

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]



**JOBS SCHEDULING ON SINGLE MACHINE
UNDER VARIOUS MAINTENANCE
POLICIES**

BY

SYED ASIF RAZA

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE
In
SYSTEMS ENGINEERING

MAY 2002

UMI Number: 1411245

UMI[®]

UMI Microform 1411245

Copyright 2003 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS
DHAHRAN 31261, SAUDI ARABIA
DEANSHIP OF GRADUATE STUDIES

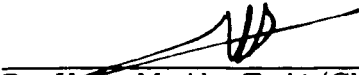
This thesis, written by


SYED ASIF RAZA

under the direction of his Thesis Advisor and approved by his Thesis Committee,
has been presented to and accepted by the Dean of Graduate Studies, in partial
fulfillment of the requirements for the degree of

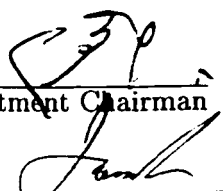
MASTER OF SCIENCE IN SYSTEMS ENGINEERING

Thesis Committee


Dr. Umar M. Al - Turki (Chairman)

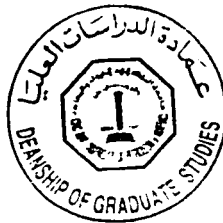

Dr. Salih Osman Duffuaa (Member)


Dr. Mohammed BenDaya (Member)


Department Chairman

Dean of Graduate Studies
(Prof. Osama Ahmed Jannadi)

29/7/2002
Date



Heartily dedicated to my family and especially to

my dear mother

whose prayers, love, and guidance
led to this accomplishment.

Acknowledgements

All praise to Allah, the most Beneficent and the most Merciful, who enabled me to complete my thesis work. I make a humble effort to thank Allah for his endless blessings on me, as His infinite blessings cannot be thanked for. Then, I pray Allah to bestow peace on his last prophet Muhammad (*Sal-allah-'Alaihe-Wa-Sallam*) and on all his righteous followers till the day of judgement.

I pay a heartily tribute to all of my family members and especially to my parents, who guided me during all my life endeavors. Their love and support motivated me to continue my education and achieve higher academic goals. Without their moral support and sincere prayers, I would have been unable to accomplish this task.

Next, I am deeply grateful to my thesis advisor Dr. Umar M. Al-Turki for his valuable guidance throughout my thesis work. At the same time, gratitude is due to my thesis committee members Dr. Salih Osman Duffuaa and Dr. Mohammed. Benda. I express my tribute to Dr. Shokri Zaki Selim for his help during my thesis work and moral support from him. I express my thanks to all of them for their valuable time and support.

I acknowledge the academic and computing facilities provided by the Systems Engineering Department of King Fahd University of Petroleum & Minerals (KFUPM).

Finally, I appreciate the friendly support from all my colleagues at KFUPM. In particular, I want to thank Atif, Moin, Mehmood, Zeeshan, Shafayat, Wasif, Aamir, Sajid, Saad, Ahmer, Junaid, Mr. Zia and Mr. Mahmood.

Contents

Acknowledgements	iv
List of Tables	ix
List of Figures	xii
Abstract (English)	xv
Abstract (Arabic)	xvi
1 Introduction	1
1.1 Problem Definition	5
1.2 Objectives of Thesis	8
1.3 Organization of Thesis	9
2 Literature Review	11
3 Total Completion Time Minimization	19

3.1	Introduction	19
3.2	Branch and Bound	20
3.3	Tabu Search	24
3.3.1	Tabu Preliminaries	25
3.3.2	Working Mechanism	28
3.4	Tabu Search Implementation	30
3.4.1	Initial Solution	30
3.4.2	Neighborhood Generation	30
3.4.3	Tabu List	30
3.4.4	Aspiration Criterion	32
3.4.5	Stopping Criterion	32
3.4.6	Algorithmic Description	32
3.5	Parameter Selection for TS	34
3.6	Simulated Annealing	35
3.6.1	Initial Solution	38
3.6.2	Neighborhood Generation	38
3.6.3	Cooling Schedule	38
3.6.4	Markov Chain Length	41
3.6.5	Stopping Criterion	42
3.6.6	Parameter Selection for SA	42
3.7	Proposed Lower Bound	43

3.8	Results	45
3.8.1	Comparison of TS and SA With BNB	45
3.8.2	Comparison of TS, SA With SPT	46
3.8.3	Deviation of TS, SA and SPT From Lower Bound	51
3.9	Discussion	53
4	Scheduling With Delayed Maintenance Operations	59
4.1	Assumptions	60
4.2	Notations	61
4.3	Properties of Optimal Schedule	61
4.4	Proposed Heuristic	71
4.5	Implementation of TS and SA	75
4.6	Proposed Lower Bound	75
4.7	Results	77
4.7.1	Comparison of Heuristic HSTC with TS and SA	77
4.7.2	Deviation of TS, SA and HSTC from Lower Bound	80
4.8	Discussion	83
5	Early-Tardy Minimization	86
5.1	Assumptions	86
5.2	Notations	87
5.3	Properties of Optimal Schedule	87

5.4	Proposed Heuristic	93
5.5	Implementation of TS and SA	94
5.6	Proposed Lower Bound for Early Tardy Minimization	98
5.7	Results	100
5.7.1	Comparison of Heuristic HSET with TS and SA	100
5.7.2	Deviation of TS, SA and HSET from Lower Bound	104
5.8	Discussion	107
6	Summary, Conclusion and Future Work	112
6.1	Summary	112
6.2	Future Work	114
A	DOE for Setting TS Parameters w.r.t Relative Improvement	116
B	DOE for Setting TS Parameters w.r.t CPU Time	120
C	DOE for Setting SA Parameters w.r.t Relative Improvement	124
D	DOE for Setting TS Parameters w.r.t CPU Time	128
E	DOE for Setting TS Parameters w.r.t Relative Improvement	132
F	Tables	136
	BIBLIOGRAPHY	145

List of Tables

3.1	Comparison of TS, SA & BNB for $n = 9$, $T = 40$	46
3.2	Comparison of SPT heuristic with TS and SA for $n = 20$	47
3.3	Comparison of SPT heuristic with TS and SA for $n = 35$	48
3.4	Comparison of SPT heuristic with TS and SA for $n = 40$	49
3.5	Comparison of SPT heuristic with TS and SA for $n = 50$	50
3.6	Percentage relative deviation of SPT, TS and SA from lower bound. .	51
3.7	Percentage relative deviation of SPT, TS and SA from lower bound. .	52
3.8	Overall comparison of TS, SA and BNB with different number of jobs.	53
4.1	Improvement made by TS and SA over HSTC for $n = 20$	77
4.2	Improvement made by TS and SA over HSTC for $n = 30$	78
4.3	Improvement made by TS and SA over HSTC for $n = 35$	79
4.4	Improvement made by TS and SA over HSTC for $n = 40$	80
4.5	Percentage relative deviation of HSTC, TS and SA from lower bound.	81
4.6	Percentage relative deviation of HSTC, TS and SA from lower bound.	82

4.7	Effect of $\frac{t_2}{t_1}$ and $\frac{T_{max}}{T_p}$ on the percentage relative deviation from lower bound.	84
4.8	Effect of $\frac{t_2}{t_1}$ and $\frac{T_{max}}{T_p}$ on the percentage relative error of HSTC. . . .	85
5.1	Average improvement made by TS and SA over HSET for $n = 10$. . .	101
5.2	Average improvement made by TS and SA over HSET for $n = 20$. . .	102
5.3	Average improvement made by TS and SA over HSET for $n = 35$. . .	103
5.4	Average improvement made by TS and SA over HSET for $n = 50$. . .	104
5.5	Average deviation of TS, SA and HSET from lower bound for $n = 10$.	105
5.6	Average deviation of TS, SA and HSET from lower bound for $n = 20$.	106
5.7	Average deviation of TS, SA and HSET from lower bound for $n = 35$.	106
5.8	Average deviation of TS, SA and HSET from lower bound for $n = 50$.	107
F.1	Comparison of TS, SA & BNB for $n = 9, T = 50$	136
F.2	Comparison of TS, SA & BNB for $n = 9, T = 60$	137
F.3	Comparison of TS, SA & BNB for $n = 9, T = 70$	137
F.4	Comparison of TS, SA & BNB for $n = 9, T = 80$	138
F.5	Comparison of TS, SA & BNB for $n = 9, T = 90$	138
F.6	Comparison of TS, SA & BNB for $n = 10, T = 40$	139
F.7	Comparison of TS, SA & BNB for $n = 10, T = 50$	139
F.8	Comparison of TS, SA & BNB for $n = 10, T = 60$	140
F.9	Comparison of TS, SA & BNB for $n = 10, T = 70$	140

F.10 Comparison of TS, SA & BNB for $n = 10$, $T = 80$	141
F.11 Comparison of TS, SA & BNB for $n = 10$, $T = 90$	141
F.12 Comparison of TS, SA & BNB for $n = 11$, $T = 40$	142
F.13 Comparison of TS, SA & BNB for $n = 11$, $T = 40$	142
F.14 Comparison of TS, SA & BNB for $n = 11$, $T = 40$	143
F.15 Comparison of TS, SA & BNB for $n = 11$, $T = 40$	143
F.16 Comparison of TS, SA & BNB for $n = 11$, $T = 50$	144
F.17 Comparison of TS, SA & BNB for $n = 11$, $T = 50$	144
F.18 Comparison of TS, SA & BNB for $n = 11$, $T = 50$	144

List of Figures

3.1	Tabu search algorithm (Sait et al. [1]).	27
3.2	Tabu search flow chart.	29
3.3	Proposed neighborhood scheme (NBR 1).	31
3.4	Insert maintenance in sequence (Contd. NBR 1).	31
3.5	Simulated annealing algorithm (Sait et al. [1]).	37
3.6	Simulated annealing algorithm flow chart.	39
3.7	Proposed Lower Bound (LB 1) scheme for total completion time minimization.	44
3.8	CPU time Comparison of TS and SA with BNB.	54
3.9	Effect of Number of jobs n on performance.	55
3.10	Effect of maintenance time t on performance.	56
3.11	Effect of maintenance time t on SPT performance.	56
3.12	Effect of Continuous working time on performance.	57
3.13	Effect of Continuous working time on SPT performance.	58

4.1	SPT property with each batch.	62
4.2	SPT property on batch.	65
4.3	Batch B_i followed by t_1 and B_{i+1} followed by t_2	67
4.4	Batch B_i followed by t_2 and B_{i+1} followed by t_1	70
4.5	Proposed Heuristic HSTC.	72
4.6	Cost of Option 1 (Contd. HSTC).	73
4.7	Cost of Option 2 (Contd. HSTC).	74
4.8	Insert job in batch (Contd. HSTC).	74
4.9	Proposed neighborhood scheme (NBR 2).	76
4.10	Effect of Number of jobs (n) on performance.	83
5.1	Proposed heuristic HSET.	94
5.2	Insert maintenance in tardy set (Contd. HSET).	95
5.3	Insert maintenance to early set (Contd. HSET).	95
5.4	Proposed neighborhood (NBR 3) for Early-Tardy minimization.	97
5.5	Algorithm for scheme 1 (Contd. NBR 3).	98
5.6	Algorithm for scheme 2 (Contd. NBR 3).	98
5.7	Proposed lower bound for Early -Tardy minimization.	99
5.8	Effect of number of jobs on performance.	108
5.9	Effect of maintenance time t on performance.	109
5.10	Effect of maintenance time t on HSET error.	110

5.11 Effect of continuous working time T on performance.	111
5.12 Effect of continuous working time T on HSET error.	111

THESIS ABSTRACT

Name: SYED ASIF RAZA
Title: JOBS SCHEDULING ON SINGLE MACHINE
UNDER VARIOUS MAINTENANCE POLICIES
Major Field: SYSTEMS ENGINEERING
Date of Degree: MAY 2002

In this thesis, scheduling jobs on a single machine that is subjected to preventive maintenance is considered. Three models that differ in maintenance policies and performance measures are studied. Two scheduling performance measures mean completion time and total earliness and tardiness are studied. For each model, the properties of the optimal schedule are identified and a simple heuristic that satisfies these properties is constructed. Tabu search, Simulated annealing algorithms and Lower bounds are also developed for each model. Experimentation showed that the results obtained by Tabu search and Simulated annealing are less than 1 percent of optimal solution and the performance of simple heuristics and Lower bounds is sensitive to the maintenance policy.

MASTER OF SCIENCE DEGREE

King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia.

MAY 2002

خلاصة الرسالة

الاسم : سيد أصف رضا
عنوان الرسالة : " جدولة الأعمال على آلة مفردة تحت عدة سياسات للصيانة "
التخصص : هندسة النظم
تاريخ الشهادة : مايو 2002م.

