algorithm goes to step 5.

Step 5. Calculate total earliness cost of the solution as shown in Figure 4.5 (page 65). The total earliness cost = *9300* units.

Step 6. Set the current solution as shown in Figure 4.5 (page 65) to be the current best solution, called *R*. Also set $R'_1 \leftarrow R$.

Step 7. Search for improved solution by evaluating the changing of the sequence position of movable operations in the current schedule. For each operation, $O_{(ijkl)'}$,in R, repeat the step 7a.

Let consider $O_{(ijkl)'}$ = $O_{(1312)}$ in Figure 4.5 (page 65).

7a. Determine the operations which are scheduled before $O_{(1312)}$ in R and can be moved to the position of $O_{(1312)}$. Let $P_{(1312)}$ be the set of operations that can be moved to take the position of $O_{(1312)}$ in the R , then $P_{(1312)} = \{ O_{(1412)} \}$ and $O_{(ijkl)''} = O_{(1412)}$ in this example.

7b. Remove operation $O_{(1312)}$ from R and set $O_{(1312)}$ to be an unscheduled operation.

7c. Let $LTIME$ be the latest available time on machine after removing $O_{(1312)}$. (i.e. $LTIME = S_{(1312)} = 360$).

7d. Schedule $O_{(1412)}$ into the previous position of $O_{(1312)}$ in R, set $C_{(1412)} = 360$, $S_{(1412)} = 300$. Set $O_{(1412)}$ as a scheduled operation.

7e. Set all operations, $O_{(ijkl)}$, except $O_{(1412)}$ which are previously scheduled before $O_{(1312)}$ in R (i.e., $C_{(ijkl)} \leq 320$) as unscheduled operations. At this step, $O_{(1312)}$, $O_{(2212)}$, $O_{(2312)}$, and $O_{(2113)}$ are set as unscheduled operations.

7f. Repeat step1 – step5 to schedule the unscheduled operations (see Figure C.13) , called R_1', and keep R_1' in M.

7g. Reset $R \leftarrow R_1'$.

Repeat step 7 for the $O_{(ijkl)'}$ = $O_{(2212)}$ and $O_{(1111)}$ in Figure 4.5 (page 65). The solutions also are kept in M. Since all the solutions in M are not better than the solution as shown in Figure 4.5, then the solution as shown in Figure 4.5 is the best solution. Then, the algorithm V stops.

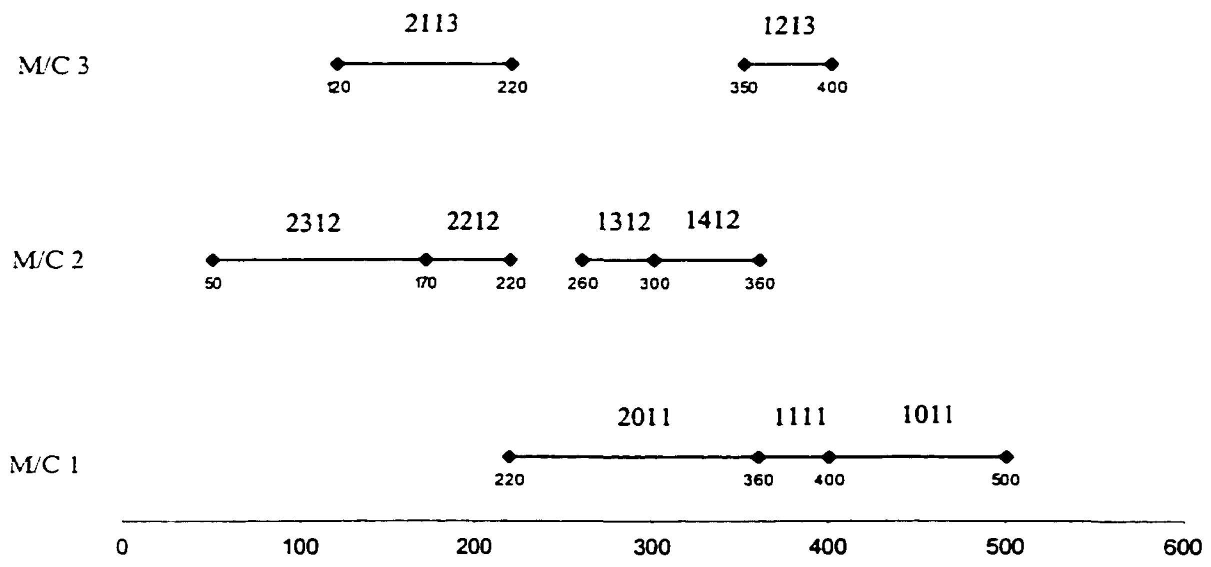


Figure C.13 The schedule of example 4.1 after moving operation 1412 to the position of operation 1312 in Figure 4.5

C.4 Algorithm VI illustration

To illustrate the steps of algorithm VI, consider example 4.2 in Chapter 4 (page 76), whose product structures and parameters are described as in Figure 4.7(page 77) and Table 4.2(page 76), respectively. The step by step of employing algorithm VI to example 4.2 are described as follows:

Step 0. Set $M = \phi$ and $P_1 = P_2 = P_3 = \phi$.

Step 1. Employ algorithm V to construct an initial schedule, called schedule R (see Figure 4.9 page 79).

Step 2. Check feasibility of R. Since R is an infeasible solution (i.e., $S_{(3413)} < 0$), go to step 3.

Step 3. While maintaining precedence relationship between operations, perform rightward shift in *R* until feasible solution is obtained (i.e., $S_{(3413)} = 0$). This schedule is as shown in Figure 4.11 (page 80). Keep this solution as a feasible solution in *M*, where *M* is a set of feasible schedules. Set schedule back to *R* (see Fig. 4.9 page 79).

Step 4. Select the earliest starting time of all operations *B* in *R* (i.e., $S_{(ijkl)^*} \le \min_{O(ijkl)\in B} \{S_{(ijkl)}\}$), where *B* is the set of all operations in *R*. In this step, $S_{(ijkl)^*} \le 0$.

Let *G* be the appropriate amount of tardiness. Set $G = |S_{(3413)}| = |-8| = 8$ (see Figure 4.9 page 79).

Step 5. For each product *i*, set the virtual due-date of product equal to the actual due-date plus the appropriate amount of tardiness (i.e., virtual due-date of product $i = D_i + G$). and repeat the following steps (5a-5c).

At this step there are three virtual due-date sets *"68-50-40"*, *"60-58-40"*, and *"60-50-48"*.

Lets consider *"68-50-40"* due-date set.

5a. Employ algorithm V to schedule the sequence of operations based on *"68-50-40"* due-date set. The schedule is the same as shown in Figure 4.11 (page 80).

5b. It is a feasible solution. Then keeps this schedule in *M*,

5c. Reset the schedule back to *R* (see Fig. 4.9 page 79).

Repeat step 5 for the *"60-50-48"* and *"60-58-40"* due-date sets and keep the feasible solutions in *M*. After repeating all due-date sets go to step 6.

Step 6. Select the best solution in *M*, called *R'*, and set *R* = *R'*. In this example, the best solution is the schedule from *"68-50-40"* due-date set. The solution is selected to be the current best solution at this moment.

Step 7. Set $M = \phi$. For each product *i* in *R* (product 1, 2, and 3), do the following steps (7a-7d).

Lets consider product 1.

7a. Remove all operations of product 1 from *R* (see Figure C.14) .

7b. Let *R** be the schedule after removing product 1 from *R*. For each

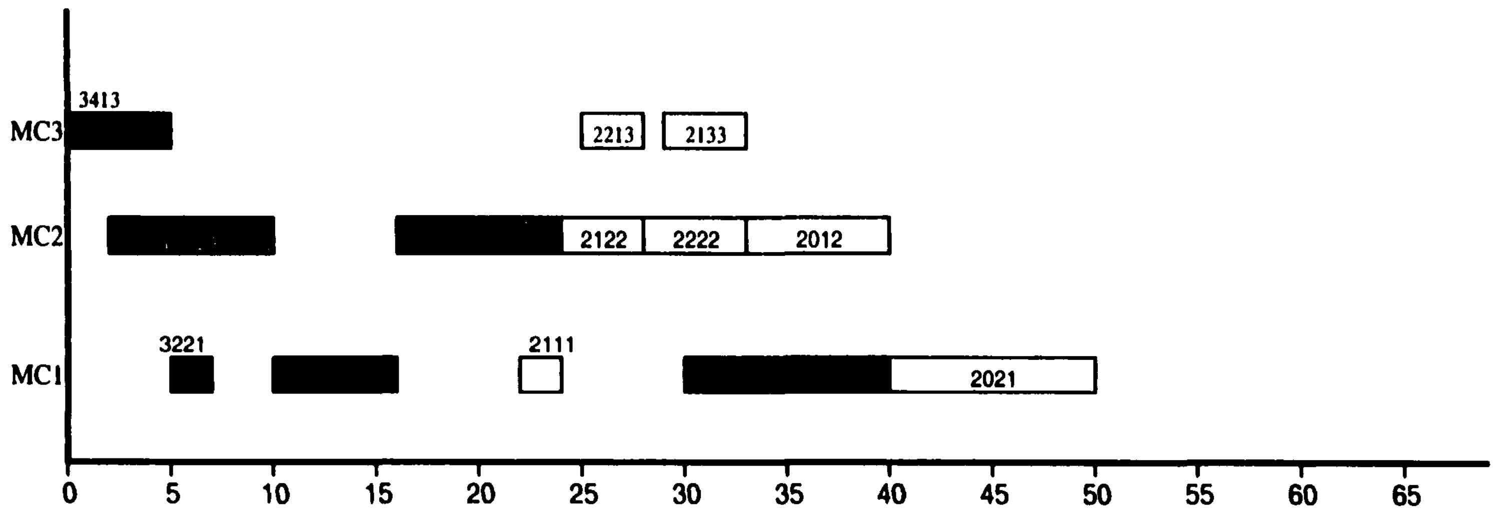


Figure C.14 A schedule in example 4.2 after removing product 1 from the schedule in Figure 4.11.

operation $O_{(ijkl)'}$ in R^*, do the steps (7b(1)-7b(2)).

