Minimization of Weighted Tardiness in Job Shops Using Shifting Bottleneck and Tabu Search Procedures

Srinivasa Rao Bongarala

A Thesis

in

The Department

of

Mechanical Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

January 2000

0-612-47835-1

Canada

ABSTRACT

Minimization of Weighted Tardiness in Job Shops Using Shifting Bottleneck and Tabu Search Procedures

Srinivasa Rao Bongarala

Scheduling to meet set due dates is one of the most critical issues in modern production systems. One measure of schedule effectiveness in the presence of due dates is weighted tardiness since it can capture the cost of contractual penalties and other losses to an organization. This research work concentrates on the problem of minimizing the total weighted tardiness in classical job shops.

Job shop scheduling problems are among the hardest known combinatorial optimization problems. In particular the problem of minimizing tardiness in job shops is strongly NP-hard, which makes finding optimal solutions to it impractical in most realistic situations. We approach this problem using two well known heuristics, namely the shifting bottleneck procedure and tabu search. Both heuristics are known to perform very well for minimizing makespan in job shops. Here we adapt them for the objective of minimizing weighted tardiness and test both under different parameter settings in order to select implementations that give the best results for a given computational effort.

We test our algorithms with problem instances taken from the literature and with randomly generated instances. We present our observations and conclusions regarding the relative performance of these heuristics and their performance in comparison with dispatching rules. In particular, we find that the shifting bottleneck procedure outperforms tabu search for this problem.

TO MY PARENTS . . . Satyavathi, Satyanarayana Rao

ACKNOWLEDGEMENTS

I would like to express my gratitude and appreciation to my thesis supervisor, Dr. Samir Amiouny for his continuous support and guidance during all stages of this thesis. His care and encouregement has enabled me to achieve this educational step. I would like to thank the faculty, staff and fellow students of the Mechanical Engineering Department at Concordia University, for their cooperation.

I would like to thank my roommates Bhanu, Vasu, Ravi, Prasad and my friend Roopa Shekar for their continuous encouragement through out this work. I would like to thank Mr. Harikirat Singh Sahambi and Ramaraju for their help in documenting this thesis.

This thesis is dedicated to my parents for their love and encouragement to achieve excellence, without them this would not have been possible. I also wish to thank my sisters Santoshi and Anu for continuously pestering me to specify the date of my Master's graduation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

i	Job index
j	Machine index
d	Common due date when all jobs have same due date
d_i	Due date of operation with respect to job i
r_i	Release time of operation with respect to job i
p_{ij}	Processing time of job i on machine j
d_{ij}	Due date of job i on machine j
r_{ij}	Release time of job i on machine j
T_{ij}^k	Tardiness of job k due to the operation i on machine j
T_i	Tardiness of job i
w_i	Weight of job i
$\sum w_i T_i$	Total weighted tardiness of all jobs
$C_{\max}$	Makespan or total completion time
f	Due date tightness factor
S	Schedule
D_s	Disjunctive graph
N	Node set
A	Conjunctive arcs in the graph
B	Disjunctive arcs in the graph
α	Machine environment
β	Job characterstics
γ	Optimality criteria
n	Number of jobs
O	Order of complexity in time
p	Function of processing time

Chapter 1

Introduction

Scheduling consists of planning and prioritizing activities that need to be performed in an orderly sequence of operations. It is a tool that optimizes the use of available resources. Scheduling leads to increased efficiency and capacity utilization, reducing the time required to complete jobs and consequently increasing the profitability of an organization [65].

Typical scheduling problems are railway time-tabling, project scheduling, production scheduling, hydro-power scheduling, scheduling nurse shifts in a hospital, etc. Emerging application examples of scheduling are in flexible manufacturing systems, multiprocessor scheduling, robot activity scheduling, scheduling in very large scale networks and real-time scheduling. Furthermore, there are a number of related problems belonging to the larger class of planning problems, e.g., the early stage of project management. Even this small sample shows the wide range and diversity of scheduling problems, and their importance in almost every sphere of human life stems from the need to utilize limited resources as efficiently as possible, simultaneously satisfying several domain-specific constraints [41].

In this thesis we deal with short range scheduling in a machine shop environment. Scheduling plays an important role in shop floor planning. A schedule shows the planned time when processing of a specific job will start on each machine that the job requires. It also indicates when the job will be completed on every machine. Thus it is a timetable for both the jobs and machines. Most of the scheduling problems in production planning are very complex and far from being completely solved, because of their combinatorial nature [1].

1.1 Scheduling models

In production planning terminology, scheduling models may be classified in terms of a three field classification $\alpha \mid \beta \mid \gamma$ introduced by Graham et al. [19] where α field specifies the machine environment and contains a single entry; the β field specifies the job characteristics, constraints and may contain no entries, a single entry or multiple entries; and the γ field denotes the optimality criterion. The classification given below is adapted from Pinedo [50].

1.1.1 Machine environment

The following are some of the possible machine environments specified in the α field

- Single machine ($1||$): There is only one machine (server) available and arriving jobs (work) require services from this machine. Jobs are processed by the machine one at a time. Each job has a processing time and a due date or an ideal completion time and may have other characteristics such as priority. There may also be a penalty function for deviating from the due date [65].

- Identical machines in parallel ($P_m||$): There are m identical machines in parallel. Job j requires a single operation and may be processed on any one of the m machines or on any one belonging to a given subset. If job j is not allowed to be processed on just any one, but rather some one belonging to a given subset, say subset M_j, then the entry M_j appears in the β field.

- Flow shop ($F_s||$): Jobs are processed on multiple machines in an identical sequence. However the processing times of the jobs on each machine may be different.

- Job shop ($J_m||$): This is one of the widely used generalized production systems. There are different machines in the shop, and a job may require some or all of these machines in some specific sequence, the only restriction being that a job can not use the same machine more than once.

- Open shop ($O||$): An open shop is similar to job shop except that a job may be processed on the machines in any sequence the job needs. In other words, there is no operationally dependent sequence that a job must follow.

1.1.2 Job characteristics

The processing restrictions and constraints specified in the β field may include multiple entries. Here are some of the most common job characteristics that can be observed on the shop floor.

- Release dates ($|r_j|$): If this symbol is present in the β field, job j may not start its processing before its release date r_j. If r_j does not appear in the β field, the processing of job j may start at any time. In contrast to the release dates, due dates are not specified in this field. The type of objective function gives sufficient indication whether there are due dates or not.

- Sequence dependent setup times ($|s_{ij}|$): The S_{jk} represents the sequence dependent setup time between jobs j and k; S_{ok} denotes the setup time for job k if job k is first in the sequence and S_{jo} the clean-up time after job j if job j is the last in the sequence (of course, S_{ok} and S_{jo} may be zero). If the setup time between jobs j and k depends on the machine, then the subscript i is included, that is, S_{ijk}. If no S_{jk} appears in the β field, all setup times are assumed to be zero or sequence independent, in which case they can simply be added to the processing time.

- Preemption ($|prmp|$): This characteristic indicates whether preemption (or job splitting) is allowed. Preemption of a job or an operation means that processing may be interrupted and resumed at a later time, even on another machine. A job or operation may be interrupted several times. If preemption is allowed, the β field shows *prmp*.

- Precedence constraints ($|prec|$): Precedence constraints may appear in single machine or in parallel machines environments, requiring that one or more jobs may have to be completed before another is allowed to start its processing. There are several special forms of precedence constraints. If each job has at most one predecessor and one successor, the constraints are referred to as chains. If each job has at most one successor, the constraints are referred to as *intree*. If each job has at most one predecessor, the constraints are referred to as *outtree*.

- Blocking ($|block|$): Blocking is a phenomenon that occur in flow shops. If a flow shop has a limited buffer between two successive machines, it may happen that when a buffer is full the upstream machine is not allowed to release a completed job. The completed job has to remain on the upstream machine preventing or blocking that machine from working on another job.

1.1.3 Optimality criteria

Generally the performance of any schedule can be evaluated with respect to some objective function to be optimized. Below are some of the most common objectives used to evaluate the performance of any schedule as given by Pentico and Thompson [24].

- Makespan ($||C_{\max}$): This is the most widely used performance measuring parameter of schedules. It is the total time required to complete all the jobs in a schedule.

- Maximum weighted flow time ($||F_{wt}$): Weighted flow time is the sum over all jobs of the weighted amount of time between a job arrival into and departure from the system. Minimizing maximum weighted flow time can be accomplished by setting the due dates equal to the arrival times and minimizing maximum lateness.

- Total weighted tardiness ($||\sum w_j T_j$): Weighted tardiness is the sum over all jobs of the weighted length of time taken to complete job after its due-date. Weighted tardiness takes into account only the positive difference between completion time and due date.

- Maximum weighted lateness ($||L_{\max}$): Minimizing maximum lateness also has significant importance because it can be used as a tool for solving many other complex shop floor problems.

In some cases more than one objective are relevant to a particular problem. The importance of multiple objective functions arises from the fact that a single objective can be optimized at the expense of others. A widely used technique of defining multiple objective functions is by expressing single objectives into penalty and minimizing the total penalty [77]. The researchers who studied these types of multiple objectives include Yano and Kim [76], Davis and Kanet [37], Herrman and Kim [74]. The weighted early/tardy and weighted early/tardy flow time fall under this category.

All the above mentioned scheduling models could fall under the following categories:

- Static: A static problem is one where the number of all jobs, their release times and all their characteristics are known and fixed before scheduling decisions are taken [70].

- Dynamic: A dynamic environment is one in which the jobs arrive randomly over a period of time.
- Deterministic: In deterministic models, all the parameters of the problem, such as arrival times, due dates, processing times and machine availability times are known before starting scheduling [55].
- Stochastic: In stochastic models, at least one of the above mentioned problem parameters includes stochastic factors [55].

1.2 Solution strategies

Generally production scheduling problems are difficult to solve. Most of the problems are NP-hard and only a few problems were solved to optimality. Various optimization and approximation algorithms were developed for these problems.

The optimization algorithms were mainly based on the branch and bound scheme as developed by Lagewey et al. [40], Carlier and Pinson [35], Applegate and Cook [9]. While considerable progress has been made in this approach, practitioners still find such algorithms unattractive. They are time consuming, and the size of the problems which can be solved with in a reasonable time limit is small (up to 100 operations). Moreover their implementation demands a high level of programmer sophistication.

On the other hand approximation algorithms typically require only a fraction of the time needed for optimization procedures, but do not guarantee optimal solutions. The most common approximation algorithms for scheduling problems are:

Dispatching or priority rules These are rules used for choosing the next job to be dispatched (scheduled) on a machine according to some fixed priority rules. These procedures are very fast, but the quality of the solution that they produce usually leaves plenty of room for improvement [14].

Local search procedure Local search is an iterative procedure which moves from one solution in search space to another as long as necessary. In order to systematically search through the solution space, the possible moves from a solution to the next solution are restricted in some way. To describe such restrictions neighborhood structure is used, which holds the subset of solutions that can be reached in one step by moving from the original solution. The search is continued until it finds the best move in the neighborhood, which is only a local optima with respect to the neighborhood [56].

Tabu search Tabu search is a meta-heuristic search procedure designed to explore the solution space beyond local optimality, it was initially proposed by Fred Glover [28]. This search algorithm keeps a "tabu" list of recently made moves and selects in each iteration the best solution that is the neighborhood of the current solution but not in the tabu list, even if that sequence results in an increase in the objective function. This allows for diversification in the search and may prevent it from being trapped at a local optima [77].

Simulated Annealing This is a procedure to solve large combinatorial problems that works in a way similar to the physical annealing process of solids. Solutions in a combinatorial problem are equivalent to the states of a physical system, and the cost of a solution is equivalent to the energy of a state. In the searching process, simulated annealing accepts not only better but also worse neighboring solutions with a certain probability. At the beginning, the probability of accepting a worse solution is larger at higher temperatures and decreases gradually as the search progresses [79].