يعني هذا البحث بدراسة وتحليل مسألة جدولة عدة وظائف على مكيئة واحدة يتم عليها صيانة وقائية. تم دراسة ثلاثة نماذج جدولة تختلف في استراتيجية الصيانة وقياس الأداء. وتم قياس الأداء عن طريق متوسط زمن جدولة جميع الوظائف ومقدار التبدير والتأخير عن موعد محدد. وتم في هذا البحث تحديد خصائص الجدول الأمثل لكل نموذج. ومن ثم طورت خوارزمية تحقق تلك الخصائص كما تم تطوير خوارزميات بحث مستقراء من طريقة التابو (Tabu Search) والأنيلىق (Simulated Annealing) ودلت التجارب أن حلول خوارزميات التابو والأنيلىق تبعد أقل من 1% من الحل الأمثل.

درجة ماجستير في العلوم
جامعة الملك فهد للبترول والمعادن
الظهران - المملكة العربية السعودية
مايو 2002 هـ

Chapter 1

Introduction

Sequencing and scheduling are forms of decision-making which play a crucial role in manufacturing as well as in service industries. In the current competitive environment, effective sequencing has become a necessity for survival in the market. Companies have to meet shipping dates committed to the customers, as failure to do so may result in a significant loss of good will. They also have to schedule activities in such a way as to use the resources available in an efficient manner.

Scheduling concerns the allocation of limited resources to task over time. It is decision making process that has a goal to optimize one or more objectives.

The resources may take any forms. They may be machines in a workshop, runways at an airport, crews at a construction site, processing units in a computing environment, and so on. Tasks may be operations in a production process, take-offs and landing at an airport, stages in a construction project, execution of computer

programs, and so on. Each task may have different priority level, earliest possible starting time, and due date. The objectives may also take many forms. One possible objective is the minimization of completion time of the last task, and another is the minimization of the number of tasks completed after the committed due dates.

Scheduling exists in most manufacturing and production systems as well as in most information-processing environments. It also exists in transportation and distribution settings and in other types of service industries.

Scheduling can be difficult from both a technical and an implementation point of view. The type of difficulties encountered in the technical aspects are similar to the difficulties encountered in other branches of combinatorial optimization and stochastic modeling. The difficulties encountered on the implementation side are of a completely different kind and are related to the real-world scheduling problems and retrieval of information.

In a manufacturing system orders have to be released and have to be translated into jobs with associated due dates. The jobs often have to be processed by the machines in workcenters in a given sequence. Jobs may have to wait for processing on machines that are busy. Detailed scheduling of the tasks to be performed in a production system is necessary to maintain efficiency and control of operations.

The standard assumptions in scheduling according to French [2] and Pinedo [3] are:

1. No pre-emption is allowed.
2. No cancellation.
3. The processing times are independent of the schedule.
4. Machine may be idle.
5. No machine may process more than one operation at a time.
6. *Machine never breakdown and are available throughout the scheduling period.*
7. The technological constraints are known in advance and are immutable.
8. There is no randomness. In particular
 - (a) The number of jobs is known and fixed.
 - (b) The processing times are known and fixed.
 - (c) The ready times are known and fixed.
 - (d) All other quantities needed to define a particular problem are known and fixed.

In the problem considered in this thesis all standard scheduling assumptions are applicable except assumption number 6. In the scheduling problem the machine may not be available over the whole scheduling period. A machine may become unavailable during the production process due to preventive maintenance, which is

scheduled in advance, and/or breakdowns, which happen randomly. Only recently has the study of scheduling problems concerning machine unavailability carried out.

The problem stated by X. Qi et al. [4] has become relevant and important due to the development of computer aided production planning systems and automated manufacturing. Automated systems cause high yield but the investment in automated systems demand higher availability and longer life cycle and must be kept at higher production rate. This can be achieved by incorporating an efficient maintenance policy. In this situation the jobs and maintenance planning has to be done simultaneously, so it becomes machine scheduling problem that includes machine unavailability.

The motivation for studying deterministic unavailability (machine vacation) comes from production practice. For example, a machine may be designed to be examined, refueled or maintained after working for a period of time. The problem encountered belongs to such a case. A machine may also be scheduled to do some other special works in certain time intervals and thus becomes unavailable. So machine maintenance is a sort of unavailability.

1.1 Problem Definition

Assume that there are n jobs ready at time zero to be processed on a single machine for which the maintenance has to be done along with jobs processing. It is required to schedule these jobs so that certain objective function is minimized. The machine is needed to be maintained before completing some predetermined time T of continuous operation. Several objective functions may be of interest such as mean completion time, makespan, and some other functions related to predetermined due dates.

Suppose there are n jobs $J_1, J_2, J_3, \dots, J_n$ to be processed on a single machine. The processing time of job J_i is p_i .

The following assumptions describe the problem.

1. The jobs are indexed as $p_1 \leq p_2 \leq p_3 \leq \dots \leq p_n$.
2. All jobs are available at time zero.
3. The maximum allowed continuously working time of the machine is T and the maintaining time is t .
4. T is assumed to be more than p_n .
5. Processing of a job can not be interrupted by processing another job or a maintenance operation.

The Schedule π contains a sequence of jobs and the maintenance has to be inserted in between the jobs such that the total continuous processing time of the

machine does not exceed the time period T . M represents maintenance in schedule. The preventive maintenance time inserted in job sequence in the span 0 to T . In a schedule, jobs processed continuously from a batch, denoted as B . Thus a schedule π can be denoted as $\pi = (B_1, M, B_2, M, \dots, M, B_L)$, where M represents preventive maintenance in π , which is deterministic and known, and L is the number of batches. The problem is to determine an optimal schedule that contains order of jobs and maintenance operations in the schedule along with jobs.

Let $J_{[i]}$ be the i th job in a schedule, C_i be the completion of job J_i , then, $C_{[i]} = \sum_{k=1}^i p_{[k]} + (m_{[i]} - 1)t$ where $m_{[i]}$ means $J_{[i]}$ is in batch $B_{m_{[i]}}$. Let q_i be total processing time of jobs in batch B_i and n_i be the number of jobs in batch B_i .

Two important objective functions concerning completion time are the maximum completion time of jobs C_{max} and total completion time. For an optimal schedule under minimization of C_{max} it must contain minimum number of batches, hence it is minimizing the maintaining time. The C_{max} problem is NP-hard in the strong sense [5]. X. Qi et al. [4] discussed this problem under the objectives of minimization of total completion times ($\sum C_i$). The problem is also shown to be NP-hard in strong sense. The objective function can be written as

$$f(\pi) = \sum_{i=1}^n C_i$$

$$\begin{aligned}
&= \sum_{i=1}^n (p_{[1]} + \dots + p_{[i]} + (m_{[i]} - 1)t) \\
&= \sum_{i=1}^n (n - i + 1)p_{[i]} + \sum_{i=2}^L (i - 1)n_i t
\end{aligned}$$

The $f(\pi)$ contains two parts in the total completion time, the processing times $\sum_{i=1}^n (n - i + 1)p_{[i]}$ and the maintenance time $\sum_{i=2}^L (i - 1)n_i t$.

The properties of the optimal schedule studied by X Qi et al. [4] are:

1. In each batch of the optimal schedule jobs are sequenced according to *SPT* rule.
2. In an optimal schedule, $n_1 \geq n_2 \geq n_3 \geq \dots \geq n_L$.
3. In an optimal schedule,

$$\frac{q_i}{n_i} \leq \frac{q_{i+1}}{n_{i+1}}, \quad i = 1, 2, \dots, L - 1 \quad (1.1)$$

Under the minimization of the $\sum C_i$ the minimum number of batches may not be the optimal solution. This can be shown by the following example taken from X. Qi et al. [4].

Let $n = 4$, $p_1 = p_2 = 1$, $p_3 = p_4 = 4$, $T = 5$. The schedule π_1 with the minimum number of batches is $\pi_1 = (J_1, J_3, M, J_2, J_4)$ with $L_1 = 2$ and $f(\Pi_1) = \sum_{i=1}^n C_i = 22 + 2t$. Now consider the schedule $\pi_2 = (J_1, J_2, M, J_3, M, J_4)$ with $L_2 = 3$ and $f(\Pi_2) = \sum_{i=1}^n C_i = 19 + 3t$. When $t < 3$, we have $f(\pi_2) < f(\pi_1)$ and π_2 is optimal.

X. Qi et al. [4] proposed Shortest Processing Time Heuristic *SPT*, Fewest Batches Heuristic *FBH* and a Branch and Bound algorithm. The problem discussed by X. Qi et al. [4] could be extended in another direction. The suggested extension is to divide the continuous working time limit of machine into two categories preferred time T_p and maximum allowed time T_{max} . If the machine is maintained within T_p the maintenance time will be t_1 , if it exceeds T_p then the maintenance time will be t_2 units.

Other extensions to the problem defined by X. Qi et al. [4] could be by considering other measures of performance such as minimizing total earliness and tardiness.

1.2 Objectives of Thesis

The objectives of this thesis are following:

1. Develop iterative heuristic algorithm(s) (Tabu search, Simulated annealing), and compare the results of the iterative heuristic algorithm(s) with the results of Branch and Bound algorithm proposed by X. Qi et al. [4] for the problem with objective of minimizing total completion time.
2. Extend the work of X. Qi et al. [4] by considering the objective function of minimizing the total completion time where continuous working time of machine is divided into categories (see Section 1.1) preferred to do maintenance T_p and maximum time to do maintenance T_{max} with respective maintenance

time t_1 and t_2 . The proposed work to this extension is as follows:

- (a) Study the properties of optimal schedule.
 - (b) Propose a heuristic based on the properties developed.
 - (c) Develop a Tabu search (TS) and a Simulated annealing (SA) algorithms for the problem.
 - (d) Compare the proposed heuristic with TS and SA.
3. Extend the work of X. Qi et al. [4] by considering the objective function of minimizing total Earliness and Tardiness about a common due date. Then perform the following:
- (a) Study the properties of optimal schedule.
 - (b) Propose a heuristic based on the properties developed.
 - (c) Develop a Tabu search (TS) and a Simulated annealing (SA) algorithms to the problem.
 - (d) Compare the proposed heuristic with TS and SA.

1.3 Organization of Thesis

The first chapter of this thesis is used to introduce and define the problem, state the thesis objective and outline the thesis organization. The second chapter covers the

selected literature in the areas of scheduling with maintenance, iterative algorithms, TS and SA. In the third chapter the Tabu search implementation is shown under the objective of minimizing the total completion time. In this chapter the parameters of the tabu search algorithm were also tuned using the General factorial design of experiments. This Chapter also discusses the Simulated annealing implementation (SA) for total completion time minimization, The SA parameters are also tuned in this chapter using general factorial design (DOE). Implementation of Branch and Bound Algorithm (BNB) proposed by X Qi et al. [4] is discussed and computational experience is presented with BNB, SA and TS. Chapter 4 discusses the proposed extension to X Qi et al. [4] where the continuous working time limit of machine is divided into two categories (see Section 1.1). Properties of the optimal schedule are developed followed by t heuristic, Tabu search and Simulated annealing approach to the problem. Chapter 5 is about total earliness and tardiness minimization about a common due date as explained in thesis objectives (see Section 1.2). Chapter 6 concludes the thesis with summary and future research directions.

Chapter 2

Literature Review

This chapter covers the literature reviewed related to the problem under maintenance scheduling, iterative heuristics (Tabu search and Simulated annealing), and early-tardy minimization about a common due date.

In the first part, the literature regarding scheduling with machine unavailability is presented.

Schmidt [6] studied parallel machine preemptive scheduling problem where each job has a deadline and each machine has different availability intervals. An algorithm to find a feasible schedule is presented.

Kenneth and Scudder [7] reviewed the literature regarding basic models that contain symmetric penalties, one machine and a common due date. On these basis they added features, like parallel machines, complex penalty functions and distinct due dates.

Adiri et al. [8] considered the scheduling of jobs on single machine with the objective function of minimization of flow time. The machine is supposed to fail during the processing of jobs. The time for break down was random variable. The machine is unable to process the jobs till it is repaired. They showed that in the case of single break down if the failure distribution is concave then *SPT* minimizes the flow time. In the case of multiple break downs *SPT* also minimizes the flow time, but with exponential distribution of failure. When the time for single break down and processing times are known the problem is *NP* complete. They also calculated lower bounds for performance of *SPT* under deterministic case.

Lee and Liman [9] investigated the problem of minimizing the total flow time when machine is subjected to scheduled maintenance. They showed the problem to be *NP*-hard. They proposed a tight bound for *SPT* sequence at a value of $\frac{2}{7}$. This bound is shown to be better than Adiri [8].

Narashims and Addagatla [10] Incorporated early-tardy penalty and machine vacation (unavailability) and developed a model. They proposed heuristic methods for solving the early-tardy minimization problem and presented computational results. The time (position) of the machine vacation was deterministic and fixed. In **X Qi et al.** [4] the machine vacation is non deterministic.

Lee [11] studied the parallel-machine scheduling problem of minimizing the makespan where machines may not be available at zero ready time.