7b(1). Shift $O_{(ijkl)'}$ to the right without violating precedence constraint. Let $Q_{(ijkl)'}$ be the amount that $O_{(ijkl)'}$ can be shifted to the right in time. Since no operations in R^* (see Figure C.14) can be shifted to the right without violating precedence constraint (i.e., $Q_{(ijkl)'} = 0$). Then the only member of P_1 is 0 . Set $P_1 = \{0\}$.

7b(2). Set the schedule back to R^* (see Fig. 4.11 page 80).

7c. Let $P_1(a)$ be a member of P_1. (i.e., $P_1(a) = 0$) do the following steps.

7c(1). Set virtual due-date of product *1* equal to the due-date of product *1* plus $P_1(a)$ (i.e., $68+0$). Then the due-date set is still *"68-50-40"*.

7c(2). Based on virtual *"68-50-40"* due-date set, the solution is still the schedule shown in Figure 4.11.

7d. Reset the schedule back to *R*.

Step 7 is repeat for products 2 and 3 (i.e., P_2 and P_3). The solutions are kept in *M* and go to step 8.

Step 8. Select the best solution in *M*, called *R'*. Since the best solution is as shown in Figure 4.11 (page 80), then there is no improvement in step 7. Algorithm VI stops.

APPENDIX D
SENSITIVITY ANALYSIS

In this section, sensitivity analysis is performed on two problem types (i.e., minimizing earliness penalty and minimizing the sum of weighted earliness and tardiness penalties) on single machine problems. In minimizing earliness penalty problem, sensitivity analysis is studied in two different cases:

1) Variation in the percentage of conflicts in an ideal solution where the percentage of conflict is defined as the percentage of processing times of operations that overlap in an ideal solution. Mathematically, this is defined as

$$\text{Percentage of conflicts} = \left(\frac{\text{Overlapped processing times of operations in ideal solution}}{\text{Total processing times of all operations in the problem}}\right) * 100$$

2) Variation of the ratio of average Y_i to the number of jobs (i.e., $\frac{\left(\sum_{i=1}^{N} Y_i\right)}{N}$ where $Y_i = \frac{E_i}{t_i}$). This is the average ratio of earliness penalty and processing time to the number of jobs.

In the problem of minimizing the sum of weighted earliness and tardiness penalties, the sensitivity analysis is studied under four different cases:

1) Variation of the the percentage of conflicts in ideal solution.

2) Variation of the ratio of average Y_i to the number of jobs (i.e., $\frac{\left(\sum_{i=1}^{N} Y_i\right)}{N}$ where $Y_i = \frac{E_i}{t_i}$). This is the average ratio of earliness penalty and processing time to the number of jobs.

3) Variation of the ratio of average Z_i to the number of jobs (i.e., $\frac{\left(\sum_{i=1}^{N} Z_i\right)}{N}$ where $Z_i = \frac{W_i}{t_i}$). This is the average ratio of tardiness penalty and processing time to the number of jobs.

4) Variation of the average of ratio of earliness penalty and tardiness penalty to the number of jobs (i.e., $\frac{\left(\sum_{i=1}^{N} \frac{E_i}{W_i}\right)}{N}$),

The sensitivity analysis is limited to the single machine problem because of the fact that the optimality conditions derived for the single machine problem formed the basis for the subsequent problem cases, including the assembly job shop problems.

In this section, four examples involving different number of jobs (i.e., 15, 20, 25, 30 jobs) are randomly generated for the problem with earliness cost minimization and another four problems of different sizes in job count (i.e., 15, 20, 25, 30 jobs) are randomly generated for the case of minimizing the sum of weighted earliness and tardiness cost.

In the case of the percentage of conflict variation, the due-dates of jobs are varied while the other parameters are unchanged. Only the earliness penalties are varied in the case of average ratio of earliness penalty and processing time variation. Only the tardiness penalties are varied in the case of variation of the average ratio of tardiness penalty and processing time. Both earliness and tardiness penalties are varied when the case of variation of average ratio of earliness and tardiness penalties is studied.

Tables D.1 to D.8 and Figures D.1 to D.8 show the sensitivity analysis of problems with earliness penalty minimization. Tables D.9 to D.24 and Fig. D.9 to D.24 present the sensitivity analysis of problems with the sum of weighted earliness and tardiness penalty minimization.

In studying sensitivity analysis, an example was first randomly generated. It means that all parameters were generated randomly. It consists of the processing times, earliness penalties, and due-dates of all jobs in single machine problem with earliness penalty minimization cases. For the cases of minimizing the sum of weighted earliness and tardiness penalties, the processing times, earliness penalties, tardiness penalties and due-dates of all jobs were randomly generated. The sensitivity analysis for the case involving variation in the percentage of conflicts was studied by changing the due-dates of jobs in the problem while keeping the other parameters (i.e., the processing time, earliness penalties and tardiness penalties) unchanged. The changing of due-dates effects the percentage of conflicts of jobs in an ideal solution. When the percentage of conflicts was changed algorithms I and II were applied to solve the problem and the results obtained by the two algorithms are compared to each other. This was the same for the case of algorithms III and IV. For example, in Table D.1 and Figure D.1, an example of a 15 job size problem was randomly generated. Then the processing times and earliness penalties of all jobs were fixed while the due-dates were changed. It generated a set of percentage of conflicts, which ranged from 8% to 97%. Both algorithms I and II were applied to solve this set of problems. It showed that both algorithms yielded the same solution. In the 8% conflict case, algorithms I and II gave the same solution with 40 in total cost. The remaining problem instances with changes in percentage of job conflicts were similarly analyzed.

For the case involving the variation of the average ratio of earliness penalty and processing time to the number of jobs (i.e., $\frac{\left(\sum_{i=1}^{N} Y_i\right)}{N}$ where $Y_i = \frac{E_i}{t_i}$), only the earliness penalties of jobs were changed while the other parameters were fixed. For example, in Table D.5 and Figure D.5, the range of average E/t was varied from 0.37 to 2.1. Algorithms I and II were applied to solve this set of problems. In the 0.37 E/t case, both algorithms generated the same schedule with 277 in total cost.

For the case involving the variation of the average ratio of tardiness penalty and processing time to the number of jobs (i.e., $\frac{\left(\sum_{i=1}^{N} Z_i\right)}{N}$ where $Z_i = \frac{W_i}{t_i}$), only the tardiness penalties of jobs were changed while the other parameters were fixed. For example, in Table D.17 and Figure D.17, the range of average W/t was varied from 1.7 to 4.32. Algorithms III and IV were applied to solve this set of problems. In the 1.7 W/t case, algorithm III generated a schedule with 267 in total cost, which was worse than the schedule constructed by algorithm IV, which had 214 in total cost.

For the case involving the variation of the average ratio of earliness penalty and tardiness penalty to the number of jobs (i.e., $\frac{\left(\sum_{i=1}^{N}\frac{E_i}{W_i}\right)}{N}$), both earliness and tardiness penalties were randomly changed while the other parameters were unchanged. For example, in Table D.21 and Figure D.21, the range of average E/W was varied from 0.31 to 0.84. Algorithms III and IV were applied to solve this set of problems. In the 0.31 E/W case, algorithm III generated a schedule with 243 in total cost, which was the same schedule generated by algorithm IV.

Table D.1 Performance comparison of algorithms I & II relative to the percentage of conflict in a 15 job problem with earliness penalty minimization

% Conflict	Alg.I (total cost)	Alg.II (total cost)
8	40	40
30	180	174
43	186	186
47	310	310
69	1033	1033
97	2245	2245

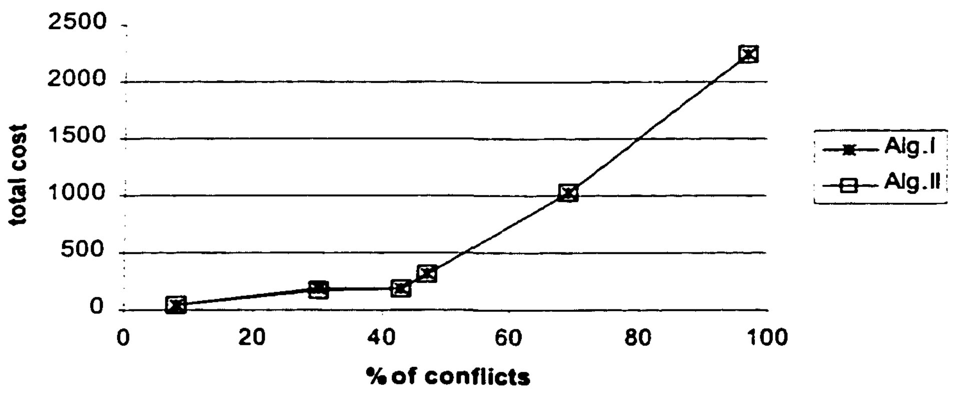


Figure D.1. Performance comparison of algorithms I & II relative to the percentage of conflict in a 15 job problem with earliness penalty minimization

Table D.2 **Performance comparison of algorithms I & II relative to the percentage of conflict in a 20 job problem with earliness penalty minimization**