Genetic algorithms This can refer to any search process simulating the natural evolutionary process. There will be a current population of possible solutions to the problem. In each generation, the best solutions (most fit individuals) are allowed to produce new solutions (children) by mixing the features of the parents (or by mutation); the worst children die off to keep the population stable. This process is repeated iteratively until some stopping criterion is reached [24].

Shifting bottleneck procedure This method is used in a multiple machine environment and works by repeatedly optimizing the sequence on each individual machine, while keeping the sequences on all other machines fixed. If pursued until no more improvement can be obtained, this procedures yields a local optimum over the neighborhood schedules obtainable from a given one by changing on any single machine [14].

1.3 Scope of research

This thesis concentrates on static, deterministic job shop scheduling problems where the objective is to minimize the total weighted tardiness. In particular we deal with problems where jobs have possibly non-zero release times and possibly different due dates. This problem can be represented by the notation $J_m|r_j|\sum w_i T_i$. Tardiness problems are of great importance in manufacturing systems. Whenever a job is not completed by its due date, certain costs are incurred. These costs include: penalty clauses in the contract, if there are any; loss of goodwill resulting in an increased probability of losing customers for some or all future jobs, and a damaged reputation which will turn other customers away [8].

Some of the practical problems that fall into general job shop structure are: scheduling of different programs on a computer; the processing of different batches of crude oil at a refinery; the repair of cars in a garage; the manufacture of paints of different colours [70].

The job shop tardiness problem is the generalization of the single machine tardiness problem $1||\sum w_jT_j$, which is known to be strongly NP-hard [45]. Hence we approach the problem with approximate heuristics instead of exact algorithms. We adapt and implement two well known heuristic procedures for the above job shop tardiness problem. We present our computational experience using these heuristics.

Chapter 2

Literature Review

The problem of completing a job as close as possible to a promised delivery date is of primary importance to operations managers. As a result, research into scheduling jobs for processing, under a wide variety of shop conditions has long occupied a prominent place in production planning literature. Some of the various objectives on which extensive work was reported include minimizing the makespan or total completion time [3] [25], maximum lateness [32], earliness and tardiness penalties [72], mean flow time [39] etc. A vast research was reported towards the single machine tardiness problems, but little work was done for solving the tardiness problem in job shops. Here we review the literature on both single machine and job shop tardiness problems and recent extensions to them.

2.1 Single machine tardiness problems

Single machine problems are important for various reasons. A single machine environment is simple and a special case of all other environments. The results that can be obtained for single machine models not only provide insights into the single machine environment, but also provide a good basis for heuristics for more complicated machine environments. In practice, scheduling problems of more complex machine environments are often decomposed into sub problems that deal with single machines [50].

2.1.1 Distinct due date

The single machine tardiness problem with distinct due dates and identical release time was first studied by Wilkerson and Irwin [44]. They proposed one of the first heuristic procedures to solve problem using an adjacent pairwise comparison strategy. Later Baker [61] suggested a dynamic programming based algorithm for jobs having precedence constraints.

Lawler [47] proposed a pseudo-polynomial dynamic algorithm to this problem, but it requires $O(n)$ steps and $O(n^4p)$ space although for most of the problems only a small portion of this storage requirement is needed. In 1978, he gave a fully polynomial approximation scheme for the total tardiness problem. Baker and Schrage [62] presented a chain algorithm in terms of solution time for both weighted and unweighted tardiness problems. Later in the year they devised a labeling procedure that can uniquely address each of the sequences that are generated while avoiding the necessity for decoding. This labeling procedure assigns a unique code to a sequence, identifies its location within an array and enables the sequence to be directly retrieved from the storage.

Potts and Van Wassenhowe [58] introduced a decomposition based algorithm for the problem. Three years later, they introduced a branch and bound algorithm using lagrangian relaxation and a multiplier adjustment method to compute and update a lower bound. Their method uses a heuristic to form an initial sequence and then chooses the multipliers so that the heuristic solution also solves the lagrangian problem.

In the same year Morton and Rachamudugu [22] identified a new property for optimally sequencing adjacent jobs. This property, which they called proposition 1 states that for any two adjacent jobs J_j and J_k, J_j must precede J_k in an optimal sequence.

Fry et al. [10] developed a very simple but effective algorithm based on adjacent pairwise interchange (API) methodology to solve the problem. The idea behind their algorithm was to explore a solution space by creating a new sequence with the hope that it may result in a local, although not necessarily global optimum. Du and Leung [38] proved that single machine tardiness problems are NP-hard.

Potts and Van Wassenhowe [59] proposed a simulated annealing based algorithm for the problem. They generated the neighborhood using all pair interchanges which makes their neighborhood size $n(n-1)/2$. Chambers et al. [36] proposed a heuristic which uses the dominance properties and problem decomposition to quickly solve single machine problems with up to 50 jobs.

Holsenbeck and Russel [21] developed a heuristic based on emmons corollary. Their principle stated that in EDD (Earliest Due Date) sequence, a job should be assigned to the last position if it possesses a tardiness less than or equal to its processing time.

Adams et al. [2] proposed an extremely simple construction heuristic which was based on recursive identification of the most promising job, which is then scheduled in the last open position. They claimed that their heuristic performs significantly better than the one proposed by Wilkerson and Irwin [44] which is increasingly better as the problem size increases. Later in the same year Suan et al. [46] developed an efficient algorithm based on the branch and bound technique respecting the precedence relations between the jobs. They designed an experiment to test the efficiency of their algorithm and the branch and bound procedure. But their computational experience was reported to be limited to problems with only 35 jobs.

As an improvement of the famous Lawler [47] decomposition theorem for the one-machine total tardiness problem, some conditions on decompositions were obtained by Potts and Wassenhowe [58] and were used by them to make the decomposition algorithm more efficient. More conditions on the left-most decomposition

position were proved and tested by Chang et al. [68].

Islam and Eksioglu [4] presented a tabu search approach for solving the problem. They showed that their tabu search heuristic out performs the heuristics given by Fry et al. [10], Potts and Van Larhooven [58], and Holsenbeck and Russel [20].

Holsenbeck et al. [20] proposed a method of modifying due dates that ease the construction of optimal schedules. They employed due date modification in a new heuristic which was very fast and capable of solving a 50 job problem in a very short computational time. Their heuristic was shown to be superior to previously known heuristics in minimizing total weighted tardiness.

2.1.2 Common due date

The single machine common due date problem is a scenario where all the jobs have a common due date d. Lawler and Moore [48] have presented a pseudo-polynomial dynamic programming algorithm for determining the optimal solution whose computation time is bounded by a polynomial of order n^2d. Arthnari [66] gave a branch and bound algorithm for the problem. Lenstra et al. [45] proved even if the jobs have equal processing times and different penalties, the problem still remains NP-hard. Rachamadugu et al. [23] presented myopic heuristic for the problem when all the jobs have equal processing times.

Bector et al. [16] proposed a linear programming algorithm. Starting with an arbitrary sequence they related the problem to a generalized linear program from which some basic results are proved using elementary properties of linear equations and a linear programming problem. Using those results and the idea of sensitivity analysis in linear programming, an algorithm was developed that determines the optimal due date and the corresponding optimal sequence.

Fathi et al. [75] reported three simple heuristics and shown to have arbitrarily bad worst case performance. A fourth heuristic was then proposed and shown to have a worst-case performance bound of 2. Cheng [17] proposed a partial search algorithm which was not polynomial bound, which represented a significant improvement in computational efficiency of the order $O(n^4)$ over that presented in Cheng [17] which was of order $O(O^2 2^n)$ time complexity. He provided a systematic method of solution for the single machine common due date assignment and sequencing problem.

Liman et al. [11] considered the problem in which all jobs have a common due time window. Jobs that are completed within the window incur no penalty. They proposed an $O(n \log n)$ algorithm to solve the problem. They also considered two special cases for which simple solutions can be obtained. Alidaee et al. [5] studied the problem and assumed that the weights of the jobs are proportional to their processing times . The longest processing time (LPT) order was shown to be optimal and an efficient algorithm for generating an optimal schedule was proposed.

2.1.3 Release dates

The single machine tardiness problem with release dates for each job was proved to be strongly NP-hard by RinnoyKan [30], since the release times are unequal and idle time may be inserted in optimal schedule. Chu and Portmann [7] proved a sufficient condition for local optimality in solving the problem. They defined a dominant subset of schedules on the basis of this condition and proposed several new approximate algorithms to construct schedules belonging to this subset. They tested their heuristics against the modified due date (MDD) heuristic of Baker and Bertrand [64] and concluded that their heuristics offered an improvement of 10% over the MDD heuristic.

Chu [6] developed a branch and bound algorithm for solving $1/r_j/T_j$ optimally. Lower bounds were obtained by scheduling the jobs according to shortest processing time under the assumption that the jobs are preemptive. Upper bounds for both partial and complete schedules were obtained by applying the heuristics of Chu and Portmann [7] to the appropriate subset of jobs. Their algorithms were successful in easy problems with up to 230 jobs and in hard problems with up to 30 jobs. Problem hardness was defined by the tightness of due dates and by the range of release times.

2.2 Job shop problems

The classical general job-shop scheduling problem is defined as follows: There are n jobs to be processed through m machines. Each job must pass through each machine exactly once. The processing of a job on a machine is called an operation and requires a duration called the processing time. Technological constraints demand that each job should be processed through machines in a specific order. Each job has a release time and a deadline. The general problem is to find a sequence in which jobs pass between the machines which is compatible with the technological constraints and optimal with respect to some performance criterion [41]. Various objectives such as minimizing makespan, minimizing total tardiness etc., [9] can be considered.

Although single machine tardiness problems are well studied, little was reported on tardiness in job shop scheduling. The dominance conditions and bounding mechanisms developed for single machines can not be easily extended to job shops [54]. We present here some of the work done by various researchers towards the tardiness problems in job shops.

Baker and Kanet [63] extended the idea of modified due date rule to the general job shop problem. They used the operation version of the modified operation due date (MOD) rule which employs operation due dates to pace the jobs in the shop. Again Baker [60] examined the interaction between sequencing priorities and the method of assigning due dates, primarily focusing on the average tardiness as a measure of scheduling effectiveness. He performed certain simulation experiments which illuminates how these factors interact with dispatching rules and his experimental results suggest which combinations are most effective in a scheduling system.

Vepsalainen and Morton [43] studied a number of dispatching rules, which were heuristics that assign priorities to those operations that have not been processed yet, and then schedule them in decreasing order of priority. They have tested a number of rules such as Earliest Due Date (EDD), Weighted Shortest Processing Time (WSPT), Cost Over Time (COVERT) and Apparent Tardiness Cost (ATC) and concluded that apparent tardiness rule achieves the best results.

Anderson and Nyirendra [34] presented two new dispatching rules to minimize tardiness in the job shops. Both rules are closely related to the modified operation due date (MOD) rule. The first is a combination of the shortest processing time (SPT) rule and critical ratio (CR) rule, And the second is a combination of SPT and slack per remaining work (S/RPT) rules. They reported that the performance of these two new rules were better than that of other rules that were developed till date, in order to minimize the total weighted and unweighted tardiness. Further more, these two rules were effective in minimizing the number of tardy jobs.

Raman and Talbot [54], proposed a new heuristic approach that decomposes the dynamic problem into a series of static problems. These static problems were solved to optimality and then implemented dynamically on a rolling horizon basis. They presented a specific heuristic that constructs the schedule for the entire system by focusing on the bottleneck machine. Their computational results indicate that significant due date performance improvement over traditional dispatching rules can be obtained by using the proposed approach.