X. Qi et al. [4] studied single machine scheduling with preventive maintenance. In

many cases, a machine must be maintained after it continuously works for a period of time, but most of the papers neglect the non availability of the machine. In this paper scheduling of jobs and the preventive maintenance was simultaneous. The machine was required to be maintained in the interval $(0, T]$. The preventive maintenance time t was deterministic and known. They studied the properties of the optimal schedule and proposed two heuristic algorithms and the branch and bound to solve the problem.

Mazzini and Amentano [12] published on single machine scheduling problem with due time constraints, ready time constraints and shut down constraint on the machine. The shut down is disruptive event such as holidays, breaks or machine maintenance, and has a pre-specified period when the machine will be interrupted. No tardy jobs are allowed, and if a job is completed earlier than its due time, then shop holds the job and incurs the holding cost for earliness. They proposed a heuristic algorithm which manipulates the starting and completion times of the shut down period so as to minimize the sum of the holding cost and the reduction cost in the shut down times.

The above problems assume that the maintenance of machine is done in a fixed time interval known before and study how to schedule jobs under this constraint of machine unavailability.

Lee and Chen [13] extended the work of X Qi et al. [4] for parallel machine with the objective of weighted total completion time minimization. Two case studies

were made. In first case, there was sufficient resources so that different machines could be maintained if necessary. In the second case, only one machine could be maintained at any given time. The problem was proved to be NP-hard. Branch and Bound algorithm with column generation technique was proposed to solve the problem optimally.

We will now discuss the literature in area of iterative heuristic approaches to scheduling problems.

Barnes and John [14] attempted to minimize the makespan where a set of machines perform technologically ordered operations unique to each member of a set of jobs. In the paper an effective Tabu search approach to job shop scheduling problem was discussed. The procedure starts from the best solution rendered by a set of 14 heuristic dispatching solutions. They used classical disjunctive network representation of the problem and iterative moves to another feasible solution is obtained by reversing the order of two adjacent critical path operations performed by the same machine. They made computational study and proposed some future research directions.

Bargila and Melloni [15] considered the reduction of mean completion time and makespan in the single machine scheduling problem with ready times. A Tabu search based heuristic procedure was proposed to find the sequence that allows to reach a value of the cost function as close as possible to the global minimum. Computational tests were carried out to compare the performance of the Tabu search

procedure with other heuristics. Comparison were made both in terms of mean completion time and makespan.

Colin Reeves [16] studied the problem of scheduling jobs on a single machine where the jobs have unequal release time. They proposed a heuristic and implement Tabu search to study the quality of the solution generated by the heuristic.

Tsallis and **Stariolo** [17] discussed Simulated annealing algorithm for computationally finding the global optimal of a given (not necessarily non-convex) energy cost function. They showed that Simulated annealing recovers optimal faster.

Ben-Daya and **Al-Fawzan** [18] proposed a Simulated annealing approach for solving the single machine mean tardiness scheduling problem. The results on simulation indicated that their proposed method provides much better solution than two efficient heuristics. They proved that the solution obtained by Simulated annealing is less than 1 % of optimal solution.

Al-Fawzan and **Al-Sultan** [19] developed Tabu search approach to job shop scheduling problem for the minimization of the makespan.

Lyu et al. [20] selected single machine early-tardy problem to demonstrate the usefulness of Simulated annealing algorithm. Based on the previous studies they used factorial experiments to analyze the factors that are critical to the efficiency of Simulated annealing. The resulting algorithm is compared with Branch and Bound algorithm and Neighborhood search methods by solving test problems. The results showed that Simulated annealing is very sensitive to cooling rate, generation mech-

anism, acceptance criteria and stopping criteria. For small size problems Simulated annealing was compared with Branch and Bound, it was found to produce same results. For large problems it was compared with Neighborhood search and it produced much better solution than Neighborhood search.

Xiaoyaun and Hailong [21] studied the scheduling problem on a single machine in a job shop production system. A general scheduling model for multiobjective on a single machine was proposed. Computational experiments were performed using Simulated annealing.

Torres et al. [22] studied the scheduling problem where the objective is to minimize both the number of late jobs and the average flow-time. Heuristics based on Simulated annealing and Neighborhood search were proposed with the objective to compromise between two criteria. Simulation experiments showed that heuristics perform well when compared with the optimal solution for small size problems.

Marangos et al. [23] proposed heuristic methods to minimize the variance of completion time when jobs are to be processed by two parallel machines. All the jobs visit the machine in the same order or technological sequence. The solutions were compared with the lower bound and found to be promising. A simulated annealing approach was also developed and it was found that it produces optimal solution for all problems upto job size 11.

Vinicius and Debora [24] presented Tabu search approach to minimize total tardiness of the job shop scheduling problem. The method uses dispatching rules to

obtain an initial solution and searches for new solution in a neighborhood based on the critical path of the jobs. They compare the quality of the solution produced by Tabu search with optimal solution for small size problems and for large size problems, Tabu search was compared with heuristics proposed in similar literature.

Islam and **Eksioglu** [25] proposed a Tabu search approach for solving the single machine mean tardiness scheduling problem. Simulation experiments obtained from Tabu search approach and other three approaches were compared. Although computation time is increased but the results indicated that the proposed heuristic provides a much better solution than the other three approaches.

Yi and **Wang** [26] studied a job scheduling model of identical parallel machines. The model assumes that a setup time was incurred when a machine changes from processing one type of parts to a different type of part, and the scheduling objective was to minimize the sum of total flow time. The problem was proved as NP-hard. Two methods were developed, comparison of the solutions showed that Tabu search combined with heuristic algorithm is more reliable and has abilities to solve larger scale practical problems.

The following section discusses the literature in early-tardy minimization about a common due date.

Kanet [27] considered single machine scheduling problem in which penalty occurs when jobs are completed early or late. The objective is to minimize total penalty subject restrictive assumptions on the due dates and penalty function for jobs. A

procedure was proposed to find the optimal schedule.

Kanet [28] addressed the problem of scheduling jobs on single machine in such a way that flow time variation is minimized. He showed that when the measure of variation is the variance of flow time the problem is much more difficult. For this case a heuristic method for scheduling was proposed.

Sundraraghvan and **Mesbah** [29] extended the Kanet's work [27] for identical parallel machines. They proposed a heuristic algorithm for single machine absolute lateness minimization with no restriction on common due date.

Szwarc [30] considered the single machine scheduling problem to minimize the sum of absolute lateness given a common due date. They proposed two models depending on whether the start time of schedule is arbitrary or fixed. They provided the conditions where both models coincide and developed branch and bound procedure.

Gupta et al. [31] proposed a heuristic based on complementary pair-exchange principle for minimization flow time variance on single machine. It was concluded that the heuristic is superior to other heuristics. The complexity of the proposed heuristic found to be $O(n \log n)$.

Leon and **Wu** [32] studied single machine scheduling problem in which there are ready time and due date constraint on the machine. Each vacation has fixed starting and finishing time, without allowing preemption. The objective was to minimize maximum lateness. They obtained a relaxation of the problem by modeling the vacations as a set of jobs with flexible ready times and artificial due dates.

Chapter 3

Total Completion Time

Minimization

3.1 Introduction

In this chapter Branch and Bound algorithm proposed by X. Qi et al. [4] is implemented for solving the problem. Then a Tabu search (TS) algorithm and a Simulated annealing (SA) algorithm will be developed for the problem. The three methods will be compared to each other along with SPT heuristic proposed by X. Qi et al. [4].

3.2 Branch and Bound

The Branch and Bound (BNB) algorithm is an implicit enumeration technique. Apart from heuristics the Branch and Bound is probably the most widely used solution technique in scheduling. Like Dynamic programming, it is an enumeration technique, which converges to optimality. A tree structure is used in this technique. Branches consist of nodes which contain either a semi finished schedule or a complete schedule. For every node, a Lower Bound (LB) is calculated which consists of actual cost of known job sequence in a node and ideal cost of remaining unknown part. The tree enormously grows as the number of jobs increase. A better lower bounding criteria should be used to prune the tree efficiently. Branch and Bound becomes computationally prohibitive for medium size problems, and intractable for large size problems. Some trial solution from an appropriate heuristic could also be used to prune the tree as an Upper Bound (UB), so that any node with lower bound value greater than upper bound is fathomed.

Now we will discuss the implementation issues for Branch and Bound algorithm presented by X.Qi et al. [4]. For any node D , the total number of jobs n are separated into two groups: jobs scheduled and jobs unscheduled. Suppose n_D jobs have been scheduled and occupied positions from 1 to n_D in node D .

For any job J_j in S'_D , a new node D_j can be reached from node D by placing job J_j at the end of S_D i.e., at the position of $n_D + 1$. The following two rules are

- S_D = the partial schedule on n_D scheduled jobs.
 S'_D = the set of unscheduled jobs.
 pt_1 = total processing time in the second last batch of S_D .
 pn_1 = total number of jobs in the second last batch of S_D .
 pt_2 = total processing time in the last batch of S'_D .
 pn_2 = total number of jobs in the last batch of S'_D .

very useful in the tree pruning.

Rule 1

if $pt_2 + p_j \leq T$, then D_j should be eliminated if one of the following conditions is satisfied

1. $p_j < p_{slb}$ here slb is the set of job in last batch.
2. $pn_1 = pn_2$.
3. $pn_1 = pn_2 + 1$ and $pt_1 > pt_2 + p_j$.

Rule 2

If $pt_2 + p_j \leq T$, then D_j should be eliminated if one of the following conditions is satisfied.

1. $\frac{pt_1}{pn_1} > \frac{pt_2}{pn_2}$.
2. $pt_2 + p_{min} \leq T$, where p_{min} is the shortest processing time in S'_D .

The rules are derived from properties of the optimal schedule developed by X.

Qi et al. [4]. If for any node any of the two rules are not satisfied then the lower

bound (LB) of that node will be calculated. For a node D , total completion time of jobs can be divided into two parts, total completion time of S_D , which is known, and total completion time of S'_D , which has to be estimated. Let $f_1(D)$ be total completion time of S_D , then $f_1(D)$ can be computed directly

$$f_1(D) = \sum_{i=1}^{n_D} C_{[i]} \quad (3.1)$$

Given a partial schedule π' of S'_D , consider the contribution of processing time, $f_2(\pi')$, and the contribution of maintenance time, $f_3(\pi')$. Since π' starts at $C_{[n_D]}$, the completion time of S_D , we have

$$\begin{aligned} f_2(\pi') &= (n - n_D)C_{[n_D]} + (n - n_D)p_{[n_D+1]} \\ &+ (n - n_D - 1)p_{[n_D+2]} + \dots + p_{[n_D]} \end{aligned} \quad (3.2)$$

Obviously, when π' is an SPT schedule, $f_2(\pi')$ is minimal. We use SPT schedule of jobs in S'_D as lower bound of $f_2(\pi')$, denoted by $f_2(D)$, now

$$\begin{aligned} f_3(\pi') &= n'_2 \cdot t + n'_3 \cdot 2t + n'_4 \cdot 3t + \dots \\ &= (n - n_D - n'_1)t + n'_3 \cdot t + n'_4 \cdot 2t + \dots \\ &\geq (n - n_D - n'_1)t. \end{aligned} \quad (3.3)$$

Then $f_3(D) = (n - n_D - n'_1)t$ can be used as the lower bound of $f_3(\pi')$. Therefore, we have the lower bound of the node D as follows

$$f(D) = f_1(D) + f_2(D) + f_3(D) \quad (3.4)$$

The Branch and Bound algorithm is outlined as follows:

1. Use the total completion time of SPT heuristic as the upper bound of BNB.
2. Initialize the root node D such that, $S_D = \{\emptyset\}$, $S'_D = \{J_1, J_2, \dots, J_n\}$, $pt_1 = 0$, $pn_1 = 0$, $pt_2 = 0$, $pn_2 = 0$
3. If the search tree is already empty, then stop. Otherwise, select an unsearched node D .
4. If D is a leaf node, that is $S'_D = \{\emptyset\}$, then a complete schedule is obtained. If total completion time is smaller than the upper bound, update the upper bound and go to step 3.
5. For each job in S'_D , generate a new node by putting the job at the end of S_D . If the new node satisfies rule 1 or rule 2, eliminate it. Compute the lower bound of the new node. If the lower bound is greater than the upper bound eliminate it, otherwise keep this node in the search tree and go to step 3.

3.3 Tabu Search

Tabu search (TS) was introduced by Glover [33], [34] and [35] is an *iterative* heuristic for solving combinatorial optimization problems. Initial ideas of the technique were also proposed by Hansen in his steepest ascent mildest descent heuristic. Tabu search is conceptually simple and elegant, it has a powerful search capability for many combinatorial optimization problems. Tabu search is generalization of local search. At each step, the local neighborhood of the current solution is explored and the best solution in that neighborhood is selected as the new current solution. Unlike local search which stops when no improved new solution is found in the current neighborhood, tabu search continues the search from the best solution in the neighborhood even if it is worse than the current solution. To prevent cycling, information pertaining to the most recently visited solutions are inserted in a list called *tabu list*. Moves to tabu solutions are not allowed. The tabu status of a solution is overridden when certain criteria (aspiration criteria) are satisfied. One example of an aspiration criterion is when the cost of the selected solution is better than the best seen so far, which is an indication that the search is not cycling back, but rather moving to new solution not encountered before.