% Conflict	**Alg. I (total cost)**	**Alg. II (total cost)**
28	137	137
30	141	141
34	86	86
48	186	186
55	268	267
75	317	317
76	383	443
84	330	330
86	669	669
94	829	829

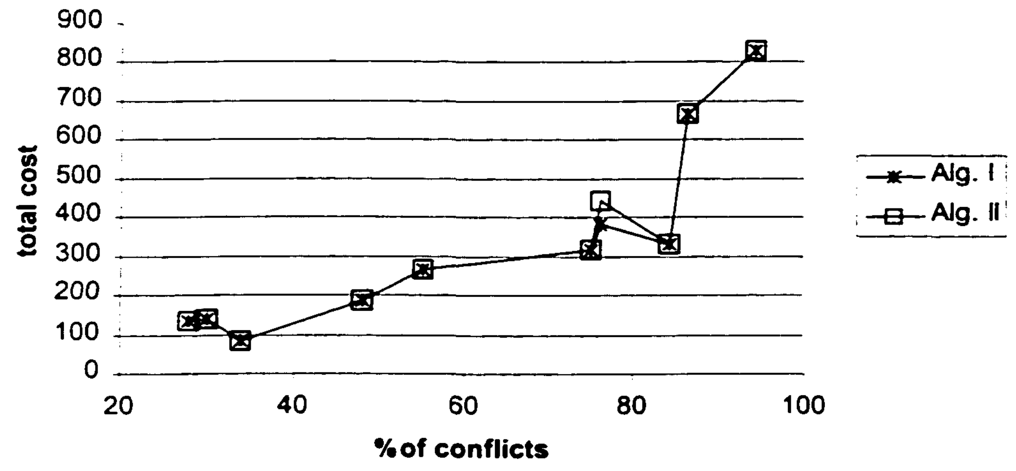


Figure D.2. Performance comparison of algorithms I & II relative to the percentage of conflict in a 20 job problem with earliness penalty minimization

Table D.3 Performance comparison of algorithms I & II relative to the percentage of conflict in a 25 job problem with earliness penalty minimization

% Conflict	Alg. I (total cost)	Alg. II (total cost)
34	375	375
35	565	565
54	461	461
65	578	578
85	709	709
86	767	924
91	824	824

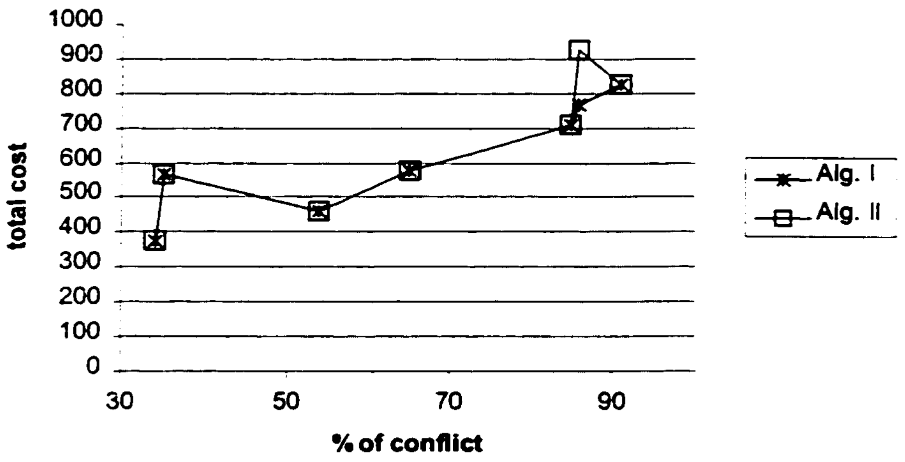


Figure D.3. Performance comparison of algorithms I & II relative to the percentage of conflict in a 25 job problem with earliness penalty minimization

Table D.4 Performance comparison of algorithms I & II relative to the percentage of conflict in a 30 job problem with earliness penalty minimization

% Conflict	Alg. I (total cost)	Alg. II (total cost)
18	160	160
38	262	262
50	304	304
51	349	336
59	277	384
67	371	358
70	394	394
78	407	407
82	403	403

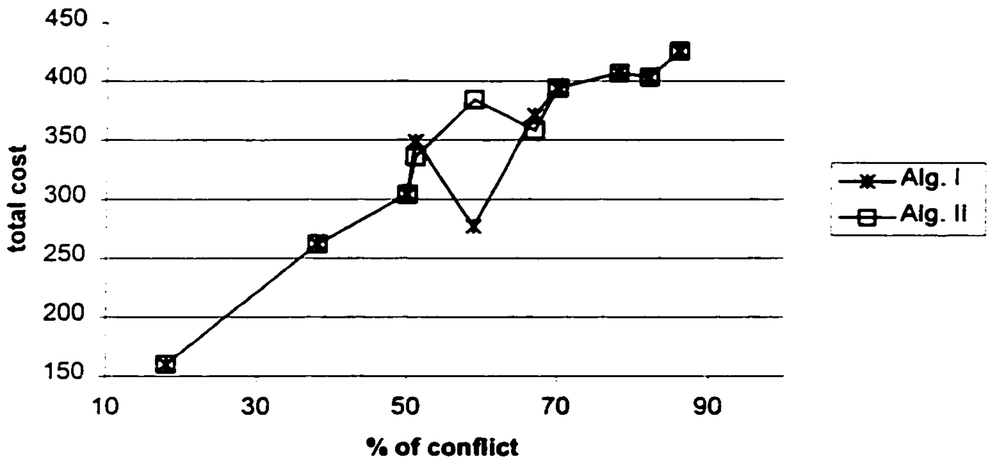


Figure D.4 Performance comparison of algorithms I & II relative to the percentage of conflict in a 30 job problem with earliness penalty minimization

Table D.5 **Performance comparison of algorithms I & II relative to the ratio of earliness penalty and processing time in a 15 job problem with earliness penalty minimization**

Average E/t	Alg. I (total cost)	Alg. II (total cost)
0.37	277	277
0.52	346	337
0.71	522	516
0.91	687	641
1.11	811	873
1.29	1033	1033
1.55	1301	1301
1.73	1503	1503
1.91	1705	1705
2.10	1907	1907

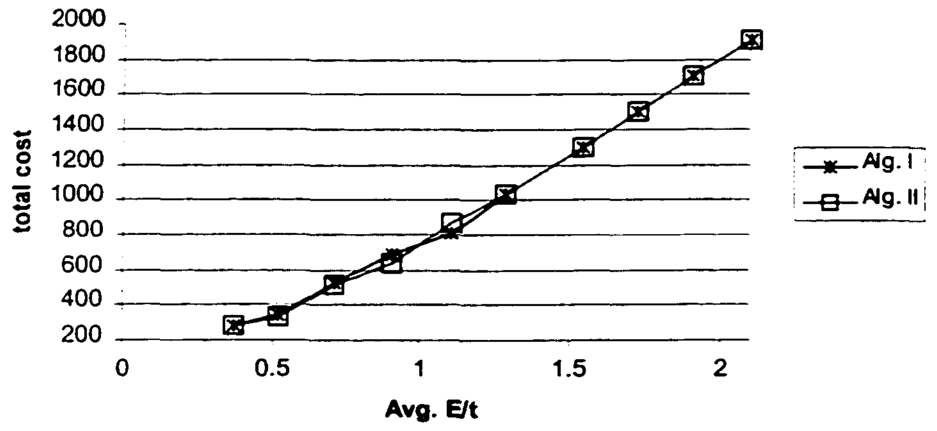


Figure D.5 **Performance comparison of algorithms I & II relative to the ratio of earliness penalty and processing time in a 15 job problem with earliness penalty minimization**

Table D.6 Performance comparison of algorithms I & II relative to the ratio of earliness penalty and processing time in a 20 job problem with earliness penalty minimization

Avg. E/t	Alg. I (total cost)	Alg. II (total cost)
0.34	46	46
0.43	77	77
0.46	56	56
0.64	86	86
0.72	77	77
1.09	164	164
1.32	200	200
1.52	230	230
1.75	266	266
1.97	302	302

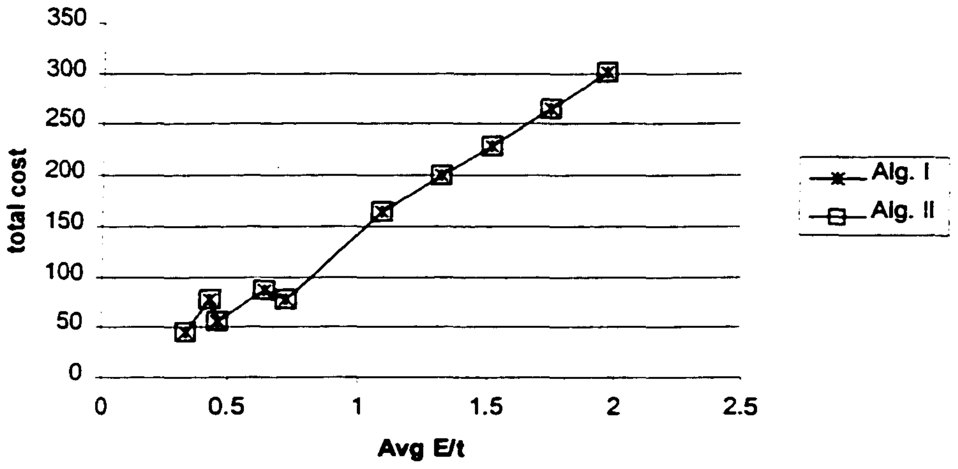


Figure D.6 Performance comparison of algorithms I & II relative to the ratio of earliness penalty and processing time in a 20 job problem with earliness penalty minimization