Deal et al. [80] proposed a multi-pass heuristic algorithm considering the due dates, where the objective was to minimize the total job tardiness. Their algorithms operation was carried out into two phases. In phase 1, a dispatching rule is employed to generate an active or non delay initial schedule. In phase 2, tasks selected from a predetermined set of promising target operations in the initial schedule are tested to ascertain whether by left shifting their start times and re arranging some subset of the remaining operations one can reduce tardiness in the job shop. Later Tang, He, and Cho developed an efficient heuristic algorithm and named it as revised exchange heuristic algorithm (REHA). They have also shown that the algorithm can be completed in polynomial time. Results, generated over a range of shop sizes with different due date tightness levels, indicated that the proposed algorithm was capable of yielding notable reductions in total tardiness for practical size problems [78].

Singer and Pinedo [51] presented and compared a number of branch and bound algorithms for minimizing the total weighted tardiness in job shops. Basically their branching schemes were of two types. The first one inserts the operations in a partial schedule, while the second one fixes the arcs in the disjunctive graph formulation of the problem. Their bounding schemes were based on the analysis of precedence constraints, and on the solution of nonpremtive single machine subproblems that are subjected to so-called delayed precedence constraints. They obtained optimal solutions for all the instances with ten jobs and ten machines that they considered, including three tardiness versions of a well-known 10×10 instance introduced by Muth and Thompson [29].

Byeon, Wu and Storer [67] proposed a heuristic based on a graph theoretic decomposition for job shop total weighted tardiness problem. Their heuristic assigns the operations of a job scheduling problem into a series of subsets by solving the variant of the generalized assignment problem. The assignment problem imposes additional precedence constraints on the scheduling graph, defining partial schedule. This partial schedule preserves global perspective of system objectives over the planning horizon while retaining local flexibility. Later, they developed another decomposition scheme. The only difference is the variant assignment problem was solved using a branch and bound algorithm. Although their algorithm shows superior results compared to various traditional methods, it was limited to smaller problems.

Pinedo and Singer [52] presented a shifting bottleneck heuristic for the tardiness problem in job shops. This method decomposes the job shop into a number of single machine sub problems that are solved one after the other. Each machine is scheduled according to the solution of its corresponding sub problem. The order in which the single machine sub problems are solved has significant impact on the quality of the overall solution and on the time required to obtain the solution. They claim that their heuristic yields solutions that are close to optimal.

2.3 Purpose and outline of research

The above research survey indicates that little work was reported on tardiness problems in general job shops. Job shop scheduling is among the hardest combinatorial optimization problems. The difficulty of this problem may be illustrated by the fact that the optimal solution of an instance with 10 jobs and 10 machines, proposed by Fisher and Thompson [29] was not found until 20 years after the problem was introduced. Most of the heuristic job shop scheduling procedures described in the literature are based on the priority dispatching rules. These are one pass procedures of the greedy type, in that they construct a solution through a sequence of decisions based on what seems locally best, and decisions once made are final.

In many situations this is all that is needed, and so their need is justified. However, with the rapid increase in the speed of computing and growing need for efficiency in scheduling, it becomes increasingly important to explore the ways of obtaining better schedules at some extra computational cost. It was observed that the most successful heuristics of the job shops with makespan objective are shifting bottleneck and tabu search heuristics. Hence we focus on adapting these heuristics to job shop problems with total weighted tardiness minimization, as objective and we compare their performance in computational testing.

We start with the shifting bottleneck heuristic, where we decompose the job shop problem into a sequence of single machine problems. These single machine problems are solved separately using the tabu search heuristic. We implement and test various bottleneck selection schemes and re-optimization procedures. In chapter 3 we present a brief overview of shifting bottleneck heuristic, its components and the way we adapted it to our job shop problem.

The second procedure we adapt is tabu search for the overall job shop. In chapter 4, we first introduce the various concepts of the tabu search methodology. As we solved the overall job shop problem and the single machine optimization of shifting bottleneck using this tabu search technique, we felt it appropriate to present the tabu structural parameters of both in the same chapter. To our knowledge this is the first time that the tabu search is being applied to job shop problems with total weighted tardiness as objective and having possibly different due dates for the jobs.

Later we perform a preliminary testing on our algorithms with a few problems instances taken from the literature. Various structural parameters of both the procedures are tuned with different settings. We carry out our final testing with randomly generated test instances with the best settings obtained in our initial testing. The experimental set up, computational results and comparisons are tabulated in chapter 5. Finally we present our observations and conclusions.

Chapter 3

Shifting Bottleneck Procedure

In this chapter, we apply the well known shifting bottleneck procedure of Adams et al. [33] to our job shop weighted tardiness problem. This approach decomposes the job shop problem into a sequence of single machine problems and solves each machine one at a time. At each iteration, a critical subproblem is identified and solved. We demonstrate various bottleneck or critical subproblem selection schemes and re-optimization procedures.

3.1 Overview of shifting bottleneck procedure

Modern engineering workstations and the implementation of shop-floor information systems which track job and machine status in real time have made scheduling systems which consider the status of the entire shop, or at least majority of it , a practical possibility. Hence there is a need to focus on heuristics which exploit this type of global shop information to develop improved schedules at the cost of increased computation times [18]. One such approach used to accomplish the above goal is the Shifting Bottleneck (SB) heuristic.

The Shifting Bottleneck (SB) procedure is a heuristic decomposition approach which decomposes the job shop scheduling problem into single machine subproblems. At each iteration, a critical subproblem is identified and solved, and the subproblems solved up to that point are re-optimized based on this new information. Interactions between the subproblems are captured using the disjunctive graph representation of the job shop problem, which forms the core of shifting bottleneck heuristic for the job shop problems. Hence we describe this representation before proceeding with the method itself. Several authors extended the basic disjunctive graph representation to other implementations to model sequence-dependent setup times, assembly-type routings, parallel machine workcenters, batch processing machines and different production and transfer batch sizes [49] [13] [42] [57].

3.2 Disjunctive graph representation

It is convenient for the analysis to represent job shop problem using a graph. For disjunctive representation and formulation of the problem, we adopt the idea given by Singer and Pinedo [51]. Initially the basic disjunctive graph was designed for the problems with makespan objective. The problem of minimizing the makespan $C_{\max} = \max(C_1 \ldots, C_n)$ in a job shop is represented by a disjunctive graph $G = (N, A, B)$ by Balas [12]. With slight modifications, this representation was adapted by Singer and Pinedo [51] to the problem $J_m|r_j|\sum w_jT_j$. The set of nodes N contains one element for each operation (i,j), one source node U representing the start of the schedule and n sink nodes V_j representing the end of each job. The set of conjunctive arcs $A = (i,j) \rightarrow (k,j)$ consists of the arcs connecting the nodes that represent each pair of successive operations (i,j) and (k,j) of job j. Each arc $(i,j) \rightarrow (k,j)$ of length $|(i,j)(k,j)| = p_{ij}$ specifies that any operation (k,j) can be started at the earliest p_{ij} time units after the start of operation (i,j). The node that represents the final operation of job j, say (h,j), has an arc of length p_{hj} going to V_j. The source node U has n outgoing arcs going to the first operations of the n jobs.

Job	w_j	r_j	d_j	**Machine sequence**	**Processing times**
1	3	0	25	1,2,3,4	$p_{11} = 10$, $p_{21} = 3$, $p_{31} = 6$, $p_{41} = 4$
2	2	7	25	2,1,3	$p_{22} = 7$, $p_{12} = 8$, $p_{32} = 3$
3	1	8	21	4,1,2	$p_{43} = 5$, $p_{13} = 1$, $p_{23} = 7$
4	1	5	26	1,4,2,3	$p_{14} = 2$, $p_{44} = 4$, $p_{24} = 5$, $p_{34} = 6$

Table 3.1: 4 $\times$ 4 problem instance

Let N_i denote the set of nodes that correspond to the operations that have to be processed on machine i. The set of disjunctive arcs $B = (i,j) \leftrightarrow (i,k)$ has for every pair of nodes (i,j) and (i,k) in N_i, a pair of arcs $(i,j) \leftrightarrow (i,k)$ going in opposite directions with $|(i,j) \rightarrow (i,k)| = p_{ij}$ and $|(i,k) \rightarrow (i,j)| = p_{ik}$. The following disjunctive graph shows the above instance of $J_m|r_j|\sum w_jT_j$ problem with four jobs and four machines (4 $\times$ 4) as shown in Table 3.1. Only the disjunctive arcs that correspond to machine 2 were depicted for clarity. Let $\sigma(B)$ denote a selection of disjunctive arcs from B. Any solution for the job shop problem is equivalent to a selection $\sigma(B)$, as long as the selection $\sigma(B)$ has one and only one arc from every pair $(i,j)(i,k)$, and the resulting graph $G(N, A, \sigma(B))$ is not cyclic. Conversely, any selection $\sigma(B)$ satisfying the above properties corresponds to a feasible schedule. Let $L(v, v')$ denote the length of the critical path (longest path) from node v to node v' in graph $G(N, A, \sigma(B))$ (if there is no path, then $L(v, v')$ is not defined). The completion time C_j of job j is equal to $L(U, V_j)$ where values $L(U, v), v$ are any nodes.

The shifting bottleneck heuristic, which solves the job shop problem one machine at a time, involves the following steps:

- Subproblem formulation and optimization.
- Bottleneck selection.
- Re-optimization.

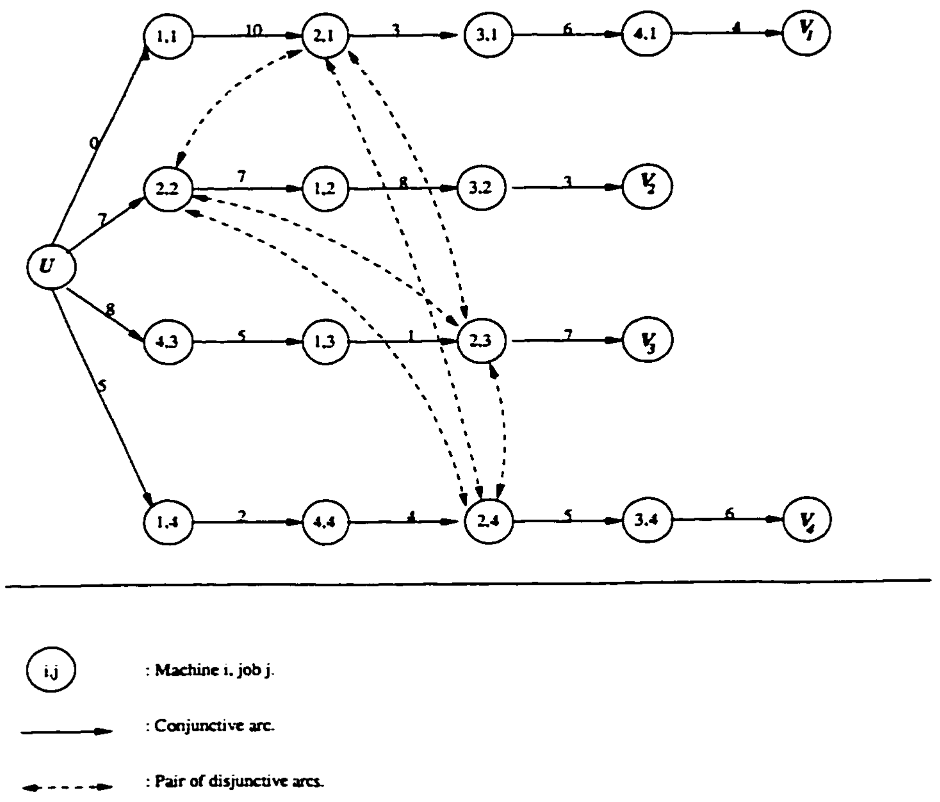

Figure 3.1: Disjunctive graph representation, the disjunctive arcs correspond to machine 2. Other disjunctive arcs not shown.