Tabu search is a *metaheuristic*, which can be used not only to guide search in complex solution spaces, but also to direct the operations of other heuristic procedures. It can be superimposed on any heuristic whose operations are characterized as per-

forming a sequence of *moves* that lead the procedure from one trial solution to another. In addition to several other characteristics, the attractiveness of tabu search comes from ability to escape local optima. There are many applications of Tabu search, but only recently it is applied to machine scheduling, employee scheduling, character recognition, telecommunication path assignment, quadratic assignment problem, graph coloring and partitioning problems, traveling salesman problem and VLSI placement.

3.3.1 Tabu Preliminaries

We will explain some phrases and terms used in Tabu search.

Trail solution: Trail solution is a solution generated from current solution as a result of a move.

Tabu restriction: A device to avoid cycling back to previously visited solutions by making selected attributes of moves tabu (forbidden). They allow the search to go beyond the points of local optimality.

Aspiration criterion: A device used to override the tabu status of a move whenever possible.

Candidate list size: The list containing the subset of neighborhood moves examined.

Attribute of a move: Any aspect (feature or component of a solution) that changes as a result of a move from current solution to trial solution can be attribute

of that move. A single move can have several attributes.

In Tabu search initially there is one trial solution, we call it starting solution (Seed solution). This solution is perturbed randomly. Several moves are made, and with each move a candidate solution is generated. Among the generated candidate solutions the best solution is selected. The best candidate solution is taken as current solution. To avoid the possibility of cycling this solution move is made tabu (forbidden). Basic algorithm for Tabu search presented is shown in Figure 3.1.

Algorithm *Tabu_Search*

Ω : Set of feasible solutions
S : Current solution
S* : Best solution
Cost: Objective function
N(S): Neighborhood of $S \in \Omega$
V* : Sample of neighborhood solutions
T : Tabu list
AL : Aspirartion level

Begin

Start with an initial feasible solution $S \in \Omega$

Initialize tabu list and aspiration level

For fixed number of iterations **Do**

 Generate neighbor solutions $V^* \subset N(S)$

 Find best $S^* \in V^*$

If move S to S^* is not in T **Then**

 Accept move and update best solution

 Update T and AL

Else

If $\text{Cost}(S^*) < \text{AL}$ **Then**

 Accept move and update best solution

 Update T and AL

End If

End If

End For

End.

Figure 3.1: Tabu search algorithm (Sait et al. [1]).

3.3.2 Working Mechanism

The Tabu search starts with an initial solution. The initial solution is the best solution at the start. The initial (current) solution is perturbed to obtain a set of new candidate solutions. The move which generates the best solution among the set of candidate solutions is selected. This move is checked in tabu list. If it is not in the tabu list then the best candidate solution becomes the current solution. If the best candidate solution is found in tabu list, its aspiration criteria is checked. If it passes the aspiration criteria, i.e., better than solutions visited so far, then it is also accepted. Hence aspiration criteria and best solution is updated. Otherwise more moves are generated to get another set of new candidate solutions and the process is repeated till termination. Detailed flow chart of Tabu search is given in Figure 3.2. Tabu restrictions allow the search to go beyond the points of local optimality while still making the best possible move in each iteration. The tabu list is initially empty, and it is constructed in number of iterations equivalent to the *tabu list size (TLS)*. A difficult part in the implementation of Tabu search is to find the right tabu list size. No single rule gives good sizes for all classes of problems. If the tabu list size is too small, the search process may start cycling and if it is too large, the search may be too restrictive. Therefore an appropriate list size has to be determined by noting the occurrence of cycling when the size is too small and the quality of solution when it is too large.

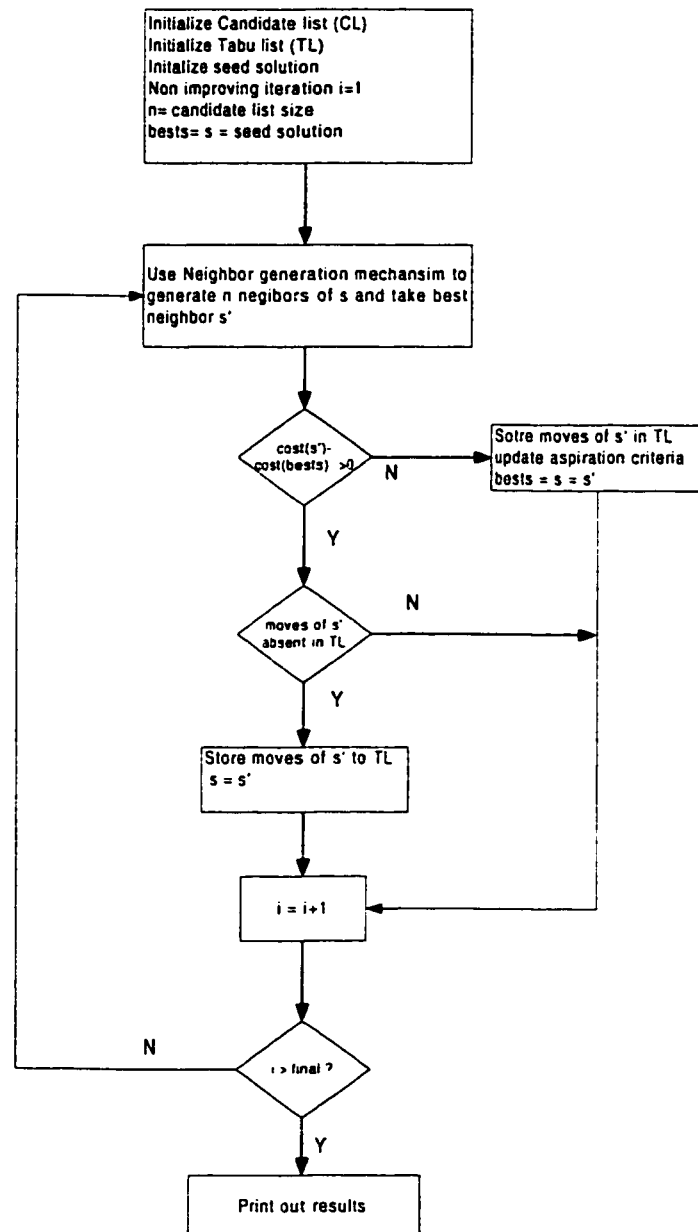


Figure 3.2: Tabu search flow chart.

3.4 Tabu Search Implementation

3.4.1 Initial Solution

Initial solution (Schedule) is randomly generated. The initial solution is the job sequence along with the preventive maintenance inserted in sequence.

3.4.2 Neighborhood Generation

The neighbor sequence is generated as follows.

Two jobs in two randomly selected batches are swapped randomly. The resulting sequence is checked for feasibility. In case the schedule is infeasible then feasibility is retained by a procedure given in Figure 3.4. The details of neighborhood generation are given in Figure 3.3. The quality of the final solution generated by TS may be dependent on the number of candidate (neighbor) solutions generated. An appropriate candidate list size works effectively to explore search space.

3.4.3 Tabu List

The attributes of a solution (Schedule) are recorded in tabu list. In our implementations attributes of a schedule are jobs swapped in a schedule and the positions of maintenance.

```

Algorithm NBR 1
  (*  $S$  = Schedule *)
  (*  $S'$  = Neighbor Schedule *)
  (*  $M$  = Maintenance *)
  (*  $p_j$  = Processing time of job  $j$ . *)
  (*  $S_i$  = Member of  $S$  (either Job or maintenance) at position  $i$ . *)
Begin
   $S' \leftarrow \{\emptyset\}$ ,  $B_j \leftarrow \{\emptyset\}$ ,  $i \leftarrow 1$ ,  $j \leftarrow 1$ .
  Repeat
    If ( $S_i = M$ ) Then
       $j \leftarrow j + 1$ 
    Else
      Insert job at  $S_i$  at the end of batch  $B_j$ 
    Endif
     $i \leftarrow i + 1$ 
  Until ( $i \leq |S|$ )
  Generate two unique random numbers  $a$ , and  $b$  with  $U(1, |j|)$ .
  Generate  $x$  and  $y$  with  $U(1, |B_a|)$  and  $U(1, |B_b|)$ .
   $S' \leftarrow S$ 
  Swap job at position  $x$  in batch  $B_a$  with job at position  $y$  in batch  $B_b$  in  $S'$ .
  Sort jobs in batches  $a$  and  $b$  in SPT order.
  If ( $S'$  is feasible) Then
    Return
  Else
    Call Insert_M ( $p, S, S'$ )
  Endif
END

```

Figure 3.3: Proposed neighborhood scheme (NBR 1).

```

Algorithm Insert_M ( $p, S, S'$ )
   $i \leftarrow 1$ ,  $j \leftarrow 1$ ,  $K \leftarrow \{\emptyset\}$ ,  $S' \leftarrow \{\emptyset\}$ 
  Repeat
    If ( $S_i \neq M$ ) Then
       $K_j \leftarrow S_i$ 
       $j \leftarrow j + 1$ 
    Endif
     $i \leftarrow i + 1$ 
  Until ( $i \leq |S|$ )
  Repeat
     $i \leftarrow 1$ ,  $q \leftarrow 0.0$ ,  $D \leftarrow \{\emptyset\}$ 
    Repeat
      Add job  $j_i$  at the end of  $D$ 
       $q \leftarrow q + p_i$ ,  $\forall i \in D$ 
    Until ( $q \leq T \parallel |K| = 0$ )
    Generate  $z \sim U(1, |D|)$  and remove the jobs beyond position  $z$  in  $D$ .
    Insert jobs in  $D$  at the end of  $S'$ 
    Insert  $M$  at the end of  $S'$ 
    Remove the jobs in  $D$  from  $K$ 
  Until ( $|K| \leq 0$ )
End

```

Figure 3.4: Insert maintenance in sequence (Contd. NBR 1).

3.4.4 Aspiration Criterion

The aspiration criteria used in the implementation is to override the tabu status of a forbidden (Tabu) solution. If the best neighbor solution of the current iteration is better than the global best solution. The global solution and aspiration level are updated.

3.4.5 Stopping Criterion

It is decided to run the algorithm for a fixed number of non improving iterations. The major reason for stopping after a fixed number of non improving iterations is that continuing beyond costs run time without any significant improvement in the results.

3.4.6 Algorithmic Description

The various steps in the proposed Tabu search algorithm are as following :

1. Initialize the tabu lists and iteration counter and assign the processing times to the job.
2. Arrange the jobs in ascending order of their processing times and index jobs in ascending order of processing time.
3. Generate a random initial solution.

4. The objective value of the current solution is taken as the best value, and thus the aspiration criteria is initialized.
5. The candidate solutions are generated by using the procedure described in Section 3.4.2.
6. The objective value of each candidate solution is calculated and the best candidate is selected.
7. If the solution of best candidate solution is better than the solution visited so far then it is taken as best solution and the aspiration criteria is updated. This solution becomes seed for next iteration.
8. The best candidate solution is checked in Tabu list. If the solution of best candidate solution is not better than best solution so far visited then if there is any move stored in the tabu list (see Section 3.4.3) then the move of best candidate is checked with tabu list.
9. If the move is not found in the tabu list then tabu list is updated with attributes of best candidate solution. The selected best candidate solution becomes the current solution for the next iteration.
10. The process is repeated until the stopping criteria is achieved.

3.5 Parameter Selection for TS

Initial experimentation was conducted to fine tune the Tabu search (TS). The General factorial design is used to observe the effect of parameters used in Algorithm.

The General factorial design (DOE) was performed with following factors

1. Number of jobs to be Scheduled (n). Three levels are selected at 10, 50 and 100.
2. Candidate list size (CLS). Two levels are selected at 10 and 20.
3. Tabu list size (TLS). Two levels are selected at 7 and 11.

18 replications for each of the 12 combinations is conducted with two different measures.

Relative improvement form starting seed solution:

It is the measure of improvement made by algorithm with respect to initial solution (seed solution) as follows

$$\text{Relative Improvement (RI)} = \frac{\text{Initial Cost} - \text{Best Cost}}{\text{Initial Cost}} \quad (3.5)$$

The General factorial design of experiment is done using IMSL libraries on FORTRAN 90 and the output of the program is given in Appendix A. None of the three factors found to be significantly affecting the performance of TS.

CPU time:

The similar design described above was used to determine the effect of problem size and parameters selection on CPU time, details of the experiment are given in Appendix B. It was found that all three factors were significant in the experiment. The CPU time increases as Number of jobs, Candidate list size and Tabu list size increase.