Table D.7 **Performance comparison of algorithms I & II relative to the ratio of earliness penalty and processing time in a 25 job problem with earliness penalty minimization**

Avg. E/t	Alg. I (total cost)	Alg. II (total cost)
0.43	239	239
0.46	242	242
0.52	269	265
0.69	407	531
0.89	565	565
1.09	723	723
1.29	880	880
1.70	1192	1181
1.90	1314	1314

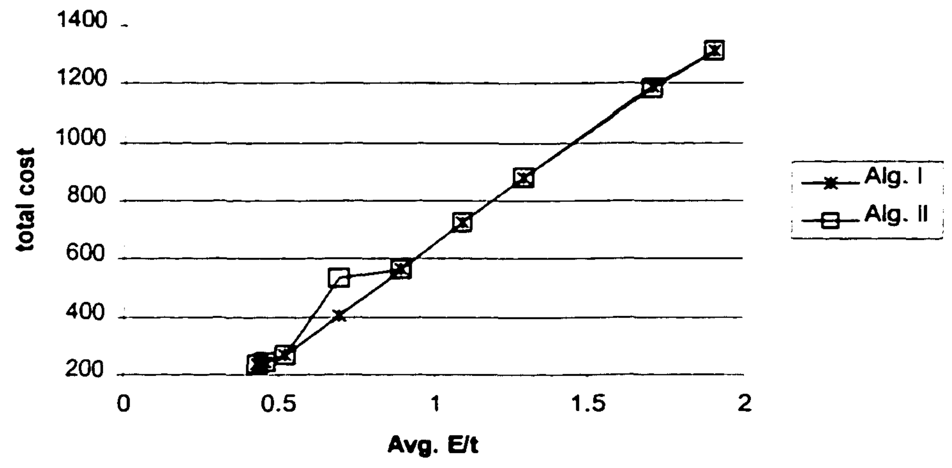


Figure D.7 **Performance comparison of algorithms I & II relative to the ratio of earliness penalty and processing time in a 25 job problem with earliness penalty minimization**

Table D.8 **Performance comparison of algorithms I & II relative to the ratio of earliness penalty and processing time in a 30 job problem with earliness penalty minimization**

Avg. E/t	Alg. I (total cost)	Alg. II (total cost)
0.50	94	94
0.60	112	112
0.80	148	148
0.86	160	160
1.01	184	184
1.27	233	233
1.47	270	270
1.89	344	344
2.30	416	416

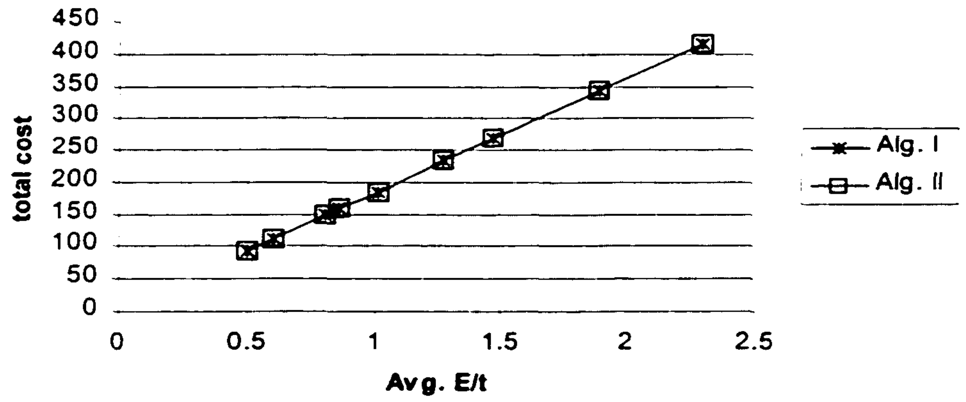


Figure D.8. **Performance comparison of algorithms I & II relative to the ratio of earliness penalty and processing time in a 30 job problem with earliness penalty minimization**

Table D.9 **Performance comparison of algorithms III & IV relative to the percentage of conflict in a 15 job problem with the sum of weighted E/T penalty minimization**

% Conflict	Alg.III (total cost)	Alg. IV (total cost)
32	389	389
39	362	362
42	375	375
45	371	371
51	780	780
58	580	567
76	612	612
84	685	687
87	620	606

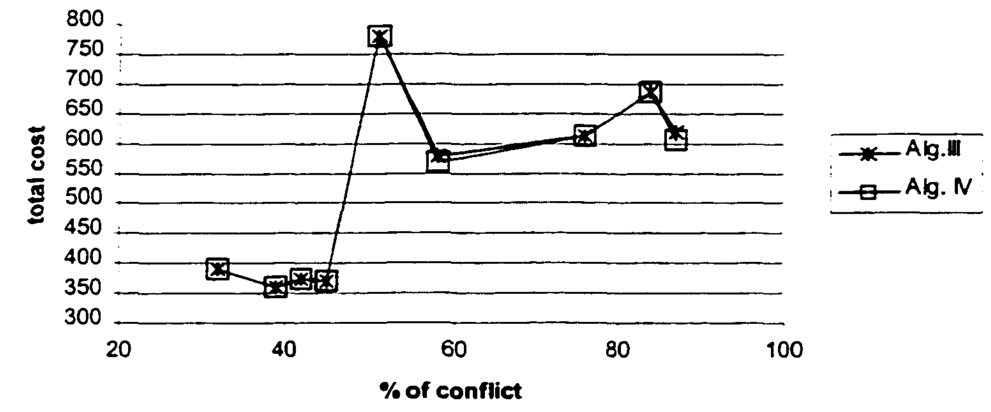


Figure D.9. **Performance comparison of algorithms III & IV relative to the percentage of conflict in a 15 job problem with the sum of weighted E/T penalty minimization**

Table D.10 Performance comparison of algorithms III & IV relative to the percentage of conflict in a 20 job problem with the sum of weighted E/T penalty minimization

% Conflict	Alg.III (total cost)	Alg. IV (total cost)
39	493	471
41	308	306
43	469	467
45	523	509
53	628	628
61	863	817
69	805	760
76	665	611
82	956	950
86	870	864

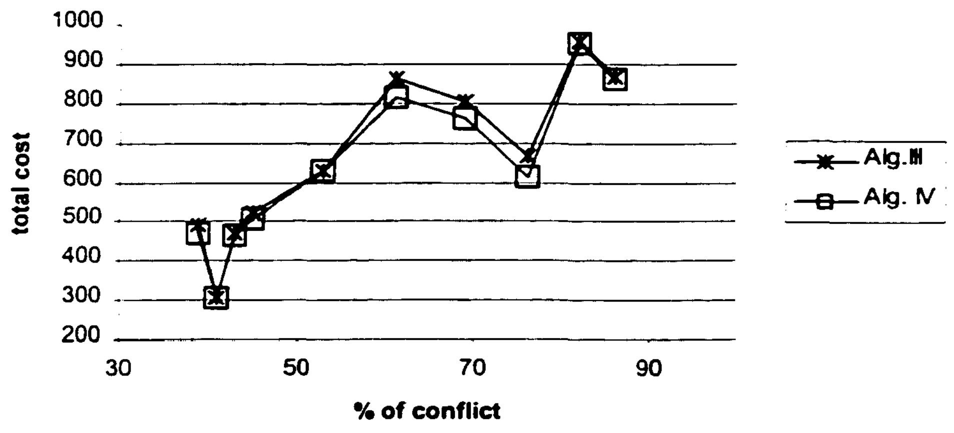


Figure D.10 Performance comparison of algorithms III & IV relative to the percentage of conflict in a 20 job problem with the sum of weighted E/T penalty minimization

Table D.11 Performance comparison of algorithms III & IV relative to the percentage of conflict in a 25 job problem with the sum of weighted E/T penalty minimization

% Conflict	Alg.III (total cost)	Alg. IV (total cost)
0.24	330	330
0.35	463	451
0.36	417	417
0.43	338	338
0.58	1066	1049
0.67	1087	1062
0.78	1192	1117
0.88	1328	1330
0.91	1467	1467
0.94	1399	1399

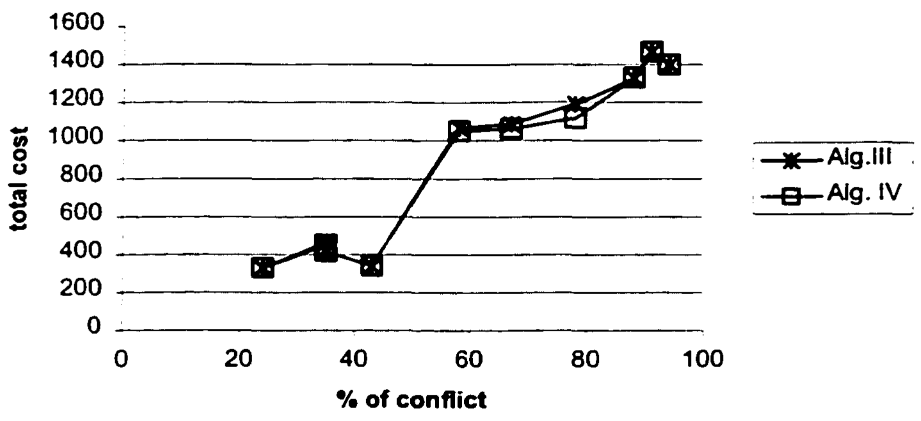


Figure D.11 Performance comparison of algorithms III & IV relative to the percentage of conflict in a 25 job problem with the sum of weighted E/T penalty minimization