A solution for subproblem i corresponds to a sequence of operations for machine i, and the value of this solution reflects the total weighted tardiness cost for the overall job shop problem. The subproblem optimization step finds a sequence to this single machine problem which minimizes the objective function value when mapped onto the job shop problem. The bottleneck selection step selects the next machine to be scheduled from the unscheduled machines with respect to some measure of performance. Re-optimization step re-schedules some or all previously scheduled machines in order to adapt the schedule to the constraints imposed by the last scheduled machines. The above SB heuristic can be represented by the following flow chart in figure 3.2. The following sections discuss these steps in detail.

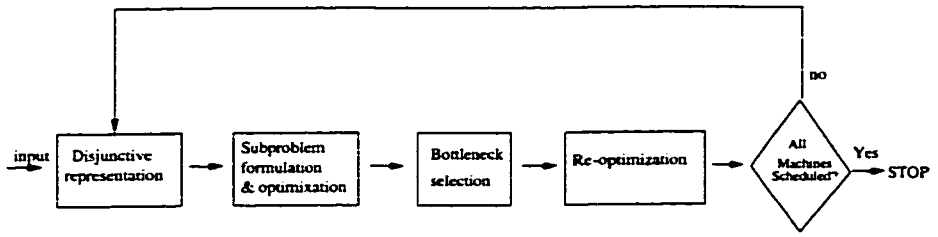

Figure 3.2: Flow chart of shifting bottleneck procedure

3.3 Subproblem formulation and optimization

The idea of formulating the single machine is similar to the idea of Singer and Pinedo [52]. The subproblem optimization step involves scheduling a single machine. However at intermediate iterations of the algorithm some machines have been scheduled and others not. This creates a need to consider the effects of scheduling decisions already made when scheduling as yet unscheduled machines. These effects can be seen in two ways: jobs become available for scheduling at the current machine at a certain time, and take a certain amount of time to complete their processing in the shop after leaving the current machine. For subproblem formulation, consider an operation (i,j) to be scheduled on machine i. Its earliest starting time is given by $r_{ij} = L(U,(i,j))$. A delay in the completion time of operation (i,j) may impact the tardiness of all n jobs, so we define d_{ij}^k, with $d_{ij}^k \geq 0$, as the local due date of operation (i,j) relative to job k:

$$d_{ij}^k = \begin{cases} \max(C_k, d_k) - L((i,j), V_k) + p_{ij} & \text{if } L((i,j), V_k) \text{ exists,} \\ \infty & \text{otherwise} \end{cases} \tag{3.3.1}$$

If operation (i,j) is completed after its local due date d_{ij}^k, i.e. $C_{ij} > d_{ij}^k$, then the tardiness of job k in the job shop increases by at least $C_{ij} - d_{ij}^k$. Let

$$T_{ij}^k = \max(C_{ij} - d_{ij}^k, 0) \tag{3.3.2}$$

denotes the tardiness of operation of (i,j) with respect to the due date of job k. Since all operations (i,j) assigned to machine i have to be scheduled, the tardiness of job k increases by at least

$$\max_{(i,j)\in N_i} T_{ij}^k \tag{3.3.3}$$

and the overall increase in the job shop objective function will be at least

$$\sum_{k=1}^{n} w_k (\max T_{ij}^k)_{(i,j)\in N_i} \tag{3.3.4}$$

Our aim is to find the sequence of operations on machine i that minimizes the above expression. This problem can be isolated from rest of the job shop by obtaining the release dates r_{ij}, processing times p_{ij} and due dates d_{ij}^k, $k = 1 \ldots n$, for each operation (i,j) in N_i. However the subproblem formulation to schedule the machines one at a time does not guarantee feasible solutions at intermediate iterations since cycles may introduced in the disjunctive digraph.

Other researchers [69] [15] showed that in addition to the release times and due dates given above, precedence constraints between the operations of jobs have to be followed to preserve feasibility. In order to avoid cycles, we follow the idea of delayed precedence constraints (DPC). These constraints capture precedence relationships among operations on machine i implied by the schedules of other machines that have already been scheduled.

The resulting single machine problem to be solved can be denoted by

$$1|r_j, prec|\sum_k w_k(\max_j T_j^k) \tag{3.3.5}$$

The experience of others in applying shifting bottleneck to makespan problems is the most critical part of the procedure. The problem in our case is NP-hard because optimal solutions can be found through enumeration techniques, these methods demand a tremendous amount of computational time and powerful computers. Instead we use tabu search, a meta-heuristic that is well known for these problems for finding the near optimal solutions, if not the optimal. Accordingly, approached subproblem optimization step with tabu search methodology. All the tabu structural elements considered for the single machine optimization are explained in the next chapter.

3.4 Bottleneck selection methods

The bottleneck selection procedure serves to specify which unscheduled machine will be scheduled next. There is more than one way in which a machine can be viewed as a bottleneck. Several authors considered different bottleneck selection methods according to their problem parameters. The most common approaches found in the literature are random machine selection and total workload measure.

In the first method, machines are selected in random order, with all the machines having equal probability of selection [31]. Some authors reported success using this method. Because of this fact, we considered testing this method during the early stages of our experimentation. However, our experience revealed quickly that the order of the machine is critical in the shifting bottleneck methodology. Hence we did not consider this method for our final testings. The other method to select the next machine to schedule relies on finding a relevant measure of total work load.

During the bottleneck selection process, we need a concept that expresses the bottleneck quality as a matter of degree rather than yes or no property. This quality could be measured, for instance, by the marginal utility of the machine in problems with makespan as objective [33]. As a measure of quality of machine k the value of an optimal solution to a certain one-machine scheduling problem on machine k, which requires more computational time. On the other hand selection of bottleneck machine quickly, allows us to perform intensive search on the single machine being selected.

We intend to test both the concepts of performing intensive search on all the unscheduled machines to select the bottleneck and quickly selecting the machine using the earliest available due date rule. In the first approach, we use tabu search on each unscheduled machine in order to determine a schedule that minimizes the weighted tardiness as defined in the problem formulation section. The machine with the highest objective value is selected as the next machine to schedule, since it has the largest effect on the overall schedule. If some machines result in the same weighted tardiness, we break ties using maximum weighted lateness. This is specially important at the start when all or most machines are unscheduled and, therefore, weighted tardiness of 0 is easy to achieve. Breaking ties by weighted lateness promises to reduce weighted tardiness later on in successive scheduling of the remaining machines.

In the second approach of bottleneck selection we use the same idea of evaluating machines based on weighted tardiness and breaking ties with maximum weighted lateness. But before doing this we schedule each machine using earliest available due date (EADD). According to earliest available due date dispatch rule every time as the machine becomes idle, the next selected operation is the one with earliest due date among the operations waiting. The performance variations in opting the two bottleneck selection methods will be discussed in next chapter.

3.5 Re-optimization procedures

An important component of the SB procedure is the re-optimization procedure. For each scheduled machine, the optimal sequence is found while keeping the sequences on all machines that are already scheduled fixed. If an improvement is found after this step, the process is repeated. We intend to test various re-optimization procedures taken from the literature and identify which ones perform better with our implementations.

- Re-optimizing the machines, once all the machines are scheduled.
- Re-optimizing by random selection after all the machines are scheduled.
- Re-optimizing all the previously scheduled machines, immediately after a machine is scheduled.
- Re-optimizing only those machines, whose operations lie on the critical paths immediately after scheduling a machine.

Our initial testing indicated that the re-optimizing after all machines scheduled and random machine re-optimization selection criteria were inferior to the other two re-optimization methods. These two choices, unlike the other two methods do not re-optimize the partial schedule in the light of the schedule on the most recently added machine. This could be the possible reason why they were found weaker than the later methods. The above contradictory argument can be extended to justify our selection of the third re-optimization procedure i.e., re-optimizing all the machines immediately after a machine is scheduled. The last procedure of re-optimizing the critical machines, certainly makes sense in view of the known fact that any schedule better than the one associated with S uses a selection in which at least one arc of every longest path in D_s is reversed. To conform this we performed extensive testing on randomly generated problems and found the former methods were not giving satisfactory results compared to the last two methods.

We carried out our final testing with the ones which were performing well in the initial testing. The description of experimental setup and computational results were tabulated in chapter 5.

Chapter 4

Concepts and Application of Tabu Search

In this chapter we introduce several parameters of the tabu search heuristic which govern the performance of the heuristic itself. First we present a brief description of the various elements of this global optimization technique. Then we describe in detail how we adapted these structural entities to solve the single machine optimization of shifting bottleneck procedure and to solve directly the overall job shop problem. Our implementations investigates some new features of tabu search including reversing the critical arcs in blocks and random tabu list length.

4.1 Introduction

With its roots going back to the late 1960's and early 1970's, tabu search was proposed in its present form a few years ago by Glover [28]. Since that time, tabu search has proved to be a remarkably effective approach to a wide spectrum of problems. Nowhere has this success been more marked than in production scheduling [73]. Tabu search procedures that incorporate basic elements and hybrids of these procedures with other heuristic and algorithmic methods, have succeeded in finding improved solutions to problems in scheduling, sequencing, resource allocation, investment planning, telecommunication and many other areas [28].

Tabu search is a meta-heuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality. The local search procedure is a search that uses an operation called "move" to define the neighborhood of any given solution. One of the main components of the tabu search is its use of memory, which creates a more flexible search behavior. Memory based strategies are therefore the hallmark of tabu search [26].

Tabu search is based on the premise that problem solving, in order to qualify as intelligent, must incorporate adaptive memory and responsive exploration. The use of adaptive memory contrasts with memoryless designs, such as those inspired by metaphors of physics and biology, and with rigid memory designs, such as those exemplified by branch and bound and its AI-related cousins. The emphasis on responsive exploration in tabu search, whether in a deterministic or probabilistic implementation, derives from the supposition that a bad strategic choice can yield more information than a good random choice [28].

4.2 Main concepts and strategies of tabu search

The idea of tabu search may be explained as follows. Given a function $f(x)$ to be optimized over a set X, tabu search begins in the same way as ordinary local search, proceeding iteratively from one point (solution) to another until a chosen termination criterion is attained. Each $x \in X$ has an associated neighborhood $N(x) \in X$, and each solution $x' \subset N(x)$ is reached from x by an operation called a move [26].

Tabu search has the ability to go beyond the local search for finding the search space by efficiently modifying $N(x)$, thereby effectively replacing it by another neighborhood $N^*(x)$. An impressive feature of tabu search is its special memory structure which helps to determine $N^*(x)$ and organize the way in which the space is explored.

The solutions admitted to $N^*(x)$ by these memory structures are determined in several ways. One of these which gives tabu search its name identifies solutions encountered over a specified horizon, and forbids them to belong to $N^*(x)$ by naming them tabu [28]. This forces the search to go beyond previously visited solutions. The most generic view of the tabu search is as follows:

```
1. Choose an initial solution $x$ in solution space $X$
   Set:  current best solution $x^* \leftarrow x$; Iteration counter $K \leftarrow 1$
2. While the termination criterion is not satisfied do
   Set $K \leftarrow K + 1$
   calculate $f(x')$ for all $x' \in N^*(x)$
   select the best neighbor $x''$ from $N^*(x)$
   set $x \leftarrow x''$
   if $f(x) < f(x^*)$ then $x^* \leftarrow x$
   update tabu list
   end do
3. Return $x^*$
```

4.3 Initial solution

As it was stated earlier, tabu search is an iterative technique which moves from one point to another, given a feasible solution. Sometimes the performance of the search depends on the initial solution which is the starting point of the search.