3.6 Simulated Annealing

Simulated annealing (*SA*) was proposed by Kirkpatrick et al.[36]. It is an iterative heuristic and belongs to the class of non deterministic algorithms. The term annealing refers to heating a metal to a very high temperature and then slowly cooling it down. If the cooling rate is selected appropriately, atoms of the molten metal have a greater chance to regain proper crystal structure. During this procedure, the free energy of the solid is minimized. Some issues related to SA were discussed by Eglese [37].

The SA algorithm simulates the annealing process of a metal. The algorithm starts with an initial solution, say state s_i , and a new state s_j is generated by applying a perturbation mechanism. If the energy E_j of state s_j is less than the energy E_i of state s_i , the move is accepted. Otherwise, the move from s_i to s_j may still be accepted probabilistically.

During annealing, a metal is maintained at a specific temperature T for a predetermined amount of time, before reducing the temperature in a controlled manner. At higher temperatures, atoms are more free to move, this analogous to the higher acceptance rate of uphill (bad) moves at higher temperature is SA. As the temperature goes down, the probability of accepting uphill moves becomes lower and at a absolute zero, the SA become greedy, accepting downhill (good) moves only. Hence the rate at which the temperature is lowered becomes critical parameter to tune so as to avoid premature convergence of the algorithm. The similar concept is applied to determine the optimal solution of several combinatorial optimization problems. Figure 3.5 shows outline of basic Simulated annealing algorithm.

```

Algorithm Simulated annealing ( $S_o, T_o, \alpha, \beta, M, Maxtime$ )
  (* $S_o$  is the initial solution *)
  (* $BestS$  is the best solution *)
  (* $T_o$  is the initial temperature *)
  (* $\alpha$  is the cooling rate *)
  (* $\beta$  is a constant *)
  (* $Maxtime$  is the total allowed time for the annealing process *)
  (* $M$  represents the time until the next parameter update *)

Begin
   $T = T_o$ ;
   $CurS = S_o$ ;
   $BestS = CurS$ ; /*  $BestS$  is the best solution seen so far */
   $CurCost = Cost(CurS)$ ;
   $BestCost = Cost(BestS)$ ;
   $Time = 0$ ;
  Repeat
    Call Metropolis ( $CurS, CurCost, BestS, BestCost, T, M$ )
     $Time = Time + M$ ;
     $T = \alpha T$ ;
     $M = \beta M$ ;
  Until ( $Time \geq MaxTime$ );
  Return ( $BestS$ )
End. (*of simulated_annealing*)

Algorithm Metropolis( $CurS, CurCost, BestS, BestCost, T, M$ )
Begin
  Repeat
     $NewS = Neighbor(CurS)$ ; /*Return a neighbor from aleph( $CurS$ )*/
     $NewCost = Cost(NewS)$ ;
     $\Delta Cost = (NewCost - CurCost)$ ;
    If ( $\Delta Cost < 0$ ) Then
       $CurS = NewS$ ;
      If ( $NewCost < BestCost$ ) Then
         $BestS = NewS$ ;
      endif
    Else
      If ( $RANDOM < e^{-\Delta Cost/T}$ ) Then
         $CurS = NewS$ ;
      endif
    endif
     $M = M - 1$ 
  Untill ( $M = 0$ )
End

```

Figure 3.5: Simulated annealing algorithm (Sait et al. [1]).

SA starts with initialization of some parameters which include cooling rate α , and number of iterations, the algorithm is supposed to run. The initial temperature is also calculated at the start of algorithm.

The metropolis function is an important issue in this algorithm. At a particular temperature the predefined number of moves are made. The function accepts the good moves readily, and bad moves are accepted with some probability.

3.6.1 Initial Solution

Initial solution is required as a starting (Seed) solution in simulated annealing algorithm. The initial solution is generated randomly. Unlike other iterative heuristics, Simulated annealing (SA) works with one solution rather than a population.

3.6.2 Neighborhood Generation

A transition mechanism generates a new solution (Schedule) S' from a current solution (Schedule) S . The transition mechanism is analogous to transition operator used in other iterative heuristics. Neighbor solution is generated with the method discussed in Section 3.4.2.

3.6.3 Cooling Schedule

Simulated annealing (SA) converges to a global optimum if an infinite number of transitions are allowed at each temperature T_i and the temperature approaches

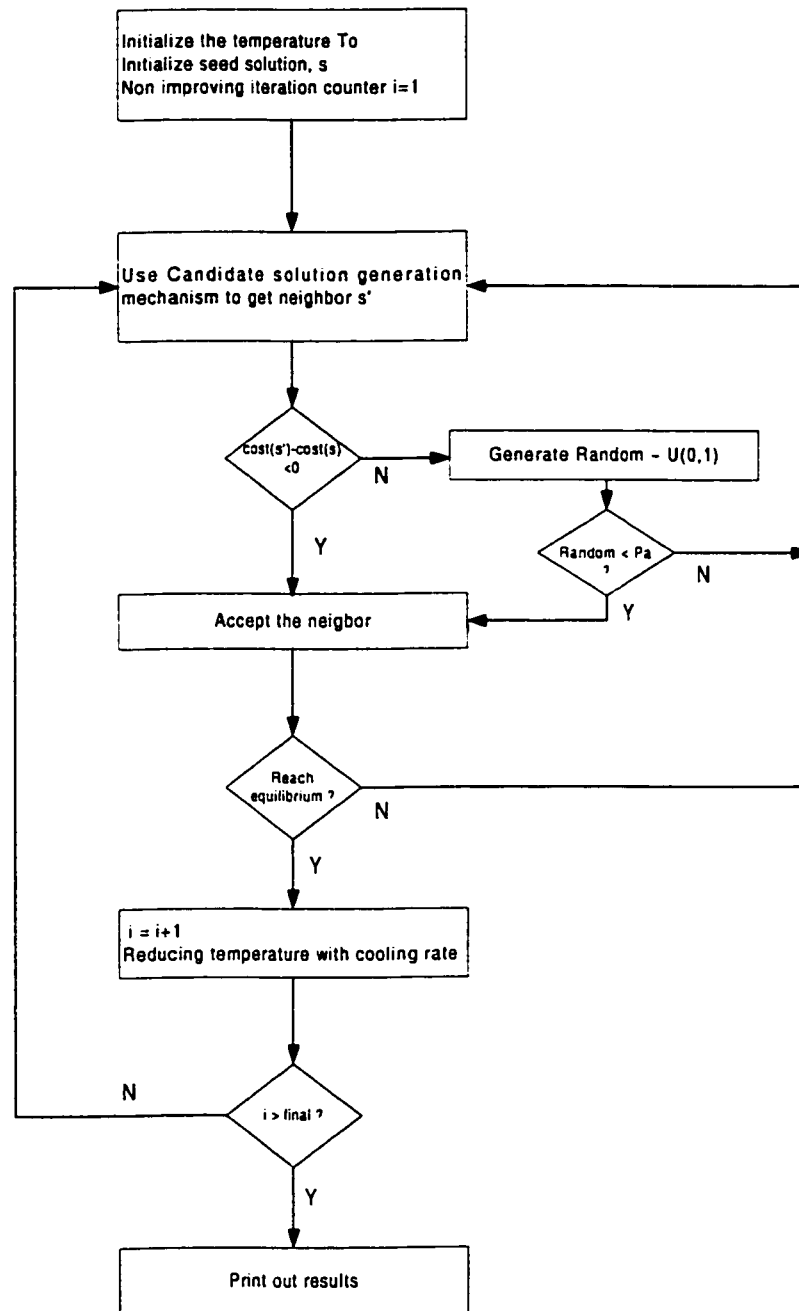


Figure 3.6: Simulated annealing algorithm flow chart.

to zero as the time approaches to infinity. However, the number of iterations at each temperature T_i and the number of temperature steps are usually finite in real implementation. Therefore SA can not find the optimal solution all the time. The three important issues which need to be considered for cooling policy are:

1. The initial temperature value.
2. The final temperature value or the stopping criteria (condition).
3. The decrement rule of the temperature of the number of temperature steps.

Initial temperature should be set such that almost all possible transitions at the initial temperature T_o are accepted, i.e. $e^{-\frac{\Delta C}{T_o}} \approx 1$. This is approximated by defining the *acceptance ratio* χ , which is obtained by dividing the number of accepted trial solutions by total number of trial solutions. We use the idea proposed by Junaid et al. [38] to determine the initial temperature. N neighbor solutions are generated on a trial solution (sequence). In our implementation we use $N = 2n$. The standard deviation of the cost for N trial moves is calculated. Initial temperature is taken as twice the standard deviation of N neighbor solutions obtained from initial solution.

$$T_o = 2\sigma N \quad (3.6)$$

Annealing starts at T_o again, N Neighbor solutions are generated and number of accepted moves k are counted. If k is greater than or equal to 90 percent of the

total number of neighbor solutions at T_o , then we keep this temperature and continue the process till stopping criteria is reached. Otherwise a new initial temperature is generated by following formula

$$T_o = (2 - \frac{k}{N})T_o \quad (3.7)$$

Again k is calculated. This process continues till the k satisfy the 90 percent acceptance criteria. This is a robust process of temperature generation and the initial temperature is updated without delaying annealing process automatically. In Annealing process, if temperature drops below 1×10^{-6} , then it is kept constant at this level for the remaining iterations.

$$T_{i+1} = \alpha T_i \quad (3.8)$$

where $i \geq 0$, $0 < \alpha < 1$, T_i is the current temperature and T_{i+1} is the new temperature to be set from the current temperature by decrement rule.

3.6.4 Markov Chain Length

The purpose of calling Metropolis loop (Figure 3.5) several times (i.e., M times) is to attain quasi equilibrium state at a given temperature during annealing process. This Metropolis loop size is also called the Markov chain length. In this thesis the

Markov chain length M is made fixed and M is determined by experimentation. The Metropolis loop is called M times, and process is repeated till there is no improvement observed in best visited solution so far during last M transitions at a given level of temperature.

3.6.5 Stopping Criterion

The stopping criterion used in our experimentation was fixed number of non improving iterations. In this thesis all implementations of Simulated annealing (SA) have the stopping criteria of 10000 non improving iterations.

3.6.6 Parameter Selection for SA

To test the effect of the parameter selection on the performance of Simulated annealing (SA) factorial design of experiments (DOE) are conducted. Three factors are selected for the test.

1. Number job to be Scheduled (n). Three levels of jobs are selected at 10, 50 and 100.
2. Number of initial trial solutions generated. Two levels at n and $3n$ are selected.
3. Cooling rate (α). Three levels selected are .90, .95 and .99.

Relative Improvement:

The problem size (number of jobs) was found to be significantly affecting the perfor-

mance of the algorithm. The relative improvement in solution quality decreases as the number of job increase. All main effects and interactions were found insignificant (Appendix C).

CPU Time:

A similar experiment was performed with CPU time. The number of jobs (n) was found significant. The interactions between number of jobs and cooling rate, and that between initial temperature criteria and cooling rate α were also found significant (Appendix D).

3.7 Proposed Lower Bound

A lower bound (LB 1) scheme is proposed by relaxing the maintenance requirement for the job sequence that would be optimal for a given objective (e.g., total completion time) without any maintenance requirement (discussed in Section 1.1). For example the Shortest Processing Time (SPT) sequence is optimal for total completion time minimization if no maintenance is performed. Now the maintenance requirement is relaxed in each batch such that the summation of processing times of jobs in each batch either equates or exceeds continuous working time limit T by a maximum of the processing time of last job in each batch only. The performance of LB 1 is dependent on the continuous working time limit (T), and the maintenance time (t). For a given continuous working time limit (T), LB 1 is tight for lower

values of t and loose for higher values of t . Similarly for given maintenance time (t) the LB 1 is tight for higher ranges of (T) and loose for lower ranges of (T). Details of Lower bound (LB 1) are given in Figure 3.7.