Table D.12 **Performance comparison of algorithms III & IV relative to the percentage of conflict in a 30 job problem with the sum of weighted E/T penalty minimization**

% Conflict	Alg.III (total cost)	Alg. IV (total cost)
18	160	160
38	262	262
50	304	304
51	349	336
59	277	384
67	371	358
70	394	394
78	407	407
82	403	403
86	426	426

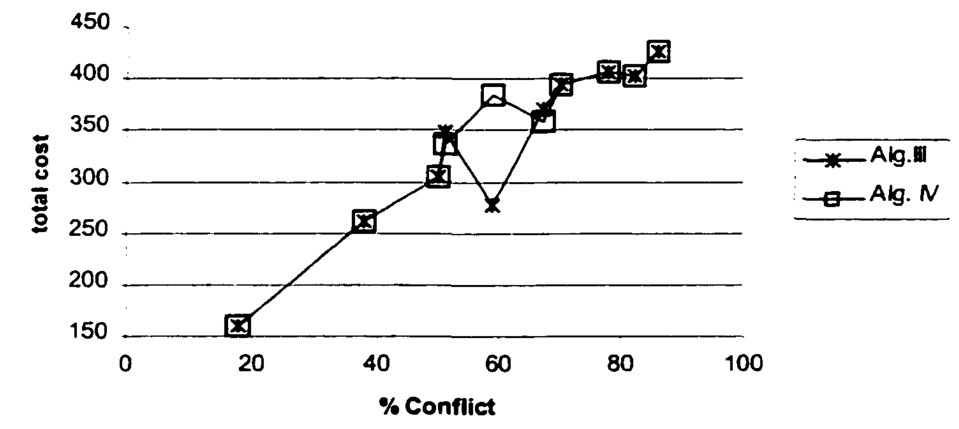


Figure D.12 **Performance comparison of algorithms III & IV relative to the percentage of conflict in a 30 job problem with the sum of weighted E/T penalty minimization**

Table D.13 Performance comparison of algorithms III & IV relative to the ratio of earliness penalty and processing time in a 15 job problem with the sum of weighted E/T penalty minimization

Avg E/t	Alg. III (total cost)	Alg.IV (total cost)
0.28	339	339
0.34	341	341
0.42	343	343
0.53	345	345
0.68	382	350
0.96	356	356
1.23	362	362
1.52	368	368
1.80	374	374
2.10	380	380

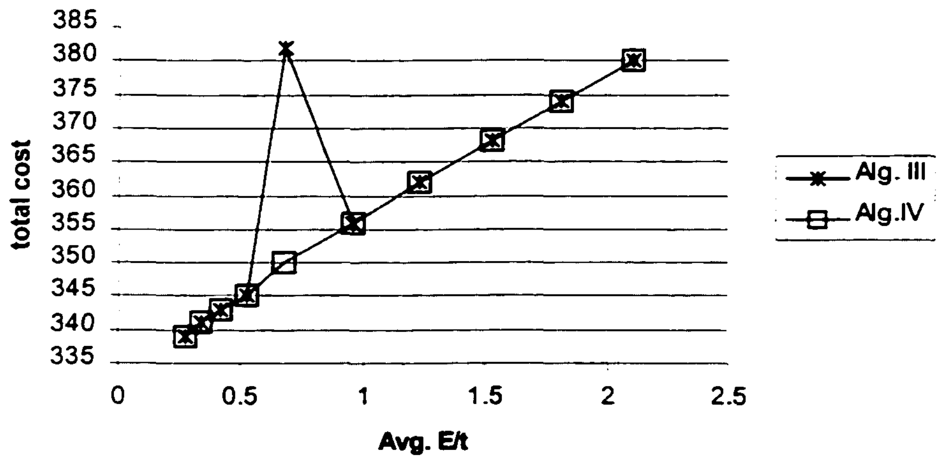


Figure D.13 Performance comparison of algorithms III & IV relative to the ratio of earliness penalty and processing time in a 15 job problem with the sum of weighted E/T penalty minimization

Table D.14 Performance comparison of algorithms III & IV relative to the ratio of earliness penalty and processing time in a 20 job problem with the sum of E/T penalty minimization

Avg E/t	Alg. III (total cost)	Alg.IV (total cost)
0.33	306	301
0.63	344	344
0.76	367	351
1.19	426	426
1.44	452	450
1.71	497	496
1.99	542	542
2.30	587	587

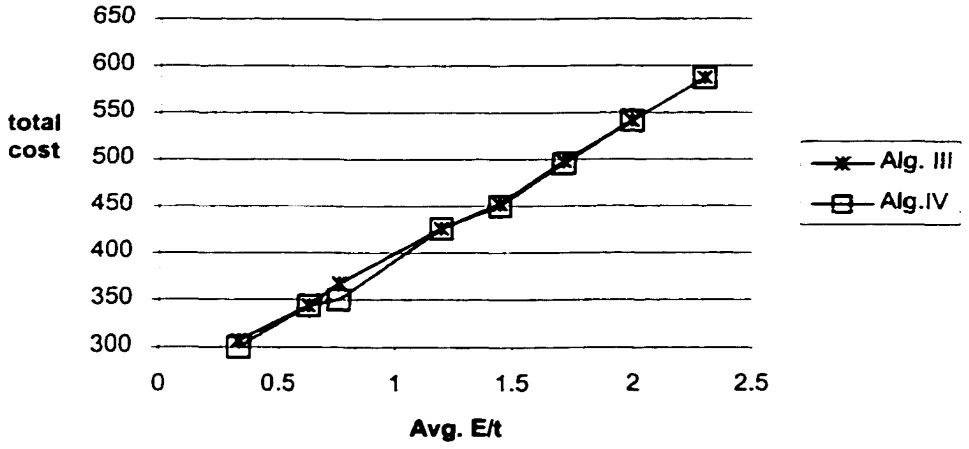


Figure D.14 Performance comparison of algorithms III & IV relative to the ratio of earliness penalty and processing time in a 20 job problem with the sum of weighted E/T penalty minimization

Table D.15 **Performance comparison of algorithms III & IV relative to the ratio of earliness penalty and processing time in a 25 job problem with the sum of E/T penalty minimization**

Avg E/t	Alg. III (total cost)	Alg.IV (total cost)
0.45	218	218
0.62	249	249
0.88	281	281
1.15	338	338
1.36	380	380
1.57	438	432
1.70	472	440
1.83	480	480
2.10	522	522
2.22	579	552

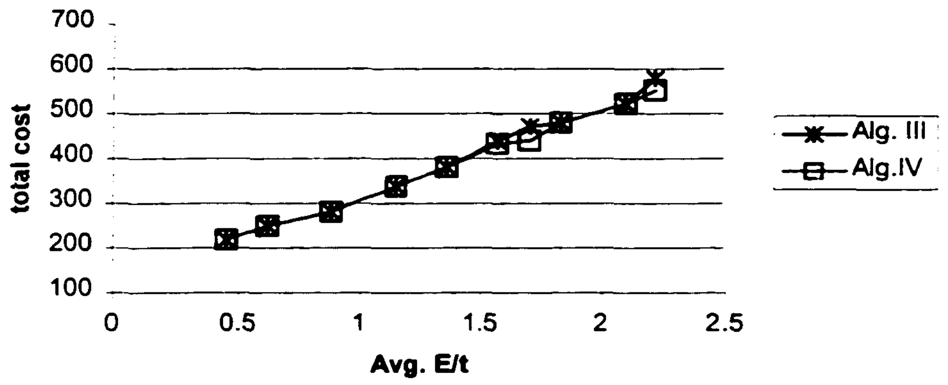


Figure D.15 **Performance comparison of algorithms III & IV relative to the ratio of earliness penalty and processing time in a 25 job problem with the sum of weighted E/T penalty minimization**

Table D.16 Performance comparison of algorithms III & IV relative to the ratio of earliness penalty and processing time in a 30 job problem with the sum of E/T penalty minimization

Avg E/t	Alg. III (total cost)	Alg.IV (total cost)
0.36	302	302
0.48	325	325
0.72	383	383
0.90	438	438
0.95	435	435
1.21	500	500
1.45	558	558
1.68	793	613
1.92	861	871

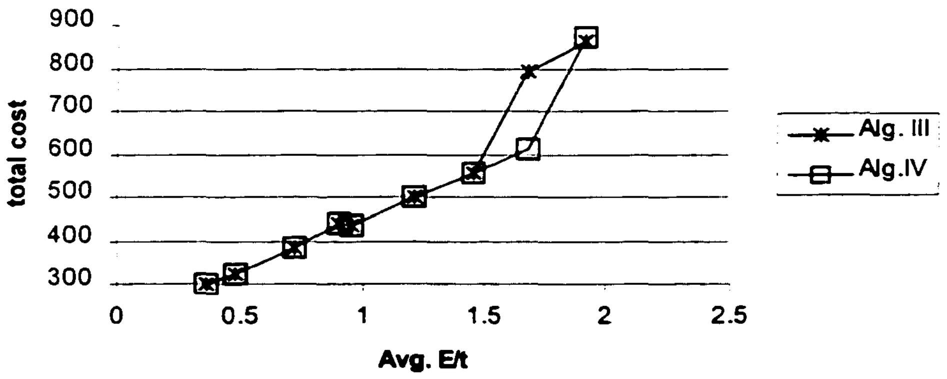


Figure D.16 Performance comparison of algorithms III & IV relative to the ratio of earliness penalty and processing time in a 30 job problem with the sum of weighted E/T penalty minimization

Table D.17 Performance comparison of algorithms III & IV relative to the ratio of tardiness penalty and processing time in a 15 job problem with the sum of E/T penalties minimization