For our adaptation to the single machine problem of shifting bottleneck procedure, we use earliest due date (EDD) dispatching rule. EDD was known to be optimal for maximum lateness problem under certain conditions of single machine [56]. Each operation could have several due dates with respect to various jobs. According to EDD, priority is given to those operations belonging to jobs which has earliest due date. It can be proved that EDD does not introduce cycles in the graph.

Theorem 4.1 *Earliest due date (EDD) dispatch does not introduce cycles in the disjunctive graph.*

Proof: If there exists a directed path from i to j in the conjunctive graph then the due date of operation i with respect to any job will be at least due date of operation j with respect to that job minus the length of the path from i to j.
EDD will schedule i before j, since the due date of i is less than that of j.
Hence no cycle is created.

This condition may not hold for weighted shortest processing time dispatch. So although that dispatching rule is known to be optimal for the single machine weighted tardiness problem, we do not consider at the start point to avoid cycles in the graph.

For our implementation of the tabu search directly to overall job shop, we try two starting points based on two dispatching rules : weighted shortest processing time rule (WSPT) and earliest available due date rule (EADD). Earliest available due date rule is similar to EDD rule, except every time as the machine becomes idle, the next selected operation is the one which belongs to the job that has earliest due date and among the operations waiting.

4.4 Neighborhood

Tabu search is a global iterative optimization method: The search moves from one solution to another, in order to improve the quality of the solutions visited. This supposes a neighborhood structure. The most common neighborhood schemes used for single machine problems are

- Adjacent pairwise interchange, where an operation may be swapped with operations directly before it or after it in the schedule.
- Swap (or all pairwise interchange), where any two operations in the schedule can exchange the positions in the sequence.
- Insertion, where any operation can be inserted in front of any other operation in the schedule.

Comparative studies of these three neighborhood schemes have shown that the insert neighborhood scheme produces consistently better results than the swap neighborhood scheme [27] and that the swap neighborhood scheme produces consistently better results than the adjacent pairwise interchange neighborhood scheme [71]. Accordingly, we select the insertion neighborhood for our single machine problem. However because of the potential of introducing cycles, we check each solution in the neighborhood with respect to the delayed precedence constraints and we eliminate all those sequences that violate any of the constraints.

One of the most common neighborhood generation methods found in the literature for job shop makespan problem was given by Nowicki and Smutnicki [25], which was successful for the problems with the objective of minimizing the makespan in the job shop. According to them "A move is defined by the interchange of two successive tasks T_i and T_j, where either T_i or T_j is the first or last task in a block that belongs to a critical path."

Note: A block is a maximal sub sequence of operations that contain operations processed on the same machine.

For our tabu implementations on the overall job shop, neighborhood consists of reversing an arc from the disjunctive arcs. This corresponds to exchanging the position of 2 adjacent operations on one of the machines. For the problems with makespan objective only such arcs that are on critical path from source node to the common target node are considered for two reasons:

- They are the only arcs that can potentially improve the objective function.
- As Van Laarhooven [53] showed, this reversal on the critical path does not introduce cycles in the disjunctive graph.

This is one of the neighborhood structure (N_1) we test for the problems with total weighted tardiness as objective.

We also test a second neighborhood structure (N_2) based on the fact that for weighted tardiness objective, each job has its own target node and therefore its own critical path from the origin. Accordingly we consider all arcs on any of these critical paths. To overcome cycles in the disjunctive graph, we extend the idea of Van Laarhoven [53] that any critical arc of an acyclic digraph will not be cyclic when it is reversed.

Theorem 4.2 *Suppose that $e = (v, w) \in E_i$ is a critical arc of an acyclic digraph D_i. Let D_j be the digraph obtained from D_i by reversing the arc e in E_i. Then D_j is also acyclic.*

Proof: Suppose that D_j is cyclic. Because D_i is acyclic, the arc (w, v) is part of the cycle in D_j. Consequently, there is path $(v, x, y, \ldots, w)$ in D_j. But this path can also be found in D_i and is clearly a longer path from v to w than the arc (v, w). This contradicts the assumption that (v, w) is on the longest path in D_i. Hence D_j is acyclic.

For both cases, if there are multiple critical paths for a given job in the disjunctive graph, we select one arbitarily.

4.5 Short term memory

An important distinction in the tabu search arises by differentiating between short term memory and longer term memory. Each type of memory is accompanied by its own special strategies. The most commonly used short term memory keeps track of solution attributes that have changed during the recent past, and is called recency-based memory. Recency based memory is exploited by assigning a tabu-active designation to selected attributes that occur in solutions recently visited. Solutions that contain tabu-active elements, or particular combinations of these attributes are those that become tabu. This prevents certain elements from belonging to $N^*(x)$ and hence from being revisited [28].

The above process is managed by maintaining a list called tabu list, which keeps track of tabu-active elements and implicitly or explicitly identifies their current status. The duration that the attributes remain tabu-active (usually measured in terms of number of iterations) is called tabu-tenure. Tabu tenure may vary for different types or combinations of attributes and can also vary over different stages of the search [28].

Tabu list can be maintained in two ways: dynamic and static. In the dynamic tabu list, the length of the tabu list varies with respect to the search history. Whenever an improvement is observed in the search then the attributes are added to the tabu list, if not the attributes are ignored. In other words, the length of the tabu list remains same. And in the static tabu list, as the name itself indicates the length of the list will be fixed. Tabu list is considered to be one of the tuning parameter of the tabu search. Variations in the length of the list will effect the performance of the algorithm. For our implementations of both job shop and single machine problem we maintain static tabu list. Many researchers reported that static tabu lists would be successful for machine scheduling problems.

Along with the length of the tabu list, tabu condition i.e., content of the tabu list also plays an important role in tabu search. For our implementations, we handled the tabu list in three different ways for the single machine problem:

1. The tabu list contains a list of operations and the original positions of those operations in the sequence during an improving move. A move is considered to be tabu if it returns any operations to its original position as stored in the tabu list.

2. The tabu list contains the list of operations and the resulting positions of those operations in the sequence during an improving move. A move is considered to be tabu, if it moves any operations from its current position as stored in the tabu list.

3. The list contains a list of operations, whose insertion resulted in the selected move. A move is considered tabu, if it involves inserting an operation that is in the tabu list.

For our overall job shop problem we considered only one tabu condition. A tabu list contains a list of reveresed arcs. A move is considered to be tabu if it reversed back an arc in the tabu list.

4.6 Aspiration criteria

Another important element of flexibility in tabu search is the aspiration criterion. The tabu status of a solution is not an absolute, but can be overruled if certain conditions are met, expressed in the form of aspiration levels. In effect these aspiration levels provide thresholds of attractiveness that govern whether the solutions may be considered admissible in spite of being classified tabu. Clearly a solution better than any previously seen deserves to be considered admissible. Similar aspiration criteria can be defined over subsets of solutions that belong to common regions or that share specified features (such as particular functional value or level of unfeasibility) [26].

The aggressive aspect of the tabu search is reinforced by seeking the best available move that can be determined with an appropriate amount of effort. The meaning of best in the tabu search applications is customarily not limited to an objective function value. The aspiration criteria which we have selected for both the job shop and the single machine problems is any tabu move is accepted, if it improves the overall best objective function found so far. The aspiration criteria can also be used as the tuning parameter of the search. We intend to test the performance of the algorithm with and with out incorporating aspiration criteria.

4.7 Long term memory

In some problems, the short term memory components are sufficient to produce very high quality solutions. However, in general, the tabu search becomes significantly stronger by including longer term memory concepts and its associated strategies.

The most common long term memory components of the tabu search are intensification and diversification strategies. Intensification strategies are based on modifying the choice rules to encourage move combinations and solution features. They may also initiate a return to attractive regions to search them thoroughly. Another type of intensification approach is intensification by decomposition, where restrictions may be imposed on parts of the problem or solution structure in order to generate a form of decomposition that allows a more concentrated focus on other parts of the structure .

Tabu search diversification strategies, as their name suggests, are designed to drive the search into the unexplored regions of the search. Some of these strategies are designed with the chief purpose of preventing the search processes from cycling, i.e., from endlessly executing the same sequence of moves (or more generally, from endlessly executing and exclusively revisiting the same set of solutions). Often they are based on modifying choice rules to bring attributes into the solution that are infrequently used .

Regarding the long term memory implementations in our algorithms, we implement simple diversification approaches which were proven to be successful in machine scheduling problems [26]. After a certain number of non-improving moves over the overall cost, we diversify the search by jumping to the best found local optima in the search history and start the search in different direction, thereby guiding the search to unexplored regions of the solution space. In returning to a previously visited locally best solution, there will be a good chance that we are returning to a good search neighborhood [73].

4.8 Termination criteria

Tabu search is an iterative technique which has no end. It is in a sense similar to an infinite loop. Termination criteria are points where the search stops and are given as an input parameter to the search. The most commonly used stopping criteria are:

- A solution which is close enough to a given lower bound of goal function value.
- A limit on the number of iterations without improvement in the objective function value is reached.
- A limit on the computational time is reached.

In our implementations, we considered three termination conditions. The first one is specifying a maximum number of iterations for the search. Minimizing the objective function value to zero was kept as the second condition of termination for our job shop problems. We do not implement the same condition for our single machine problems. During the bottleneck selection of shifting bottleneck method, we select the machine with maximum total weighted tardiness and we break the ties with maximum weighted lateness. Initially as most of the machines are unscheduled, the total weighted tardiness may be zero most of the time. In that case, we select the machine with total weighted lateness as the bottleneck machine. So, as single machine problems are solved using the tabu search, we may not select the critical machine with the above termination condition. Since the tabu search application to the overall job shop does not involve any bottleneck selection methods, the above condition can be set as one of the termination conditions.

Third condition being neighborhood of a move becoming empty. The search terminates immediately, if any one of the conditions is true. Again this termination criterion acts as a tuning parameter of the search, which has the direct impact on the performance of the algorithm.

4.8.1 Tabu search algorithm

Now that we explained all the tabu characteristics and their implementations in our heuristic, we present an algorithmic view of our tabu search heuristic. Before presenting the algorithm, we define the parameters used in our procedure as follows:

$\boldsymbol{x}$	Initial solution.
$\boldsymbol{x}'$	Best neighbor
$\boldsymbol{N(x)}$	Neighborhood of a move
$\boldsymbol{K}$	Iteration counter
$\boldsymbol{NI}$	Non-improving moves counter
$\boldsymbol{TL}$	Tabu list length
$\boldsymbol{L_{MIN}}$	Minimum tabu list length
$\boldsymbol{L_{MAX}}$	Maximum tabu list length

$\boldsymbol{F}$	Improvement indicator in the search
$\boldsymbol{LOL}$	Local-optimal-list which stores all the local optima
$\boldsymbol{E}$	Element in local-optimal-list

```
1. $K \leftarrow 0$, $TL \leftarrow 0$, $NI \leftarrow 0$, $LOL \leftarrow$ empty, $x \leftarrow$ initial schedule
2. While termination criterion is not satisfied find $N(x)$, select best
   neighbor $x'$.
   if $[f(x) < f(x')]$
     • if $TL > L_{MAX}$, Delete the oldest move and update tabu list
     • set $LOL \leftarrow LOL$
     • Set $F = 1, NI = 0$
     • set $x \leftarrow x'$ and goto step 2
3. else $[f(x) > f(x')]$
     • if $TL > L_{MAX}$, Delete the oldest move and update tabu list i.e.,
       $TL + A$
     • set $LOL \leftarrow LOL + (x', f(x') and TL)$
     • Reset $F \leftarrow 1$
     • $NI \leftarrow NI + 1$
       check if $NI < NI_{MAX}$, set $x' \leftarrow x$ and goto step 1
       else set $x' \leftarrow E$, Remove $E$ from $LOL$
       If $LOL \leftarrow 0$ STOP
       else STOP
4. Return $x^*$
```

Chapter 5

Computational Study and Comparison of Heuristics

Both shifting bottleneck and tabu search procedures have many structural elements and parameters, the selection of which can greatly affect the performance of the heuristics. In this chapter we start with some preliminary testing using problem instances taken from the literature in order to narrow down the number of structural elements and parameter settings. We focus our attention on varying the tabu parameters, in particular the neighborhood structure, the tabu conditions, the maximum number of iterations and the maximum number of non-improving moves, for both the single machine and the job shop problems. We also test various bottleneck selection methods and re-optimization procedures. Then we present more extensive results from running the procedures with the selected parameters on randomly generated problems.