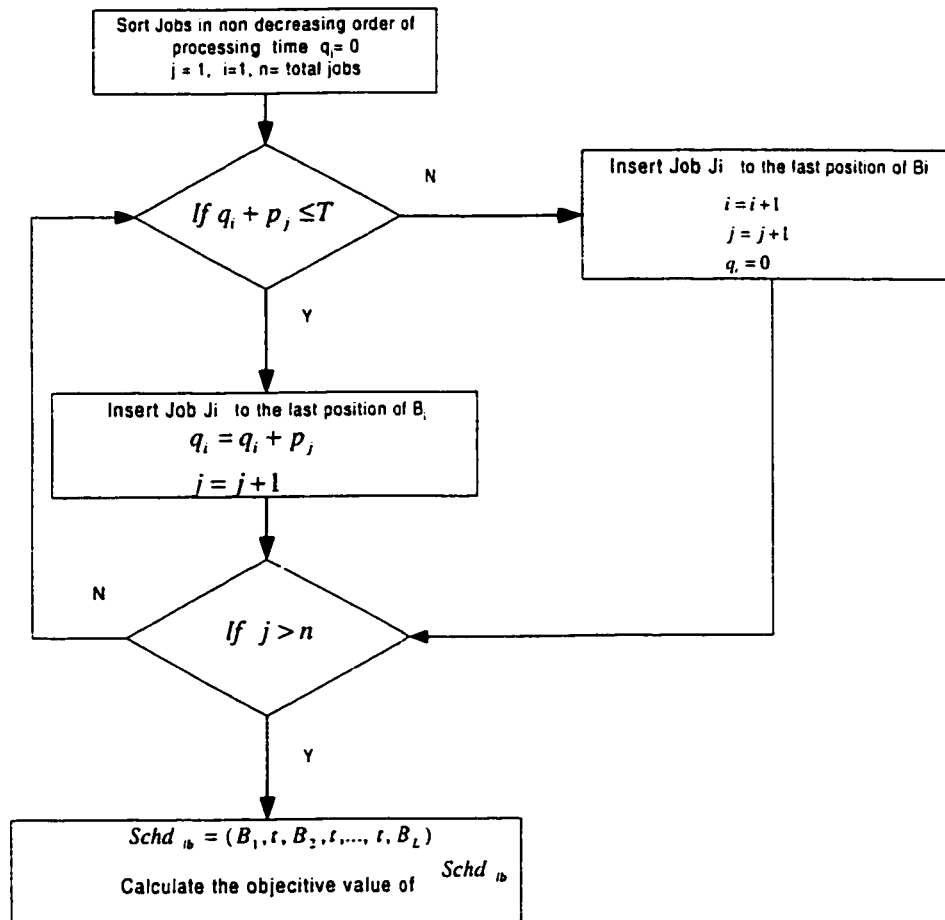


Figure 3.7: Proposed Lower Bound (LB 1) scheme for total completion time minimization.

3.8 Results

In all experimentation of this thesis test problems are randomly generated. In all the test problems the processing times are generated with uniform distribution between 1 and 30. All algorithms were coded in FORTRAN 90 and all problems are tested on DELL Pentium-III 950 MHZ processor.

3.8.1 Comparison of TS and SA With BNB

Implementation issues of Simulated annealing (SA) and Tabu search (TS) have been discussed in this chapter earlier. Experiments are performed to determine the suitable parameters for TS and SA Algorithms. For Tabu search algorithm the selected parameters are; Candidate list size (CLS) = 20 and Tabu list size (TLS)= 11. For Simulated annealing Cooling rate (α) = .95, acceptance at initial temperature is at least 90 % and Markov chain length (M) is 20. Both algorithms stop at 10,000 iterations if there is no improvement observed in the best visited solution.

Table 3.1 shows the total completion time and the CPU time required for each method for $t = 10, 20,$ and 40 for $n = 9$ and $T = 40$. The results of the two problems of each combination is given. Similarly results are obtained for other values of T and n are presented in Appendix F. Tables F.1- F.18 show the result for $n= 9, 10,11,$ $T= 40, 50, 60, 70,80$ and $t= 10, 20, 30, 40$.

	TS		SA		BNB	
t	ΣC	CPU Time sec	ΣC	CPU Time sec	ΣC	CPU Time sec
10	389.2	1.2	389.2	1.0	389.2	4
10	855.8	1.7	855.8	1.2	855.8	86
20	931.6	1.5	931.6	1.4	931.6	30
20	936.8	1.5	936.8	1.1	936.8	45
30	1220.3	1.4	1322.2	0.3	1220.3	70
30	806.3	2.0	806.3	0.7	806.3	27
40	1322.2	1.4	1322.2	1.4	1322.2	670
40	1241.2	1.7	1244.4	1.4	1241.2	60

Table 3.1: Comparison of TS, SA & BNB for $n = 9$, $T = 40$.

3.8.2 Comparison of TS, SA With SPT

Experiments are performed to compare the Tabu search (TS) and Simulated annealing (SA) with SPT heuristic for large size problems. Stopping criterion is to stop both TS and SA at 10,000 non improving iterations. For Tabu search (TS) Candidate list size (CLS) is 150 and Tabu list size (TLS) is 13 for more than 20 jobs otherwise it is 7. For Simulated annealing (SA), Metropolis loop size is same as Candidate list size (CLS), Cooling rate α is 0.99 and acceptance rate is at least 90 % at initial temperature.

Tables 3.2 - 3.5 show the average of 10 runs for each combination of T and t ranging from $T = 50$ to 80 and $t = 10$ to 40. The tables show the average of total completion

time ($\sum C$) for 10 randomly generated problems for each of the Shortest processing time (SPT) heuristic, Tabu search (TS) and Simulated annealing (SA). The average CPU time for the 10 runs is also given for TS and SA.

SPT			TS		SA	
T	t	$\sum C$	$\sum C$	CPU Time sec	$\sum C$	CPU Time sec
50	10	2643	2625	46	2612	62
50	20	2843	2831	40	2811	65
50	30	3595	3562	41	3508	66
50	40	3802	3726	40	3726	66
60	10	2416	2400	37	2400	43
60	20	3172	3146	38	3146	47
60	30	3735	3662	43	3662	45
60	40	3382	3348	37	3348	44
70	10	2130	2118	35	2118	54
70	20	2671	2671	35	2671	55
70	30	3650	3622	37	3622	52
70	40	3254	3225	53	3203	57
80	10	2638	2637	38	2637	50
80	20	3294	3288	40	3288	46
80	30	2870	2852	37	2855	48
80	40	3426	3402	35	3350	59

Table 3.2: Comparison of SPT heuristic with TS and SA for $n = 20$.

SPT			TS		SA	
T	t	ΣC	ΣC	CPU Time sec	ΣC	CPU Time sec
50	10	7472	7456	82	7456	95
50	20	10161	10138	79	10138	131
50	30	10610	10320	100	10320	133
50	40	13455	13305	92	13374	148
60	10	7223	7197	68	7177	139
60	20	8698	8667	69	8660	108
60	30	11091	11043	76	11031	125
60	40	11413	11294	76	11275	134
70	10	6676	6670	69	6640	110
70	20	8161	8082	66	8082	118
70	30	11032	10873	70	10856	128
70	40	10912	10741	73	10708	115
80	10	7904	7881	74	7878	116
80	20	10123	10019	65	10019	117
80	30	9453	9384	66	9384	113
80	40	10374	10238	64	10195	110

Table 3.3: Comparison of SPT heuristic with TS and SA for $n = 35$.

SPT			TS		SA	
T	t	ΣC	ΣC	CPU Time sec	ΣC	CPU Time sec
50	10	10332	10264	98	10264	114
50	20	13696	13582	99	13567	122
50	30	14210	14197	97	14197	145
50	40	17515	16896	125	16896	136
60	10	9598	9583	82	9598	126
60	20	11976	11882	87	11963	121
60	30	14288	14092	88	14116	158
60	40	14746	14700	83	14700	135
70	10	8829	8781	89	8781	137
70	20	10415	10356	83	10353	141
70	30	13637	13624	97	13624	153
70	40	14498	14432	85	14432	149
80	10	10256	10200	79	10200	132
80	20	13005	12977	89	12916	137
80	30	11975	11907	77	11820	135
80	40	13742	13663	74	13663	128

Table 3.4: Comparison of SPT heuristic with TS and SA for $n = 40$.

SPT			TS		SA	
T	t	ΣC	ΣC	CPU Time sec	ΣC	CPU Time sec
50	10	16333	16274	160	16274	193
50	20	21142	20920	185	21077	210
50	30	23406	22892	192	22813	200
50	40	28922	28488	187	28632	267
60	10	16365	16347	112	16347	183
60	20	17187	17111	141	17111	145
60	30	22352	21700	140	21700	128
60	40	23338	23338	123	23338	259
70	10	14617	14617	122	14617	176
70	20	16518	16513	143	16518	241
70	30	20584	20474	117	20584	208
70	40	22899	22559	196	22830	171
80	10	15172	15162	102	15162	182
80	20	19386	19300	119	19300	218
80	30	18187	18171	139	18171	176
80	40	20380	20328	99	20216	209

Table 3.5: Comparison of SPT heuristic with TS and SA for $n = 50$.

3.8.3 Deviation of TS, SA and SPT From Lower Bound

Finally we are computing percentage relative deviation of Tabu search (TS). Simulated annealing (SA) and SPT heuristic from Proposed lower bound (LB 1) for large size problems. This study will give an idea about how far the solution methods are from their super optimal values (Lower Bounds). In Tables 3.6 - 3.7 average percentage of relative deviation is presented for each combination of T and t . 10 problems are solved and the average percentage relative deviation is reported for each method.

n	T	t	SPT	TS	SA
20	50	10	4.10	3.44	3.01
20	50	20	5.59	5.04	4.45
20	50	30	5.35	7.03	5.35
20	50	40	8.12	5.96	5.96
20	60	10	1.10	1.10	1.10
20	60	20	5.09	4.01	4.01
20	60	30	6.87	4.74	6.87
20	60	40	7.39	6.21	7.39
20	70	10	2.20	1.69	1.69
20	70	20	2.11	2.11	2.08
20	70	30	4.57	4.57	3.62
20	70	40	6.77	5.91	5.09
20	80	10	1.42	1.42	1.38
20	80	20	2.57	2.36	2.36
20	80	30	3.97	3.32	3.41
20	80	40	4.88	4.19	2.57

Table 3.6: Percentage relative deviation of SPT, TS and SA from lower bound.

n	T	t	SPT	TS	SA
35	50	10	4.71	4.51	4.71
35	50	20	9.26	8.15	9.04
35	50	30	12.22	8.18	9.71
35	50	40	13.07	11.99	12.50
35	60	10	2.98	2.65	2.34
35	60	20	6.14	5.68	5.48
35	60	30	8.02	7.51	7.39
35	60	40	9.78	8.67	8.48
35	70	10	2.30	2.21	1.74
35	70	20	3.85	3.85	2.85
35	70	30	6.72	5.28	5.09
35	70	40	7.46	5.78	5.49
35	80	10	1.70	1.41	1.38
35	80	20	3.66	3.66	2.62
35	80	30	3.56	2.92	2.92
35	80	40	5.46	4.18	3.66
40	50	10	5.39	4.74	4.90
40	50	20	8.05	7.35	7.20
40	50	30	10.17	10.17	10.17
40	50	40	15.68	11.66	12.79
40	60	10	2.89	2.72	2.89
40	60	20	6.39	5.67	6.21
40	60	30	8.07	6.52	6.69
40	60	40	8.26	8.26	8.26
40	70	10	2.21	1.94	1.65
40	70	20	3.65	3.05	3.06
40	70	30	5.50	5.40	5.44
40	70	40	7.34	6.76	6.76
40	80	10	1.86	1.86	1.86
40	80	20	4.03	3.78	3.23
40	80	30	5.31	4.67	3.84
40	80	40	5.23	5.23	4.59

Table 3.7: Percentage relative deviation of SPT, TS and SA from lower bound.

3.9 Discussion

A comparative study is made between Branch and Bound (BNB) algorithm, Tabu search (TS) and Simulated annealing (SA) for small size problems. For large size problems, SPT heuristic is compared with TS and SA. The following conclusions could be drawn:

n	TS		SA		BNB		% Relative error	
	$\sum C$	CPU Time sec	$\sum C$	CPU Time sec	$\sum C$	CPU Time sec	TS	SA
9	847.5	1.0	850.03	1.0	845.7	57.25	0.22	0.51
10	1936.8	1.0	1936.35	1.1	1934.5	1664.92	0.12	0.10
11	1187.0	1.6	1187.92	1.7	1186.3	9168.44	0.05	0.14

Table 3.8: Overall comparison of TS, SA and BNB with different number of jobs.

From Table 3.8, we note that TS and SA produced results that are within 1 % of the optimal solution. Table 3.8 shows the overall comparison of average $\sum C$ of BNB, TS and SA for different number of jobs on over 100 tested problems. $\sum C$ represents the overall average of total completion time for the tested problems. CPU time comparison of BNB with TS and SA (Figure 3.8) shows that the CPU time for BNB increases exponentially with job size, but the increase in CPU time for TS and SA is insignificant in comparison to BNB.