Avg. W/t	Alg. III (total cost)	Alg. IV (total cost)
1.70	267	214
1.97	337	357
2.24	271	267
2.53	301	301
3.07	332	332
3.19	362	362
3.60	392	392
3.75	422	422
4.04	452	452
4.32	482	482

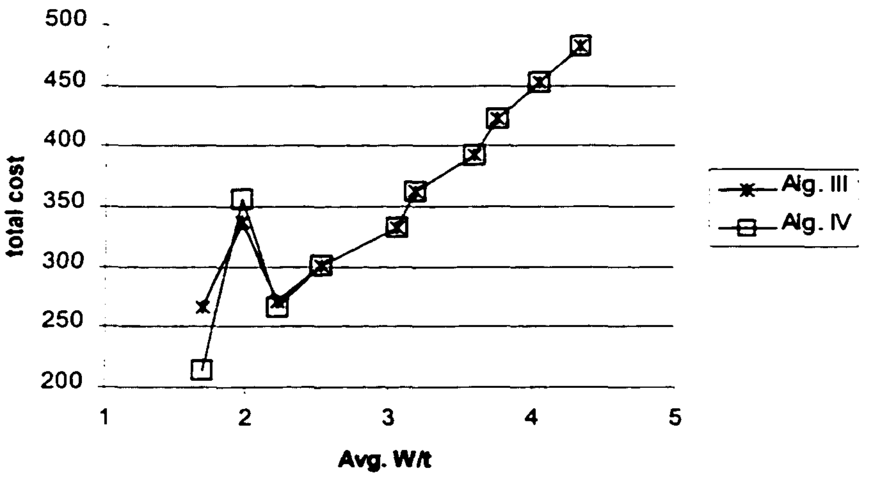


Figure D.17 Performance comparison of algorithms III & IV relative to ratio of tardiness penalty and processing time in a 15 job problem with the sum of weighted E/T penalty minimization

Table D.18 **Performance comparison of algorithms III & IV relative to the ratio of tardiness penalty and processing time in a 20 job problem with the sum of E/T penalty minimization**

Avg. W/t	Alg. III (total cost)	Alg. IV (total cost)
1.34	313	313
1.67	328	328
1.87	342	342
2.14	363	363
2.42	384	384
2.70	405	405
2.93	426	426
3.49	468	468
3.77	483	489
4.04	510	510
4.31	531	531

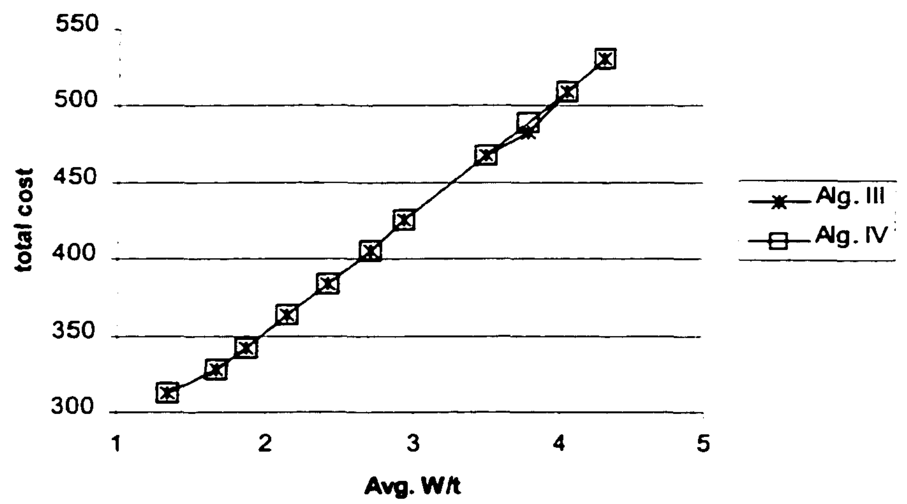


Figure D.18 **Performance comparison of algorithms III & IV relative to the ratio of tardiness penalty and processing time in a 20 job problem with the sum of weighted E/T penalty minimization**

Table D.19 Performance comparison of algorithms III & IV relative to the ratio of tardiness penalty and processing time in a 25 job problem with the sum of E/T penalty minimization

Avg. W/t	Alg. III (total cost)	Alg. IV (total cost)
1.49	260	260
1.72	269	269
1.99	282	282
2.26	299	299
2.53	315	315
2.82	338	338
3.05	347	347
3.32	363	363
3.60	379	379
3.87	401	401

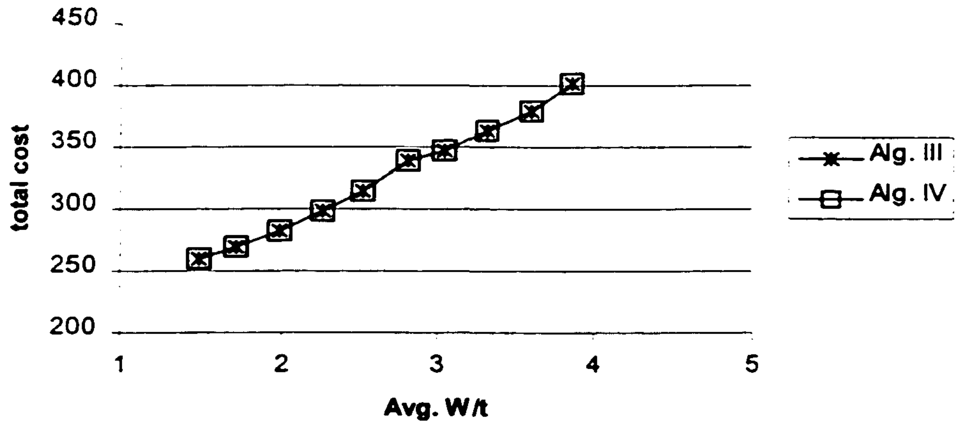


Figure D.19 Performance comparison of algorithms III & IV relative to the ratio of tardiness penalty and processing time in a 25 job problem with the sum of weighted E/T penalty minimization

Table D.20 **Performance comparison of algorithms III & IV relative to the ratio of tardiness penalty and processing time in a 30 job problem with the sum of E/T penalty minimization**

Avg. W/t	Alg. III (total cost)	Alg. IV (total cost)
1.09	288	288
1.31	310	310
1.53	333	333
2.58	412	412
2.82	435	435
2.84	435	438
3.06	458	458
3.30	481	481
3.53	504	504
3.80	527	527
4.03	550	550

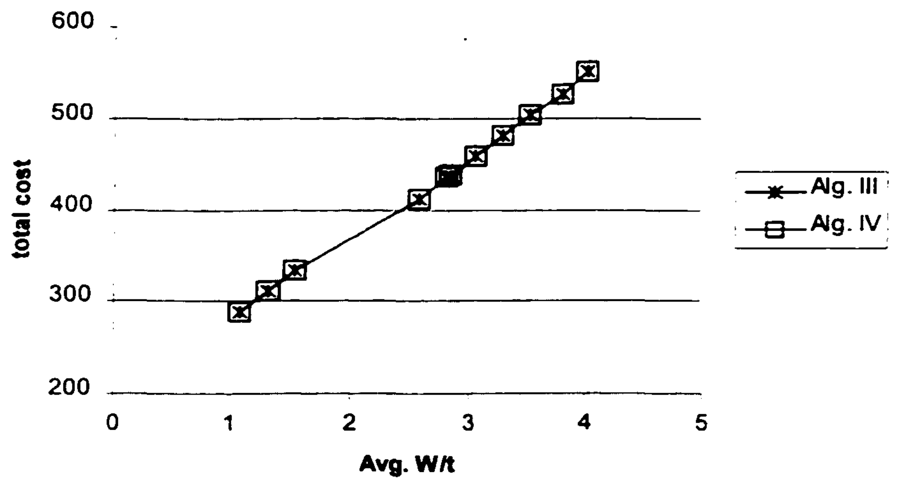


Figure D.20 **Performance comparison of algorithms III & IV relative to the ratio of tardiness penalty and processing time in a 30 job problem with the sum of weighted E/T penalty minimization**

Table D.21 **Performance comparison of algorithms III & IV relative to the ratio of earliness penalty and tardiness penalty in a 15 job problem with the sum of E/T penalty minimization**

Avg. E/W	Alg.III (total cost)	Alg.IV (total cost)
0.31	243	243
0.33	231	231
0.37	240	240
0.40	228	228
0.47	250	250
0.51	247	247
0.59	269	269
0.65	257	257
0.79	277	277
0.84	264	264

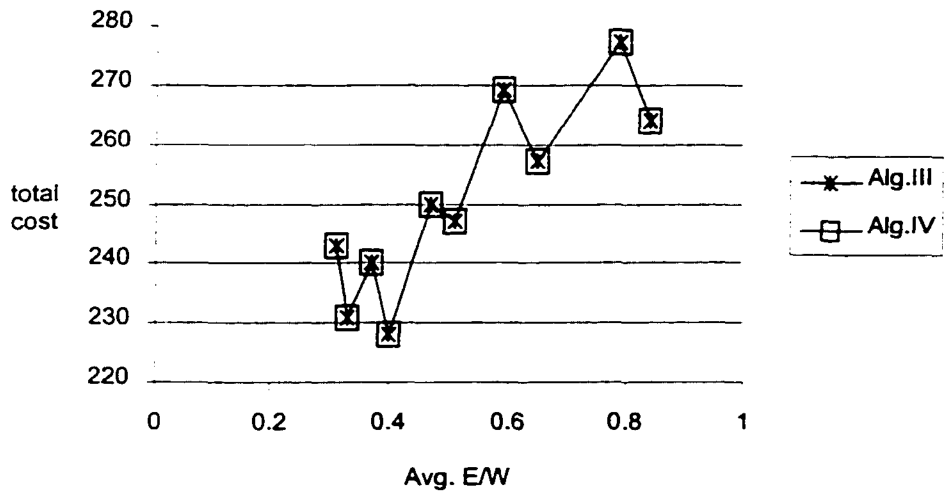


Figure D.21 **Performance comparison of algorithms III & IV relative to the ratio of earliness penalty and tardiness penalty in a 15 job problem with the sum of weighted E/T penalty minimization**