All our heuristics presented in this chapter were coded in the programming language DrScheme. The code uses a library of functions in scheduling and disjunctive graph manipulations previously developed at concordia university.

5.1 Preliminary testing

For our primary testing, we considered various problem instances from the literature. In particular we carried out these testings on problem instances ORB1-10, which were generated by Applegate and Cook [9]. Since these instances were formulated for minimizing the makespan in job shops, we have to add a weight and a due date to each job. We followed the design pattern given by Pinedo and Singer [52] for transforming problem instances with a makespan objective to problems with a total weighted tardiness objective. The first 20% of jobs are given a weight of 4, the next 60% of jobs are given a weight of 2, and last 20% of jobs are given a weight of 1. As Singer and Pinedo [52] note, in practice 20% of customers are considered very important, 60% of them are of average importance and the remaining 20% are of less importance. In this way the distribution of the weights can be justified.

The due date of a job j is set equal to the release date of the job plus the sum of the processing times of all the operations of job j and multiplied by a due date tightness factor f i.e.,

$$d_j = r_j + \left\lfloor f * \sum_{i=1}^{10} p_{ij} \right\rfloor$$

The structural elements of both procedures varied in preliminary tests are explained in the following subsections:

5.1.1 Neighborhood structure

We considered only insertion as the neighborhood generation method for single machine problems and we choose this method for our final testings as well.

For the job shop problem, we tested the adapted neighborhood structures by keeping all other parameters fixed. Table 5.1 shows the results for adapted neighborhood structures N_1 and N_2 for the overall job shop problem. Although N_2 was slightly better than N_1, the difference was not so significant that we eliminate

Problem Instances	N_1			N_2		
	(200, 10)	(200, 20)	(200, 50)	(200, 10)	(200, 20)	(200, 50)
ORB 1	2313	2376	2457	2639	2639	2639
ORB 2	1153	1153	1153	1387	1344	1394
ORB 3	2252	2248	2233	2193	2193	2193
ORB 5	1572	1554	1820	921	1053	764
ORB 6	1241	1203	1203	908	907	1148
ORB 8	1670	1670	1670	1588	1588	1588
ORB 9	757	757	757	665	665	665
ORB 10	734	734	834	773	963	893
Average	1461	1515	1515	1424	1410	1410

Table 5.1: Total weighted tardiness for neighborhood structures N_1 and N_2. The numbers in parenthises represent in order the total number of iterations performed and the maximum number of non-improving moves allowed in the search.

N_1 for our final testings. Hence we decided to carry both methods for our final testings on randomly generated problem instances.

5.1.2 Tabu condition

One of the important parameters of the tabu search for single machines was the tabu condition. As discussed in the preceeding chapter, we designated moves to be tabu in 3 different ways. We tested these tabu conditions with problem instances ORB1-10. For convenience we name them to be as insert-From, insert-To and insert-op. Among these insert-To was found best 80% of the times compared to insert-From and 20% of times with insert-op. And insert-op was observed to outperform 80% of times when compared to insert-From and 20% of times with insert-To. Based on these results, we decided to perform our final testing with the two tabu conditions i.e., insert-To and insert-op. The results of these problems are tabulated in Table 5.2.

Since we framed only one tabu condition to job shop problem, we intend to test our algorithms with the same condition. The tabu condition which was considered for the job shop was "any reversal of arc on the critical path which has

Problem-Instances	Insert-From	Insert- To	Insert-op
ORB 1	2237	1853	1981
ORB 2	1135	879	879
ORB 3	1583	2665	2671
ORB 4	2615	1991	1991
ORB 5	1730	860	915
ORB 6	2716	1363	1056
ORB 7	809	518	388
ORB 8	1611	2720	2766
ORB 9	1737	1034	1034
ORB 10	1079	1491	1768
Average	1725	1537	1544

Table 5.2: Performance of shifting bottleneck procedure with different tabu conditions in solving the single machine problem.

been already reversed is considered tabu till the tabu tenure expires".

5.1.3 Maximum number of Iterations and maximum number of non-improving moves

In an attempt to find the best iteration number and maximum number of non-improving moves for the single machine problems, we ran few cases of single machine problems by varying the non-improving moves counter for a fixed number of iteration count. Here varying the non-improving moves counter means that exploring the search space for finding better solutions. We tested by fixing the maximum iteration counter to 50 and varied the non-improving moves counter to 20, 10, and 5. Although the results did not show a significant difference, still we can bank on the lower side of the non-improving moves counter. For our final implementations, relatively we reduced total number of iterations to 20 and non-improving moves counter to be between 4 and 10. Some of the results obtained are presented in Table 5.3.

On the way of testing the performance of neighborhood structures of job shop, we could figure out other tabu parameter settings for those neighborhoods. We tested these neighborhoods keeping the iteration counter to be fixed with 200

Problem Instances	MaxCounter = 20		MaxCounter = 10		MaxCounter = 5	
	Insert-To	Insert-op	Insert-To	Insert-op	Insert-To	Insert-op
ORB 1	1853	1981	1853	1981	1853	1981
ORB 2	879	879	879	879	879	879
ORB 3	2665	2671	3079	2671	2157	2671
ORB 4	1991	1991	1991	1991	1991	1991
ORB 5	860	915	860	915	869	915
ORB 6	1391	1056	1391	1056	1391	1056
ORB 7	518	388	518	388	518	388
ORB 8	2205	2766	2205	2766	2720	2766
ORB 9	1037	1034	1034	1034	1034	942
ORB 10	1491	1768	1491	1768	1491	1768
Average	1489	1544	1530	1545	1581	1535

Table 5.3: Performance of the shifting bottleneck procedure with varying number of non-improving moves and tabu conditions in solving the single machine problems. In all cases, the total number of iterations per problem was 50.

and with variations in non-improving moves. We could observe improvements in the solutions only in the first 50 iterations and in some cases to 100 iterations. With these observations on the behavior of algorithm with the total number of iterations and non-improving moves, we reduced them relatively for our final implementations on the random problem instances.

We decided to perform our testing with 80 iterations for neighborhood N_1 and 50 for N_2. We computed seperatly the total amount of time for each iteration for both the neighborhoods and the computation time of N_2 was found approximately 1.5 times of N_1. In order to be consistent with the computational effort given to both these neighborhoods, we set for N_1 the iteration counter of 1.5 times less than N_2. In 70% of the tested problems, best improvements were observed by allowing 20 non-improving moves before the search takes the diversification. And in 30% of the problems tested, setting non-improving moves counter to 10 was found useful, but never higher than 20. As a compromise between these two settings, we vary the non-improving moves in between 8 and 20 for final implementations as shown in Table 5.1 .

5.1.4 Tabu list length

For the both single machine and job shop tabu implementations, we follow the idea of various researchers of keeping the length of the tabu list to be in the range 7 and 2 [4] [79].

5.1.5 Bottleneck selection and re-optimization methods

Bottleneck selection methods were also tested with ORB1-10, we observed that our second bottleneck selection method i.e., machine with total weighted tardiness after applying the tabu search on each single machine was found superior to the latter one. In most cases this method outperformed our first method. Hence we chose this bottleneck selection method for our final testing with randomly generated problem instances. Another important aspect which was tested in our initial testing was the re-optimization procedures.

We tested both of the re-optimization procedures. In almost all the cases both the procedures performed equally well. Re-optimizing the critical machines on the critical path after each iteration is similar to re-optimizing all the machines after each iteration, if all the machines are on critical path. It is always possible for at least one operation of each machine to be on the critical path. In very rare situations, it can happen such that no operation of any machine is on critical path. Due to the above mentioned reasons, we proceed with first method of re-optimization i.e., re-optimizing all machines immediately after a machine is scheduled. The results of proposed bottleneck machine selections including the re-optimization procedures for ORB1-10 are given in Table 5.4. For easy understanding we introduced notation for these procedures which are of the form SBP-EADD-Every-To, where SBP represents the shifting bottleneck procedure, EADD indicates the bottleneck selection condition, Every represents the re-optimization procedure used and finally To stands for the tabu condition used in the procedure.

Problem Instances	EADD-Every		EADD-CM		Tabu-Every		Tabu-CM	
	To	op	To	op	To	op	To	op
ORB 1	1853	1981	1853	1981	1917	1917	1917	1917
ORB 2	879	879	1172	1130	532	532	532	532
ORB 3	2665	2671	2515	1174	1889	2187	1889	2187
ORB 4	1991	1991	2337	2337	1106	1449	1106	1449
ORB 5	869	915	744	744	1033	1033	1033	1033
ORB 6	1391	1056	2031	1146	1199	1681	1199	1681
ORB 7	518	388	444	444	440	275	440	275
ORB 8	2205	2766	2778	1744	1663	2000	1663	2000
ORB 9	1034	942	1383	547	438	438	438	438
ORB 10	1297	1768	2203	1700	924	924	924	924
Average	1470	1535	1746	1294	1114	1243	1114	1243

Table 5.4: Performance of the shifting bottleneck procedure with adapted bottleneck conditions, re-optimization procedures and tabu conditions in solving the single machine problem.

The reason for selecting these particular problem instances from the literature was the fact that optimal solutions for these problems were known. As a result we could compare the performance of our implementations with the optimal ones. In most cases, our solutions were found 30-35% closer to the optimal solutions. As we used approximation algorithms instead of optimization methods for our implementations, finding optimal solutions was difficult within a given computational effort. This could be the possible reason for the performance variations of our implementations with the optimal solutions. However 45-50% of improvement was observed from the initial solution of the tabu search.

5.2 Random problem instances

Our preliminary testing results produced some reliable parameters settings of the tabu search for both job shop and single machine problems for our final testing. In order to test the performance of our algorithms with randomly generated problems, we designed an experimental set up which takes into account:

- The due date tightness factor f which is a measure of tightness of due dates.
- Neighborhood procedures adapted for job shop problem.

We randomly generated the processing times of the jobs from a discrete uniform [1, 30] distribution and weights from a discrete uniform [1, 10]. In order to be consistent with the results reported else where, we assumed that the release dates are 0 for all jobs and that the due dates are given by

$$d_j = r_j + \left\lfloor f * \sum_{i=1}^{10} p_{ij} \right\rfloor$$

In contrast with testing reported elsewhere, where the same due date tightness factor is used for all jobs, we generate a different factor for each job from a discrete uniform $[1, 1 + 2(f - 1)]$ distribution. This results in an average due date tightness factor equal to f.

5.2.1 Experimental results and discussion

In the first phase, we test all the tabu parameters which were selected during preliminary testing and we discuss the variations of the performance of heuristics with the random problem instances. Later we present the the performance variations by varying the due date tightness factor f. We also present a brief discussion on neighborhood behavior of the overall job shop. First we start with the neighborhood structures of job shop problem, as we do not deal with any other neighborhood methods for single machine problems except insertion method.