In Tables 3.2 - 3.5 and Tables 3.6 - 3.7 the comparison of SPT heuristic is made

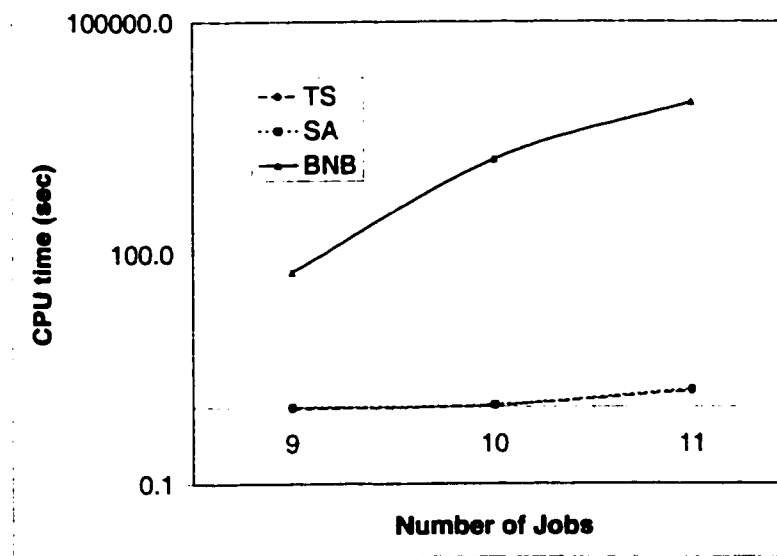


Figure 3.8: CPU time Comparison of TS and SA with BNB.

with TS and SA. The following conclusions can be drawn:

The percent relative deviation of SPT heuristic from lower bound (LB 1) increases with the increase in number of jobs. This is because as the problem size increases the number of possible solutions increase exponentially. Figure 3.9 shows the logarithmic plot of CPU time variation with problem size. For a given value of maximum continuous working time (T), the relative error of SPT heuristic increases with the maintenance time t . Figures 3.10 shows the effect of maintenance time t on performance of SPT, TS and SA heuristics based on % relative deviation form lower bound (LB 1).

Similarly Figure 3.11 shows the effect of maintenance time t on the error of SPT heuristic when it is compared with TS and SA. This is because as we know that

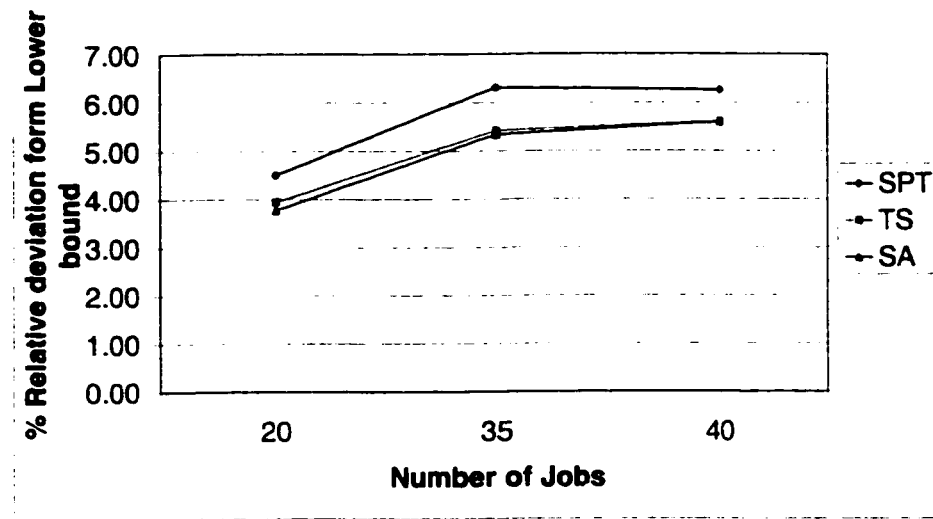


Figure 3.9: Effect of Number of jobs n on performance.

in SPT heuristic total cost of maintenance times may not be optimal, therefore as the maintenance time (t) increases the contribution of total cost of maintenance increases. Maintenance contribution to the total cost of resulting schedule becomes significant, under this condition the scheduling of maintenance becomes more important than sequencing of jobs. This is why the error of SPT heuristic increases.

Similarly for a given value of maintenance time t , the error of SPT heuristic increases with a decrease in maximum continuous working time of machine T . This is because of the fact that as the maximum continuous working time T decreases, the maintenance constraint gets tighter i.e. more requirement of maintenance. As it is discussed earlier that the total cost of schedule is total cost of processing time

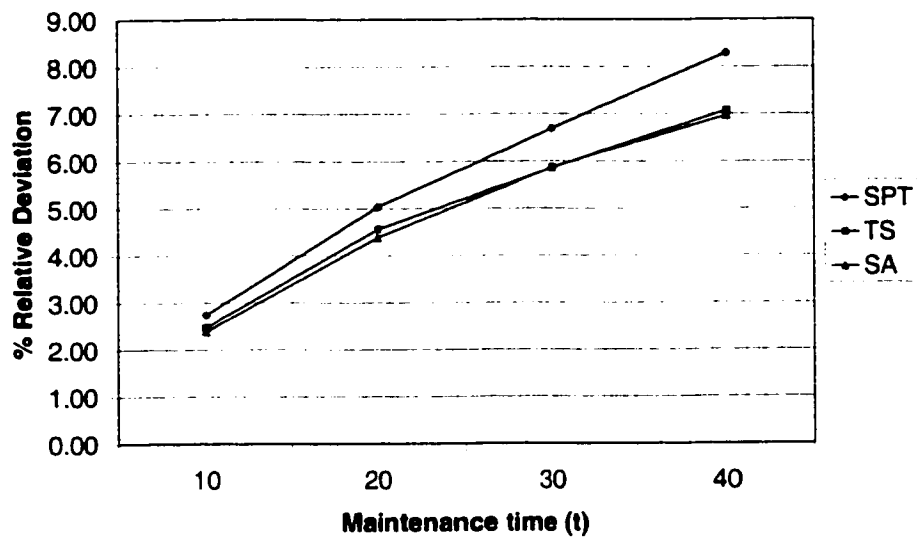


Figure 3.10: Effect of maintenance time t on performance.

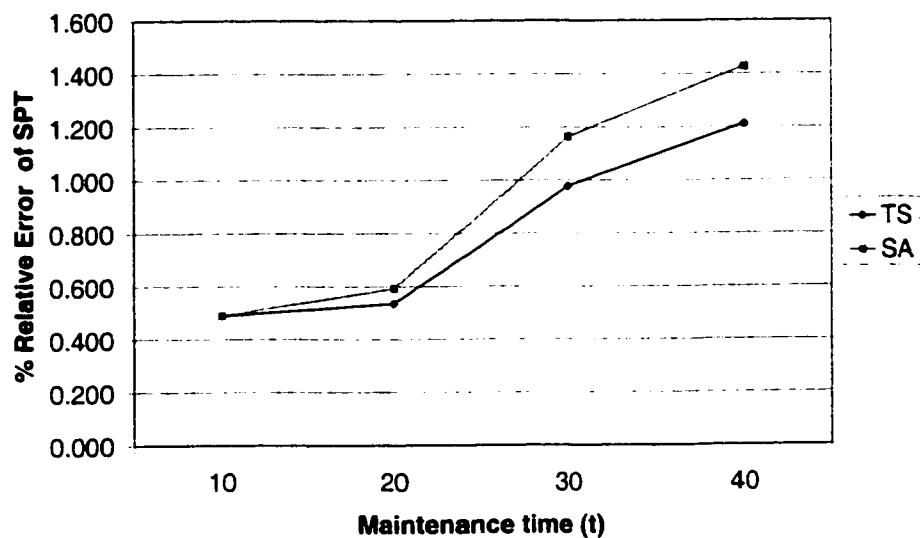


Figure 3.11: Effect of maintenance time t on SPT performance.

of jobs and the total cost of maintenance. SPT heuristic is optimal with respect to contribution of total processing time of jobs, but it may not be optimal for total contributions from maintenance times. With reduction of T the maintenance requirement increases and total contribution of maintenance becomes significant and therefore SPT heuristic produces higher relative errors. Figure 3.12 shows effect of maximum continuous working time T on the performance of the SPT, TS and SA when the comparison is made based on proposed lower bound (LB 1).

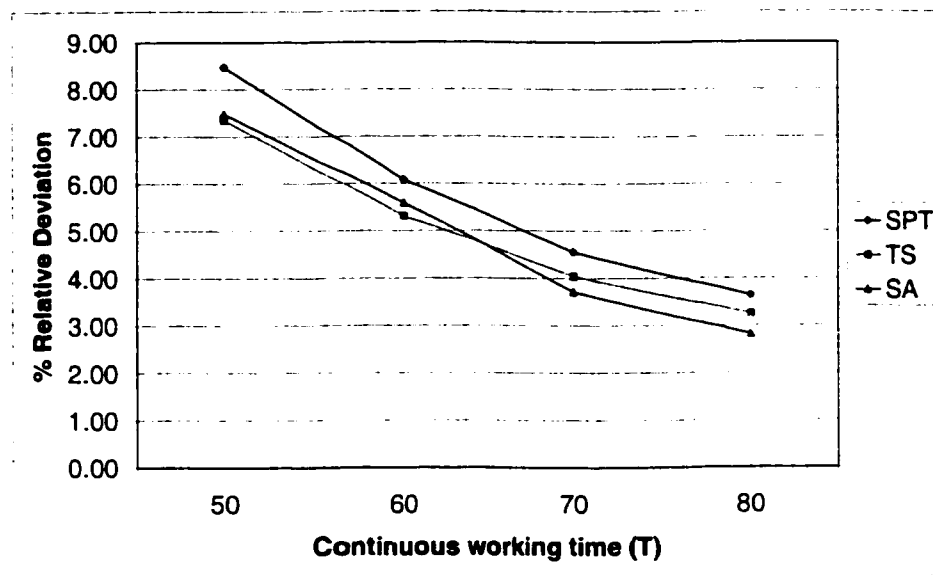


Figure 3.12: Effect of Continuous working time on performance.

Similarly Figure 3.13 shows the effect of T on the error of SPT heuristic when it is compared with TS and SA.

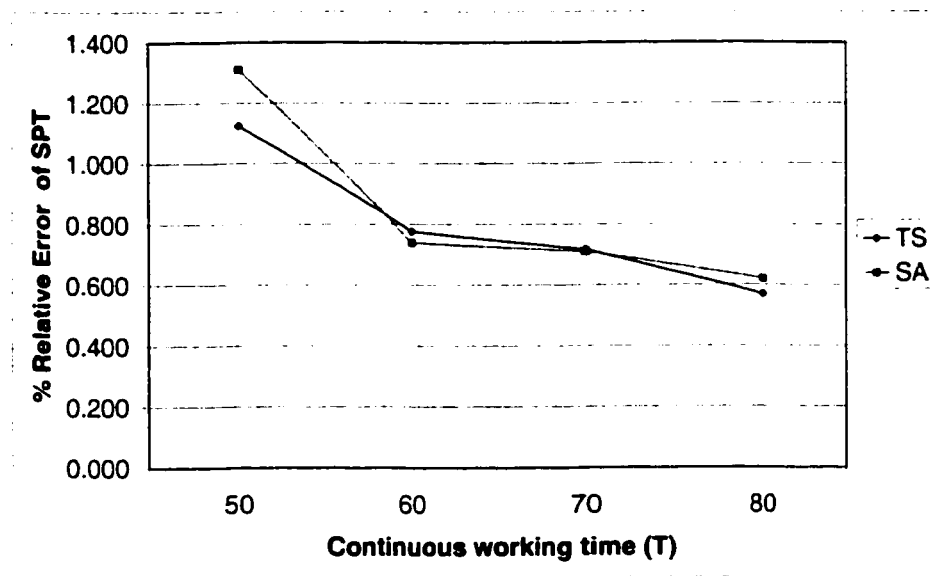


Figure 3.13: Effect of Continuous working time on SPT performance.

Chapter 4

Scheduling With Delayed Maintenance Operations

In this chapter, we will study the problem of scheduling jobs on a single machine that is subject to preventive maintenance. The study is an extension to the problem studied by X. Qi et al. [4]. The objective is to minimize the total completion time of jobs.

Maximum continuous working time of machine is divided into two categories. The first category is preferred time (T_p) and the other is maximum allowed continuous working time (T_{max}). If the machine is maintained during $(0, T_p]$, the preventive maintenance time is t_1 . If it is delayed till T_{max} , that is during $(T_p, T_{max}]$, then preventive maintenance time will be t_2 , such that $t_2 \geq t_1$ and $T_{max} \geq T_p$. When

$T_p = T_{max}$ or $t_1 = t_2$, the problem reduces to the problem studied by X. Qi et al. [4]. We will develop some properties of the optimal schedule and propose a heuristic and a lower bound based on these properties. Tabu search and Simulated annealing approaches will be used to find the solution to the problem. Finally comparative study will be made.

4.1 Assumptions

The assumptions used are similar to those defined in Chapter 1, some other specific assumptions are:

1. The maximum delay allowed, T_{max} , is less than twice the preferred time T_p .
This assumption is made to simplify the problem, otherwise there would be another option of maintaining the machine preventively with maintenance time t_1 during period $(T_p, 2T_p]$, and that would make problem very complex. This assumption seems to be practical also, because usually preventive maintenance is done well before the time period where the hazard of machine breakdown is high to avoid the failure.
2. Machine will be maintained preventively if it exceeds continuous working time limit of T_p .