Table D.22 Performance comparison of algorithms III & IV relative to ratio of earliness penalty and tardiness penalty in a 20 job problem with the sum of E/T penalty minimization

Avg.E/W	Alg.III (total cost)	Alg.IV (total cost)
0.34	466	466
0.42	426	426
0.47	490	488
0.50	475	464
0.59	513	512
0.63	504	503
0.68	507	507
0.72	540	540
0.76	531	531
0.84	657	573

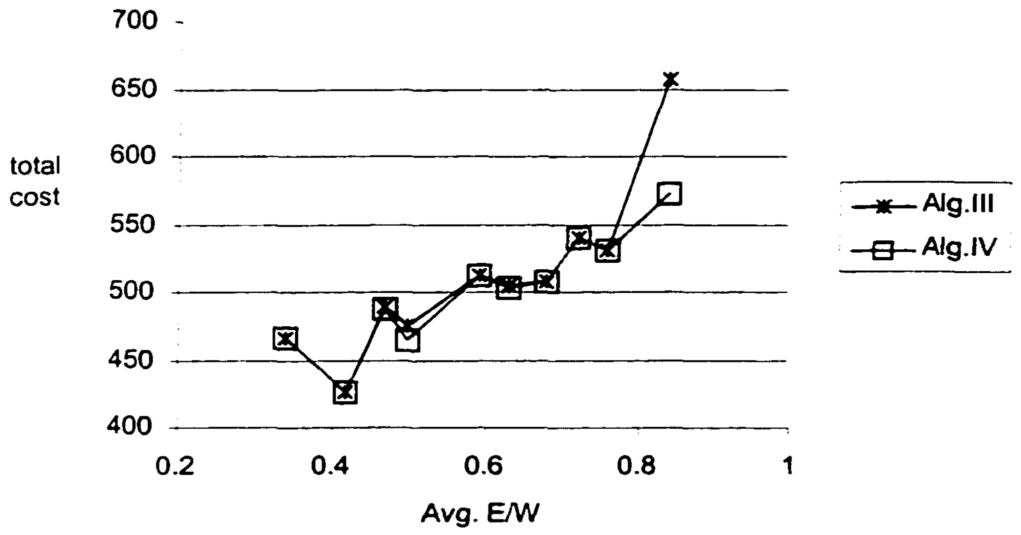


Figure D.22 Performance comparison of algorithms III & IV relative to the ratio of earliness penalty and tardiness penalty in a 20 job problem with the sum of weighted E/T penalty minimization

Table D.23 **Performance comparison of algorithms III & IV relative to the ratio of earliness penalty and tardiness penalty in a 25 job problem with the sum of E/T penalty minimization**

Avg. E/W	Alg.III (total cost)	Alg.IV (total cost)
0.32	663	663
0.45	944	941
0.49	989	986
0.54	1151	1151
0.57	1146	1146
0.64	1141	1141
0.75	1300	1300
0.80	1348	1348
0.84	1480	1480
0.91	1500	1500

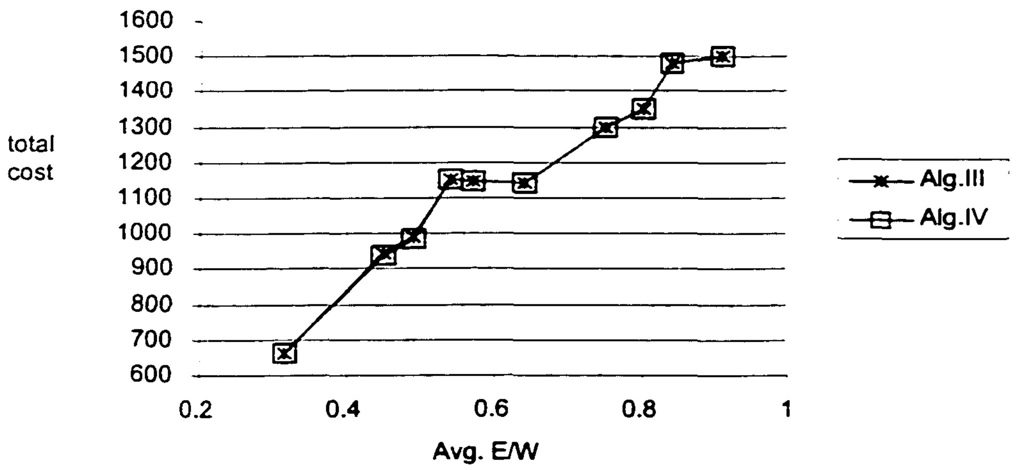


Figure D.23 **Performance comparison of algorithms III & IV relative to the ratio of earliness penalty and tardiness penalty in a 25 job problem with the sum of weighted E/T penalty minimization**

Table D.24 Performance comparison of algorithms III & IV relative to the ratio of earliness penalty and tardiness penalty in a 30 job problem with the sum of E/T penalty minimization

Avg.E/W	Alg.III (total cost)	Alg.IV (total cost)
0.32	418	418
0.35	388	388
0.37	386	386
0.40	401	401
0.44	366	366
0.48	317	317
0.50	359	359
0.53	321	321
0.60	334	334
0.90	288	288

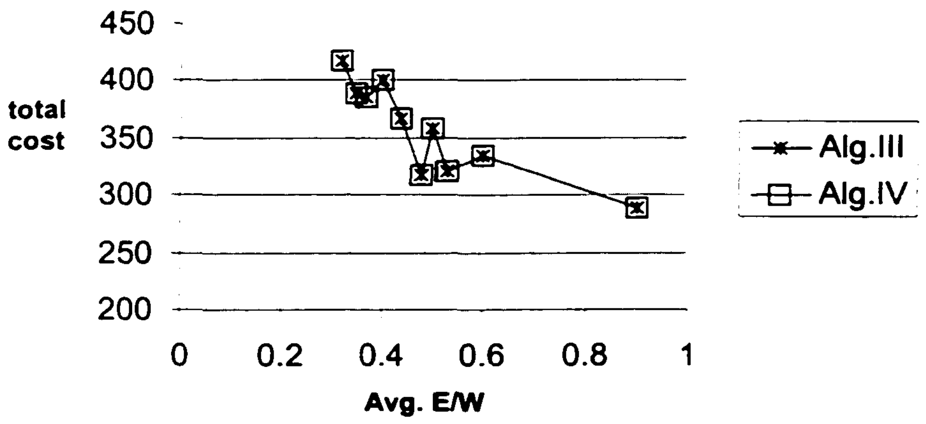


Figure D.24. Performance comparison of algorithms III & IV relative to the ratio of earliness penalty and tardiness penalty in a 30 job problem with the sum of weighted E/T penalty minimization

D.1 Inferences drawn

D.1.1 Algorithms I and II

1. Based on the percentage of job conflicts for the scenarios involving the minimization of earliness penalty, both algorithms I and II showed no clear difference in performance. The solutions they produced followed the same general trend.

2. Similarly, the solutions produced by algorithms I and II followed the same general trend relative to changes in the ratio of $\frac{\left(\sum_{i=1}^{N} Y_i\right)}{N}$ for the earliness penalty problem. In general, the relative performance of the algorithms seems to be insensitive to the changes in percentage of job conflicts or the ratio of $\frac{\left(\sum_{i=1}^{N} Y_i\right)}{N}$ for the earliness penalty problem.

3. Although the relationship is not linear, the general trend is that as the percentage of conflict increases, the total earliness penalty incurred for a given problem also increases.

4. For the earliness penalty minimization problem, the total penalty cost incurred for a problem tends to increase as the average ratio of E/t increases.

D.1.2 Algorithms III and IV

1. The solutions produced by algorithms III and IV followed the same general trend relative to changes in percentage of job conflicts, the ratio of $\frac{\left(\sum_{i=1}^{N} Y_i\right)}{N}$, the ratio of $\frac{\left(\sum_{i=1}^{N} Z_i\right)}{N}$ and the ratio of $\frac{\left(\sum_{i=1}^{N} \frac{E_i}{W_i}\right)}{N}$ for the earliness and tardiness penalty problem. In general, the relative performance of the algorithms seems to be insensitive to the changes in percentage of job conflicts or the ratio of $\frac{\left(\sum_{i=1}^{N} Y_i\right)}{N}$ or the ratio of $\frac{\left(\sum_{i=1}^{N} Z_i\right)}{N}$ or the ratio of $\frac{\left(\sum_{i=1}^{N} \frac{E_i}{W_i}\right)}{N}$ for the earliness and tardiness penalty problem.

2. Although the relationship is not linear, the general trend is that as the percentage of conflict increases, the total sum of earliness and tardiness penalty incurred for a given problem also increases.

3. For the sum of weighted earliness and tardiness penalty minimization problem, the total penalty cost incurred for a problem tends to increase as the average ratio of E/t or W/t increases.

REFERENCES

[1] Abdul-Razaq, T.S., and Potts, C.N.,"Dynamic Programming State-Space Relaxation for Single-Machine Scheduling," *Journal of the Operational Research Society*, Vol. 39, No. 2, pp. 141-152 (1988).