Neighborhood methods

To test the average performance of our heuristics with different tabu parameters, we consider EADD as the initial solution for the tabu search and due date tightness factor of 1.5 as a compromise between the tight and loose due dates.

We can see that neighborhood N_2 on average is improving by 9% over N_1 as can be computed from Table 5.5. This is comparable to the improvement of 7% resulting from the preliminary testing in Table 5.1. Of course, several other parameters like total number of iterations and maximum non-improving moves may be influencing the quality of the solution, which we explain later. We also compute the average improvement of N_1 and N_2 on the initial solution as shown in Table 5.6. From that table we observe that N_2 consistently results in better improvements than N_1.

Problem	N_1		N_2	
Size	$(80, 8)$	$(80, 20)$	$(50, 6)$	$(50, 12)$
10×10	1466	1437	1213	1206
10×15	1210	1188	1074	1041
15×10	5676	5600	5143	5190
Average	2784	2741	2476	2479

Table 5.5: Average total weighted tardiness of the overall job shop with two neighborhood structures N_1 and N_2. The numbers in parenthises represent in order the total number of iterations performed and maximum number of non-improving moves allowed in the search.

Problem Size	% Improvement of N_1 over EADD		% Improvement of N_2 over EADD	
	$(80, 8)$	$(80, 20)$	$(50, 6)$	$(50, 12)$
10×10	24.31	25.80	37.38	37.73
10×15	34.76	35.95	42.11	43.85
15×10	7.39	8.63	16.08	15.31

Table 5.6: Percent improvement of N_1 and N_2 over initial solution (EADD) with different number of iterations and non-improving moves in the search.

Tabu Condition

We test the two tabu conditions which were selected from our earlier testings insert-To and insert-op with our random instances. The average results are displayed in Table 5.7. Here we see that the Insert-To condition is slightly better, but there is no clearly superior method. The difference between Insert-To and Insert-op varies from an improvement of the first over the second of about 19% to an improvement of the second over the first of about 4%.

Procedure	(MaxIter, MaxCounter)	Average		
		10 × 10	10 × 15	15 × 10
Insert-To	(20, 4)	522	348	2648
	(20, 10)	436	348	2577
Insert-op	(20, 4)	426	361	2841
	(20, 10)	538	361	2841

Table 5.7: Performance of shifting bottleneck procedure with two tabu conditions insert-To and insert-op in solving the single machine problem.

Maximum number of iterations and maximum number of non-improving moves

We ran our random instances with the best found settings in the preliminary tests. It can be noted from the Table 5.8 that the insert-To, with 20 iterations and 10 non-improving moves performing well with different sized instances. We can also observe that as the problem size increases, the quality of insert-To with 10 non-improving moves is improving.

Procedure	(MaxIter, MaxCounter)	Average		
		10 × 10	10 × 15	15 × 10
Insert-to	(20, 4)	522	348	2648
	(20, 10)	436	348	2577
Insert-op	(20, 4)	426	361	2841
	(20, 10)	538	361	2841

Table 5.8: Performance of shifting bottleneck procedure with varying number of non-improving moves in solving the single machine problems for random instances. The numbers in parenthises represent in order the total number of iterations performed and the maximum number of non-improving moves allowed in the search.

For our job shop problem, as it is evident from Table 5.5 that N_2 performs better in all the cases irrespective of the size of the problem. The maximum iteration count 50 with non-improving moves of 12 was observed to be feasible.

Now that we have seen the effects of varying the tabu structural elements, we intend to test the performance of our heuristics by changing the due date tightness factor f. In the first phase we ran 60 instances of 10×10 problem to check the variations. We varied three different values of $f = 1.3, 1.5$ and 1.7 and tested with two neighborhood structures (N_1, N_2) of overall job shop and single machine shifting bottleneck procedure (SBP). For our implementations with various due date tightness factors, we have to set accurate tabu parameters.

From conclusions of Table 5.8, we consider total iteration and non-improving moves count to be 20 and 10 respectively for single machines. Regarding the tabu conditions of single machine problem, we test with both the tabu conditions. And based on the results of Table 5.5, we take $(80, 20)$ and $(50, 12)$ as the total number of iterations and number non-improving number for both N_1 and N_2 respectively. As mentioned earlier the initial solution is still going to be earliest available due date dispatch.

f	Total Weighted Tardiness				
	S B Procedure		Overall Job Shop		EADD
	insert-To (20,10)	insert-op (20,10)	N_1 (80,20)	N_2 (50,12)	
1.3	1321	1196	2515	2292	3000
1.5	436	485	1466	1206	1937
1.7	497	426	1194	1063	1579

Table 5.9: Resulting total weighted tardiness of different procedures for 10×10 problems with different values of f

f	% improvement over EADD			
	insert-To	insert-op	N_1	N_2
1.3	55.9	60.1	16.6	23.6
1.5	77.4	78.0	24.3	37.7
1.7	68.5	69.2	24.3	37.7

Table 5.10: Percent improvement of the heuristics over EADD

It can be observed from Table 5.9 that the variation of the due date tightness factor can greatly vary the performance of the heuristic. Under tight due date conditions, the change in tardiness cost is almost becoming triple in comparison with the other two. In Table 5.10 we also computed the percentage of improvement from the initial solution i.e. EADD. Insert-op was found superior with 78% decrease from the initial solution. It can be observed almost every where that N_2 was found superior than N_1. Since the objective function value is rapidly increasing due to very tight due dates. We considered to perform rest of runs with $f = 1.5$ and $f = 1.7$

We ran various randomly generated problem instances on our algorithms with the selected values of f. More than 120 problem instances were generated to test different problem sizes. The results obtained are presented in Table 5.11 and Table 5.12 . Some of the observations of our study are listed below:

Heuristic	Problem Size	Number Of Runs	Average Wt.Tardiness	Standard Deviation	% Improvement Over EADD
N_1	10 × 10	20	1437	732	25.8
	10 × 15	15	1188	619	35.9
	15 × 10	10	5600	1905	8.6
N_2	10 × 10	20	1206	717	37.4
	10 × 15	15	1041	599	43.8
	15 × 10	10	5143	1783	32.8
SBP	10 × 10	20	426	538	78.0
	10 × 15	15	348	484	81.2
	15 × 10	10	2577	1190	57.9

Table 5.11: Average performance of different heuristics with $f = 1.5$

5.2.2 Summary of test results

Here we summarize the interesting observations of our study:

- There was a significant difference in the average performance of heuristics with the variation in due date tightness factor. Neighborhood type N_2 of the overall job shop has performed better than N_1 as shown by Table 5.9.

Heuristic	Problem Size	Number Of Runs	Average Wt.Tardiness	Standard Deviation	% Improvement Over EADD
N_1	10×10	20	1170	770	25.8
	10×15	15	840	793	31.1
	15×10	10	4197	1471	11.1
N_2	10×10	20	1122	828	28.9
	10×15	15	641	654	47.5
	15×10	10	3926	1555	16.8
SBP	10×10	20	485	423	69.2
	10×15	15	141	170	88.4
	15×10	10	1085	634	77.0

Table 5.12: Average performance of different heuristics with $f = 1.7$

- The performance of SBP is significantly better than the tabu search with both neighborhood structures as given by Table 5.9.
- Percentage of improvement from the starting solution i.e, earliest available due date dispatch (EADD) for SBP was remarkable good between 60% and 80%, where as the other neighborhood structures of the tabu search applied to over all graph, could bring only about 50% of improvement as shown in Table 5.10.
- Percentage of improvement from starting solution was observed to be between 70 - 80 % in case of loose due dates as given by Table 5.10
- The performance of both the tabu and SBP heuristics are better for loose due dates than that of tight due dates as shown in Table 5.9 and Table 5.10.
- Performance of the heuristics is greatly varied depending on initial solution of the tabu search.
- Varying the number of non-improving moves in the tabu search, effected the performance of the algorithms as shown by Table 5.8.
- For the tabu search implementation on overall job shop problem

  - Performance of N_1 was found better by setting the maximum number of iteration to 80 and maximum non-improving moves to 20.
  - N_2 was observed to be efficient with 50 maximum number of iterations and 12 maximum number of non-improving moves before diversification.

- For tabu search implementation on single machines, maximum iteration count of 20 with 10 non-improving moves was found better most of the time.
- The tabu condition Insert-To, set for the single machine problems has outperformed the other condition insert-op.
- For SBP, bottleneck selection criterion played an important role for its performance, as the tabu search is performed on every single unscheduled machine to select the best.

Chapter 6

Conclusion and Future Work

This thesis provides insight into the total weighted tardiness problem in the job shops with all jobs having possibly different due dates. It gives a comparative study of well known heuristics shifting bottleneck procedure and tabu search. In the way of implementing shifting bottleneck procedure, this research demonstrated efficient methods of selecting the bottleneck machine. The application of the tabu search to the single machine optimization of shifting bottleneck procedure was well demonstrated.

In general, it is not easy to solve the job shop problem with jobs having different due dates, because a delay in any operation of any machine will effect the due date of every job. On the other hand, these problems are known to be difficult to solve optimally. Hence approximation algorithms are the intelligent choice to these problems.

Although all our discussion and results are for linear tardiness costs, the generalization of the procedures we give to the case of nonlinear costs is straightforward. In fact all the heuristic implementations we give only need an evaluation of the cost function for a given tardiness, and do not depend on that function being linear.

Some of the important results are:

- The due date tightness factor f has a significant effect on the performance of the heuristics both in absolute terms and in relative terms. In other words f affected the total weighted tardiness resulting from the heuristics as well as on heuristics performed in comparison with one another. The potential improvement over dispatching rules given by the heuristics is largest for an intermediate value of the tightness factor.

- The tabu structural parameters have a significant influence on the performance of heuristics.

- The shifting bottleneck procedure was comparatively yielding better results than applying the tabu search to the overall job problem with weighted tardiness as objective. This is in apparent contradiction to the observation of other researchers who noted that the reverse is true when the objective is to minimize makespan.

Finally we give some avenues for improvement in our work and for future research on this problem.

- For tabu search as applied to overall job shop scheduling problems, the most common method of generating neighborhood is by reversing arcs on the critical path in the disjunctive graph. Whenever an arc is reversed, the graph need to be updated with the new release times and due dates of the operations. In our implementation, the whole graph is updated at every iteration. However there might be some operations in the same disjunctive graph which do not need updating if they are not affected by the reversed arc. During this process of updating the graph by updating all the operations, more amount of computational time is being wasted. More efficient methods should be developed to filter the operations which need updating. This time saved can be utilized for the search itself, thereby increasing the chances of finding improved solutions.

- The 'Backtracking mechanism' is also one of the important mechanism in shifting bottleneck procedure that we did not implement. Instead of selecting a single machine to schedule next in the machine selection step, a list of the most critical β machines is selected and a tree of possible machine sequences is formed. According to this at each node of the branching tree, all machines already scheduled are re-optimized. Each node of the tree represents a partial order in which machines have been scheduled. Testing is needed to determine the optimal value of β for a given computational time as compared to the other parameters we have tested.

Bibliography

[1] Elasyed A. and Boucher O. *Analysis and control of production systems.* Prentice Hall Inc, NewJersey, 1994.

[2] Fadlalla A., Evans J., and Levy M. A greedy heuristic for the mean tardiness problem. *Computers and Operations Research*, 21:329–336, 1994.

[3] Hariri A. M. A. and Potts C. N. An algorithm for single machine sequencing with release dates to minimize the total weighted completion time. *Discrete Applied Mathematics*, 5:99–109, 1983.

[4] Islam A. and Eksioglu M. A tabu search approach for the single machine mean tardiness problem. *Journal of Operational Research Society*, 48:751–755, 1997.

[5] Alidaee B. and Dragan I. A note on minimizing the weighted sum of tardy and early common penalties in a single machine : A case of small common due date. *European Journal of Operational Research*, 96:559–563, 1997.

[6] Chu C. A branch and bound algorithm to minimize total tardiness with different release dates. *Naval Research Logistics*, 39:265–283, 1992.

[7] Chu C. and Portmann M. C. Some new heuristic methods to solve the $n/1/r_i/t_i$. *European Journal of Operations Research*, 58:404–413, 1992.

[8] Koulamas C. Total tardiness problem : Review and extensions. *Operations Research*, 42:374, 1994.

[9] Applegate D. and Cook W. A computational study of job shop scheduling. *ORSA Journal on computing*, 3:149–156, 1991.

[10] Fry T. D., Vicens L., Macleod K., and Fernandez S. A heuristic solution procedure to minimize t on a single machine. *Journal of Operational Research Society*, 40:293–297, 1989.

[11] Liman S. D., Panwalkar S. S., and Thongmee S. Determination of common due window location in a single machine scheduling problem. *European Journal of Operational Research*, 93:68–74, 1996.

[12] Balas E. Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. *Operations Research*, 17:941–957, 1969.

[13] Balas E. Project scheduling with resource constraints. *Application of Mathematical Programming Techniques, English Universities Press Ltd.,*, 1970.

[14] Balas E. and Vazacopoulos A. Guided local search with shifting bottleneck for job shop scheduling. *Management Science*, 44:262–275, 1998.

[15] Balas E., Lenstra J. K., and Vazacopoulos A. The one-machine problem with delayed precedence constraints and its use in job shop scheduling. *Management Science*, 41:94–109, 1995.

[16] Bector C. E., Gupta Y. P., and Gupta M. C. Determination of optimal common due date and optimal sequence in a single machine job shop. *Internation Journal of Production Research*, 26:613–628, 1988.

[17] Cheng T. C. E. An algorithm for con due date determination and sequencing problem. *Computers and Operation Research*, 14:537–542, 1987.

[18] Demirkol E., Mehta S., and Uzsoy R. A computational study of shifting bottleneck procedures for shop scheduling problems. *Journal of Heuristics*, 3:111–137, 1997.

[19] Graham R. E., Lenstra E. L., and RinnoyKan A. H. G. Optimization and approximation in deterministic sequencing and scheduling, a survey. *Annals of discrete mathematics*, 4:287–326, 1979.

[20] Holsenback J. E., Russell R. M., Marklandand R. E., and Philipoom P. R. An improved heuristic for the single machine, weighted tardiness problem. *Omega International Journal of Management Science*, 27:485–495, 1999.

[21] Holsenbeck J. E. and Russell R. M. A heuristic algorithm for sequencing on one machine to minimize total tardiness. *Journal of Operations Research Society*, 43:53–62, 1992.

[22] Morton T. E. and Rachamadugu R. M. U. Myopic heuristic for the single machine weighted tardiness problem. *Working paper*, 1982.

[23] Morton T. E. and Rachamadugu R. M. V. Myopic heuristics for the single machine weighted tardiness problem. *GSIA working paper*, 1982.

[24] Morton T. E. and Pentico D. W. *Heuristic scheduling systems.* Some Series here. John Wiley and Sons Inc, New York, 1993.

[25] Nowicki E. and Smutnicki C. A fast taboo search algorithm for the job shop problem. *Management Science*, 42:797–812, 1996.

[26] Glover F. *Tabu search fundamentals and uses.* University of Colorado, Colorado, 1995.

[27] Glover F. and Laguna M. Integrated target analysis and tabu search for improved scheduling systems. *Expert systems with Applications*, 6:287 –297, 1993.

[28] Glover F. and Laguna M. *Tabu search.* Kluwer acedemic publishers, Boston, 1997.

[29] Muth J. F. and Thompson G. L. *Industrial scheduling.* Prentice hall, Englewood cliffs NJ, 1963.

[30] RinnoyKan A. H. G. *Machine sequencing problems: Classification, Complexity and Computation.* Nijhoff, The Hague, 1976.

[31] Holtsclaw H. and Uzsoy R. Machine criticality measures and subproblem solution procedures in shifting bottleneck methods : A computational study. *Journal of Operational Research Society*, 47:666–677, 1996.

[32] Kise H., Ibaraki T., and Mine H. Performance analysis of six approximation algorithms for the one machine maximum lateness scheduling problem with ready times. *Journal of Operational Research Society*, 22:205–223, 1982.

[33] Adams J., Balas E., and Zawack E. The shifting bottleneck procedure for job shop scheduling. *Management Sciences*, 34:391–401, 1988.

[34] Anderson E. J. and Nyirenda J. C. Two new rules to minimize tardiness in a job shop. *International Journal of Production Research*, 28:2277–2292, 1990.

[35] Carlier J. and Pinson E. An algorithm for solving the job-shop problem. *Management Science*, 35:164–176, 1989.

[36] Chambers R. J., Carraway, Lowe T. J., and Morin T. L. Dominance and decomposition heuristics for single machine scheduling. *Operations Research*, 39:639–647, 1991.

[37] Davis J. and Kanet J. Single machine scheduling with non regular convex performance measure. *working paper*, 1988.

[38] Du J. and Leung J. Y. Minimizing total tardiness on one machine is np-hard. *Mathematics of Operations Research*, 15:483–495, 1990.

[39] Kanet J. J. Minimizing variation of flow time in single machine systems. *Management Science*, 28:643–651, 1981.

[40] Legeweg B. J., Lenstra J. K., and Rinnooy Kan A. H. G. Job shop scheduling by implicit enumeration. *Management Science*, 24:441–450, 1977.

[41] Noronha S. J. and Sharma V. V. S. Knowledge-based approaches for scheduling problems: A survey. *IIEE Transactions*, 3:160–171, 1991.

[42] Schuttan J. M. J. Practical job shop scheduling. *Working paper, Laboratory of production and operations management, Department of mechanical engineering, University of Twente, Netherlands*, 1995.

[43] Vepsalainen P. J. and Morton T. E. Priority rules for job shops with weighted tardiness costs. *Management Science*, 33:1035–1047, 1987.

[44] Wilkerson J. and Irwin J. D. An improved algorithm for scheduling independent tasks. *AIIE Transactions*, 3:333–342, 1971.

[45] Lenstra J. K., RinnoyKan A. H. G., and Brucker P. Complexity of machine scheduling problems. *Annals of Dicreate mathematics*, 1:343–362, 1977.

[46] Suna K., Omer K., and Meral A. Efficient algorithm for the single machine tardiness problem. *International Journal of Production Economics*, 36:213–219, 1994.

[47] Lawler E. L. A psuedopolynomial algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, 1:331–342, 1977.

[48] Lawler E. L. and Moore J. M. A functional equation and its application to resource allocation and sequencing problems. *Management Science*, 16:77–84, 1977.

[49] Ovacik I. M. and Uzsoy R. The shifting bottleneck algorithm for scheduling semiconductor testing operations. *Journal of Electronics Manufacturing*, 2:119–134, 1992.

[50] Pinedo M. *Scheduling theory,algorithms and systems.* Prentice-Hall Inc, New-jersey, 1995.

[51] Pinedo M. and Singer M. A computational study of branch and bound techniques for minimizing the total weighted tardiness in job shops. *IIE Transactions*, 30:109–118, 1998.

[52] Pinedo M. and Singer M. A shifting bottleneck heuristic for minimizing the total weighted tardiness in job shop. *Naval Research Logistics*, 46, 1999.

[53] Van Laarhooven P. J. M., Aarts E. H. L., and Lenstra J. K. Job shop scheduling by simulated annealing. *Operations Research*, 40:113-125, 1992.

[54] Raman N. and Talbot Brian N. The job shop tardiness problem : A decomposition approach. *European Journal of Operational Research*, 69:187–199, 1993.

[55] Esogbue E. O., Bellman, and Nabeshima I. Mathematical aspects of scheduling and applications. *International series in modern applied mathematics and computer science*, 4, 1982.

[56] Brucker P. *Scheduling algorithms.* Springer, New York, 1998.

[57] White K. P. and Rogers V. R. Job shop scheduling: Limits of the binary disjunctive formulation. *Internation Journal of Production Research*, 28:2187–2200, 1990.

[58] Potts and Wassenhowe V. A decomposition algorithm for single machine total tardiness problem. *Operations Research Letters*, 32:177–181, 1982.

[59] Potts and Wassenhowe V. Single machine tardiness sequencing heuristics. *IIE Transactions*, 23:346–354, 1991.

[60] Baker R. and Kenneth K. Sequencing rules and due-date assignments in a job shop. *Management Science*, 30:1093–1104, 1984.

[61] Baker K. R. *Introduction to sequencing and scheduling.* John Wiley and Sons Inc, New York, 1974.

[62] Baker K. R. and Shrage L. E. Finding an optimal sequence by dynamic programming: An extension to precedence-related tasks. *Operations Research*, 26:111–119, 1978.

[63] Baker K. R. and Kanet J. J. Job shop scheduling with modified due dates. *Journal of Operations Management*, 2:155–163, 1983.

[64] Baker K. R. and Bertrand J. W. M. A dynamic priority rule for scheduling against due dates. *Journal of Operations Management*, 3:37–42, 1982.

[65] Sule D. R. *Industrial scheduling.* Test Series. PWS Publishing company, Boston, 1996.

[66] Arthanari T. S. *On some problems of sequencing and grouping.* Unpublished Phd. Thesis, Indian statistical institute, Calcutta, India, 1974.

[67] Byoen E. S., Wu S., and Storer R. H. Decomposition heuristics for robust job shop scheduling. *IEEE Transactions on Robotics and Automation*, 14:303–313, 1998.

[68] Chang S., Lu Q., Tang G., and Yu W. On decomposition of the total tardiness problem. *Operations Research Letters*, 17:221–229, 1995.

[69] Dauzereperes S. A procedure for the one machine sequencing problem with dependent jobs. *European Journal of Operational Research*, 81:579–589, 1995.

[70] French S. *Sequencing and scheduling: An introduction to the mathematics of the job shop.* John Wiley and Sons Inc, New York, 1982.

[71] Tsubakitani S. and Evans J. R. Applying tabu search to the mean tardiness sequencing problem. *Working paper, University of Cincinnati*, 1992.

[72] Bagchi U., Sullivan R. S., and Chang Y. L. Minimizing absolute and squared deviations of completion times with different earliness and tardiness penalites about a common due date. *Naval Research Logistics*, 34:739–751, 1987.

[73] Barnes J. W. and Chambers J. B. Solving the job shop scheduling problem with tabu search. *IIE Transactions*, 27:257–263, 1995.

[74] Herrmann J. W. and Lee C. On scheduling to minimize earliness-tardiness and batch delivary costs with a common due date. *European Journal of Operational Research*, 70:272–288, 1993.

[75] Fathi Y. and Nuttle H. W. L. Heuristics for the common due date weighted tardiness problem. *IIE Transactions*, 22:215–225, 1990.

[76] Kim Y. and Yano C. A. Minimizing mean tardiness and earliness in single-machine scheduling problems with unequal due dates. *Naval Research Logistics*, 41:913–933, 1994.

[77] Singubabu Y. *Scheduling of jobs on single machine to minimize total cost.* Concordia University, Montreal, 1996.

[78] Taeyong Y., Zesheng H., and Kyukab C. Effective heuristic method for generalized job shop scheduling with due dates. *Computers and Industrial Engineering*, 26:647–660, 1994.

[79] Wang T. Y. and Wu K. B. Common due-date determination and sequencing using tabu search. *Computers and Operations Research*, 26:665–678, 1999.

[80] He Z., Yang T., and Deal D. E. A multipass heuristic rule for job shop scheduling with due dates. *International Journal of Production Research*, 31:2677–2692, 1993.