[2] Arkin, E.M., and Roundy, R.O., "Weighted-Tardiness Scheduling on Parallel Machines with Proportional Weights," *Operations Research*, Vol. 30, No. 1, pp. 64-81 (1991).

[3] Asano, M., and Ohta, H., "Single Machine Scheduling Using Dominance Relation to Minimize Earliness Subject to Ready and Due Times," *International Journal of Production Economics*, Vol. 44, pp. 35-43 (1996).

[4] Bagchi, U., Sullivan, R.S., Chang, Y.L., "Minimize Mean Absolute Deviation of Completion Times about a Common Due Date," *Naval Research Logistics Quarterly*, Vol. 33, pp. 227-240 (1986).

[5] Baker, K., and Schrage, L., "Finding an Optimal Sequence by Dynamic Programming: an extension to precedence-related tasks," *Operations Research*, Vol. 26, pp. 111-120 (1978).

[6] Baker, K.R., and Scudder, G.D.,"Sequencing with Earliness and Tardiness Penalties: A Review," *Operations Research*, Vol. 38, No. 1, pp. 22-36 (1990).

[7] Chand, S., and Schneeberger, H.,"Single Machine Scheduling to Minimize Weighted Earliness Subject to No Tardy Jobs," *European journal of Operational Research*, Vol. 34, pp. 221-230 (1988).

[8] Chang, P.C., and Lee, H.C., "A Greedy Heuristic for Bicriterion Single Machine Scheduling Problems," *Computers and Industrial Engineering*, Vol. 22, No. 2, pp. 121-131 (1992).

[9] Chang, P.C., "A Branch and Bound Approach for Single Machine Scheduling with Earliness and Tardiness Penalties," *Computers and Mathematics with Applications*, Vol. 37, pp. 133-144 (1999).

[10] Chen, J.F., and Wilhelm, W.E.,"An Evaluation for Allocating Components to Kits in Small Lot, Multi-Echelon Assembly Systems," *International Journal of Production Research*, Vol. 31, No. 12, pp. 2835-2856 (1993).

[11] Chen, J.F., and Wilhelm, W.E.,"Optimizing the Allocating of Components to Kits in Small Lot, Multi-Echelon Assembly Systems," *Naval Research Logistics*, Vol. 41, pp. 229-256 (1994).

[12] Chhajed, D.,"A Fixed Interval Due-Date Scheduling Problem with Earliness and Due-Date Costs," *European Journal of Operational Research*, Vol. 84, pp. 385-401 (1995).

[13] Christofides, N., Mingozzi, A., Toth, P.,"State-Space Relaxation Procedures for the Computation of Bounds to Routing Problems," *Networks*, Vol. 11, pp. 145-164 (1981).

[14] Davis, J.S., and Kanet, J.J.,"Single-Machine Scheduling with Early and Tardy Completion Costs," *Naval Research Logistics*, Vol. 40, pp. 85-101 (1993).

[15] Doctor, S.R., Cavalier, T.M., and Egbelu, P.J.,"Scheduling for Machining and Assembly in a Job-Shop Environment," *International Journal of Production Research*, Vol. 31, No. 6, pp. 1275-1297 (1993).

[16] Fisher, M.L., "A Dual Algorithm for the One-Machine Scheduling Problem," *Mathematical Programming*, Vol. 11, pp 229-251 (1976).

[17] Fry, T.D., Armstrong, R.D., and Blackstone, J.H.,"Minimizing Weighted Absolute Deviation in Single Machine Scheduling," *IIE Transactions*, Vol. 19 No. 4, pp. 445-450 (1987).

[18] Fry, T.D., Armstrong, R.D., and Rosen, L.D., "Single Machine Scheduling to Minimize Mean Absolute Lateness: A Heuristic Solution," *Computers & Operations Research*, Vol. 17, No. 1, pp. 105-112 (1990).

[19] Garey, M., Tarjan, R., and Wilfong, G.," One-Processor Scheduling with Symmetric Earliness and Tardiness Penalties," *Mathematics of Operations Research*, Vol. 13, pp. 330-348 (1988).

[20] Hall, N.G., and Posner, M.E.,"Eariiness-Tardiness Scheduling Problems, I: Weighted Deviation of Completion Times about a Common Due Date," *Operations Research*, Vol. 39, No. 5, pp.836-846 (1991).

[21] Hall, N.G., Kubiak, W., and Sethi, S.P.,"Earliness-Tardiness Scheduling Problems, II: Deviation of Completion Times about a Restrictive Common Due Date," *Operations Research*, Vol. 39, No. 5, pp.847-856 (1991).

[22] Kanet, J.,"Minimizing the Average Deviation of Job Completion Times about a Common Due Date," *Naval Research Logistics*, Vol. 28, pp. 643-651 (1981).

[23] Kao, E.P.C., and Queyranne, M.," On Dynamic Programming Methods for Assembly Line Balancing, *Operations Research*, Vol. 30, pp. 375-390 (1982).

[24] Kim, Y.D., and Yano, C.A., "Minimizing Mean Tardiness and Earliness in Single-Machine Scheduling Problems with Unequal Due Dates," *Naval Research Logistics*, Vol. 41, pp. 913-933 (1994).

[25] Lawler, L.E.,"A Pseudopolynomial Algorithm for Sequencing Jobs to Minimize Total Tardiness," *Annuals of Discrete Mathematics*, Vol. 1, pp. 331-342 (1977).

[26] Lenstra, J.K., Rinnooy Kan, A.H.G., and Brucker, P., "Complexity of Machine Scheduling Problems," *Annuals of Discrete Mathematics*, Vol. 1, pp. 343-362 (1977).

[27] McKoy, D.C., "Minimizing Production Flow Time in a Process and Assembly Job Shop," Ph.D. thesis, Department of Industrial and Manufacturing Engineering, Pennsylvania State University, university Park, Pennsylvania, USA (1995).

[28] McKoy, D.C., and Egbelu, P.J., "Minimizing Production Flow Time in a Process and Assembly Job Shop," *International Journal of Production Research*, Vol. 36, No. 8, pp. 2315-2332 (1998).

[29] McKoy, D.C., and Egbelu, P.J., "Production Scheduling in a Process and Assembly Job Shop," *Production Planning and Control*, Vol. 10, No.1 pp. 76-86 (1999).

[30] Nof, S.Y., Wilhelm, W.E., and Warnecke, H., *Industrial assembly*, Chapman & Hall, London, UK (1997).

[31] Ow, P.S., and Morton, T.E.,"The Single Machine Early/Tardy Problem," *Management Science*, Vol. 35, No. 2, pp. 177-191 (1989).

[32] Qi, X., and Tu, F.S.,"Scheduling a Single Machine to Minimize Earliness Penalties Subject to the SLK due-date Determination method," *European Journal of Operational Research*, Vol. 105, pp. 502-508 (1998).

[33] Ramudhin, A., and Marier, P.,"The Generalized Shifting Bottleneck Procedure," *European Journal of Operational Research*, Vol. 93, pp. 34-48 (1996).

[34] Smith, W.E., "Various Optimizers for Single Stage Production," *Naval Research Logistics Quarterly*, Vol. 3, No. 1, pp. 56-66 (1956).

[35] Sridharan, V., and Zhou, Z.,"A Decision Theory Based Scheduling Procedure for Single-machine Weighted Earliness and Tardiness Problems," *European Journal of Operational Research*, Vol. 94, pp. 292-301 (1996).

[36] Sundararaghavan, P., and Ahmed, M.,"Minimizing the Sum of Absolute Lateness in Single-Machine and Multimachine Scheduling," *Naval Research Logistics Quarterly*, Vol. 31, pp. 325-333 (1984).

[37] Sung, C.S., and Joo, U.G.,"A Single Machine Scheduling Problem with Earliness/Tardiness and Starting Time Penalties under a Common Due Date," *Computers & Operations Research*, Vol. 19, No, 8, pp. 757-766 (1992).

[38] Szwarc, W.,"Adjacent Orderings in Single-Machine Scheduling with Earliness and Tardiness Penalties," *Naval Research Logistics*, Vol. 40, pp. 229-243 (1993).

[39] Szwarc, W., and Liu, J.J.,"Weighted Tardiness Single Machine Scheduling with Proportional Weights," *Management Science*, Vol. 39, No. 5, pp. 626-632 (1993).

[40] Verma, S., and Dessouky, M.,"Singlie-Machine Scheduling of Unit-Time Jobs with Earliness and Tardiness Penalties," *Mathematics of Operations Research*, Vol. 23, No. 4, pp. 930-943 (1998).

[41] Yano, C.A., and Kim, Y.D.,"Algorithms for a Class of Single-Machine Weighted Tardiness and Earliness Problems," *European Journal of Operational Research*, Vol. 52, pp. 167-178 (1991).

ACKNOWLEGMENTS

I would like to thank the many people who have supported and helped me to complete this research. First I would like to express gratitude to my advisor Dr. Pius J. Egbelu for his support, guidance, and encouragement throughout the process of doing this work. This research would not have been possible without him. I would also like to thank my committee members: Dr. Timothy Van Voorhis who provided much help with formulating mathematical models, Dr. Douglas D. Gemmill, Dr. John Wacker, and Dr. W. Robert Stephenson for their review of this work and many helpful comments.

I would especially like to thank my roommate Nawapak Eua-anant, who not only suggested to me how to construct the data structure for the research, but also encouraged me with my graduate studies. I would also like to thank all of my friends for their support.

Finally, I would like to thank my parents, my brother and sisters for their love, support and encouragement.