4.2 Notations

T_p	=	Preferred time to perform maintenance (Continuous working time of machine).
T_{max}	=	Max allowed time to perform maintenance.
t_1	=	Maintenance time if performed in $[0, T_p]$.
t_2	=	Maintenance time if performed in $(T_p, T_{max}]$.
B_i	=	Set of jobs in batch i .
q_i	=	Summation of processing times of jobs in batch B_i .
n_i	=	Number of job in batch B_i .
T_a	=	Max allowance after T_p , $T_a = T_{max} - T_p$.
L	=	Total number of batches.

4.3 Properties of Optimal Schedule

In this section some properties of optimal schedule are developed. In developing the properties we will focus on a batch in the optimal schedule. A batch is defined to be a set of jobs between two consecutive maintenance operations. The summation of the processing times of jobs in batch can not exceed T_{max} . We will develop some properties of the optimal schedule and proposed a heuristic (HSTC), based on the properties studied. The proposed heuristic (HSTC) will be compared with the iterative heuristics, Tabu search (TS) and Simulated annealing (SA).

Property 4.1 *In each batch of an optimal schedule, jobs are sequenced in non decreasing order of their processing times.*

Proof

This property can be proved by contradiction using the interchange argument. Con-

consider an optimal schedule S in Figure 4.1, which does not satisfy this property. Consider two jobs K and I in batch B_i . The processing times of jobs K and I are p_K and p_I respectively, such that $p_I < p_K$. The contribution to total completion time by jobs k and I is $2a + 2p_K + p_I$. Now consider another schedule S' where job I precedes job K . The contribution of jobs k and I will be $2a + 2p_I + p_K$. As we know that $p_I < p_K$ and the completion time of all remaining jobs in S and S' is the same, the total completion time of schedule S' is less than the total completion time of schedule S .

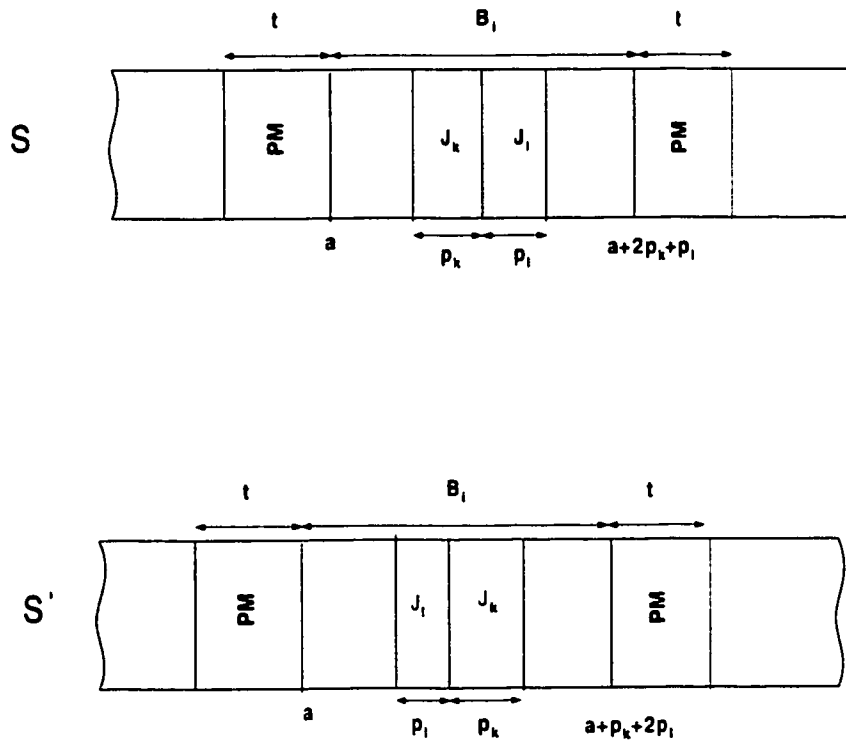


Figure 4.1: SPT property with each batch.

The following property indicates that in the optimal schedule the total processing time of jobs in any batch is less than the smallest processing time among all jobs in subsequent batches. In the following property T represents both T_{max} , and T_p respectively. Similarly t represents both t_1 and t_2 .

Property 4.2 *In an optimal schedule for any batch B_i*

$$T - q_i < p_j \quad \forall j \in B_k, \quad k = i + 1, i + 2, \dots, L$$

Proof

This property will be proven by a contradiction argument. Suppose there is an optimal schedule S where batch B_i does not satisfy this property. Let $q_i \leq T$ and job J_j with processing time p_j be the first job satisfying $p_j \leq T - q_i$, $J_j \in B_k$, and $k > i$. Let A be the set of jobs which are sequenced after batch B_i and before job J_j , and n_A be the number of jobs in set A . A new feasible schedule S' can be obtained by putting job J_j at the end of B_i without changing the completion time of other jobs except J_j and other jobs in set A . In the new schedule S' , completion time of job J_j will decrease by $(k - i)t + \sum_{k \in A} p_k$ and the completion time of jobs in set A will increase by $n_A p_j$. The completion time of all other jobs in schedule S and S' will remain the same. As we know J_j is the first job with processing time p_j satisfying $p_j \leq T - q_i$, $J_j \in B_k$, $k > i$, therefore all jobs in set A will satisfy $p_j < p_k \quad \forall k \in A$

and S will be superior to S' .

The following lemma extends the SPT (sequencing in non decreasing order of processing times) property at the level of batch. In the lemma, t represents the maintenance time for both t_1 , and t_2 .

Lemma 4.1 *The optimal schedule(s) is such that*

$$\frac{q_i}{n_i} \leq \frac{q_{i+1}}{n_{i+1}} \quad i = 1, 2, 3, \dots, L - 1. \quad (4.1)$$

where L is the total number of batches of jobs.

Proof

This lemma can be proved by interchange argument of adjacent batches. Suppose that S is a schedule in Figure 4.2 that does not satisfy this property, and there are two batches (strings) B_i and B_{i+1} such that B_{i+1} immediately precedes B_i with relationship $\frac{q_i}{n_i} < \frac{q_{i+1}}{n_{i+1}}$. Let S' be the schedule formed by interchanging the batches B_i and B_{i+1} (see Figure 4.2). The resulting change in the total completion time arises from two sources. The contribution from batch B_{i+1} increases by the amount $n_{i+1}q_i$, since each job in batch B_{i+1} has its completion time increased by amount q_i . The contribution of jobs from batch B_i decreases by amount n_iq_{i+1} , since each job in B_i has its completion time decreased by q_{i+1} . There is no change in the

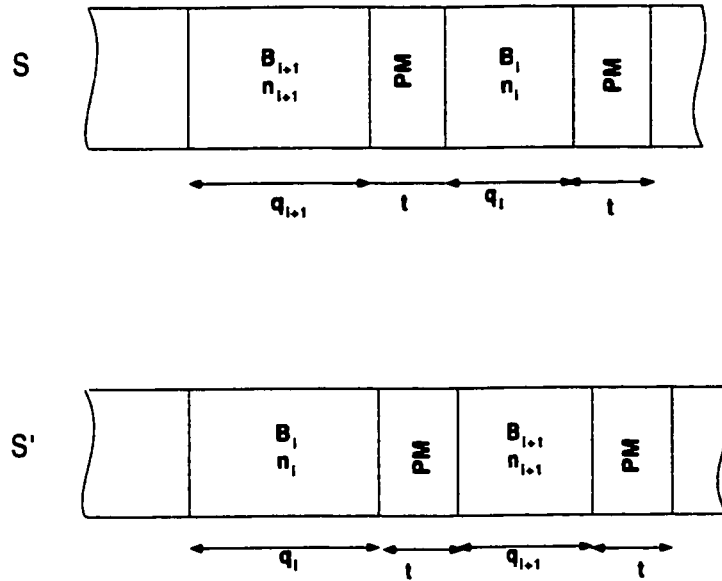


Figure 4.2: SPT property on batch.

completion time of any other job. From $\frac{q_i}{n_i} < \frac{q_{i+1}}{n_{i+1}}$, we have $n_i q_{i+1} > n_{i+1} q_i$. Here, $n_{i+1} q_i - n_i q_{i+1}$ is the net decrease in the total completion time of new schedule S' , therefore interchange of batches B_i and B_{i+1} brings net decreases in total completion time. This lemma is the extension of SPT rule to the level of batches.

Property 4.3 *In an optimal schedule,*

$$\frac{t + q_i}{n_i} \leq \frac{t + q_{i+1}}{n_{i+1}}, \quad i = 1, 2, 3, \dots, L - 1 \quad (4.2)$$

where t represents both t_1 and t_2 .

Proof

This property is the extension of SPT rule on batches with inserted maintenance time at the end of each batch. The proof is similar to Lemma 4.1.

Property 4.4 *If there are two adjacent batches B_i and B_{i+1} such that each batch follows same maintenance time, then in an optimal schedule $n_i \geq n_{i+1}$.*

Proof

Suppose S is an optimal schedule in which there exist two batches B_i and B_{i+1} such that $n_i < n_{i+1}$. Here T represents both T_p and T_{max} . According to Property 4.3, we have $\frac{t + q_i}{n_i} \leq \frac{t + q_{i+1}}{n_{i+1}}$, then

$$q_i \leq \frac{n_i q_{i+1} + (n_i - n_{i+1})t}{n_{i+1}} \quad (4.3)$$

Substituting the value of q_i from equation 4.3 in following expression, we get.

$$\begin{aligned} T - q_i &\geq T - \left(\frac{n_i(q_{i+1} + t) - n_{i+1}t}{n_{i+1}} \right) \\ &= \frac{n_{i+1}T - n_i q_{i+1} - n_i t + n_{i+1}t}{n_{i+1}} \\ &= \frac{(n_{i+1} - n_i)T}{n_{i+1}} + \frac{n_i(T - q_{i+1})}{n_{i+1}} + \frac{(n_{i+1} - n_i)t}{n_{i+1}} \end{aligned} \quad (4.4)$$

Let w be the shortest processing time of a job in batch B_{i+1} and $n_{i+1} > n_i$,

$$w = \min\{p_j \mid j \in B_{i+1}\}, w \leq \frac{T}{n_{i+1}}.$$

$$T - q_i \geq \frac{(n_{i+1} - n_i)T}{n_{i+1}} + \frac{(n_{i+1} - n_i)t}{n_{i+1}} \quad (4.5)$$

$$\begin{aligned} &\geq \frac{(n_{i+1} - n_i)T}{n_{i+1}} && \text{Since } n_{i+1} - n_i \geq 1 \\ &\geq w && \end{aligned} \quad (4.6)$$

Hence if $n_{i+1} > n_i$, then it contradicts the Property 4.2 of optimality which states that, in any arbitrary batch of optimal schedule none of the jobs in following batches could be shifted to that batch.

Property 4.5 *If there are two adjacent batches B_i and B_{i+1} such that maintenance time t_1 immediately follows B_i and maintenance time t_2 immediately follows B_{i+1} , then in optimal schedule $\frac{n_{i+1}}{n_i} < \frac{T_{max} + t_2}{T_p + t_1 - w}$.*

Proof

Consider two batches B_i and B_{i+1} as shown in Figure 4.3. Let w be the shortest

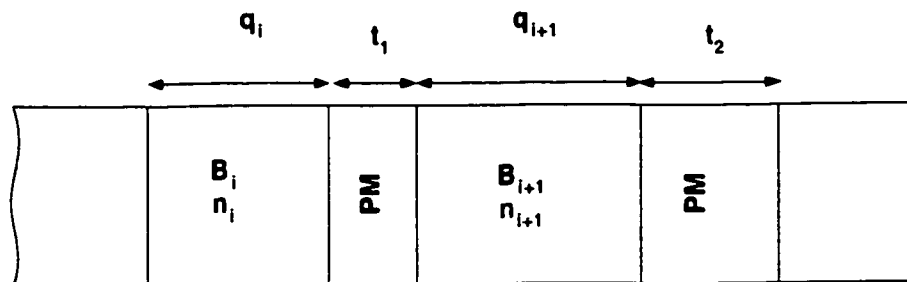


Figure 4.3: Batch B_i followed by t_1 and B_{i+1} followed by t_2 .

processing time in batch B_{i+1} , $w = \min\{p_j \mid j \in B_{i+1}\}$, let $w \leq \frac{T_{max}}{n_{i+1}}$ (Average processing time of a job in batch). Maintenance for time t_1 is performed if the continuous working time does not exceed T_p , otherwise if it exceeds T_p then time t_2 is required for maintenance. Batch B_i contains the jobs such that the summation of the processing times of jobs in B_i does not exceed T_p and summation of the processing time of jobs in batch B_{i+1} exceeds T_{max} .

Now consider the batch B_i . Again from Property 4.3 we can write

$$\frac{t_1 + q_i}{n_i} \leq \frac{t_2 + q_{i+1}}{n_{i+1}} \quad (4.7)$$

The equation 4.7 can be rewritten as

$$q_i \leq \frac{n_i(t_2 + q_{i+1})}{n_{i+1}} - t_1 \quad (4.8)$$

Substituting the value of q_i from equation 4.8 in the following expression.

$$\begin{aligned} T_p - q_i &\geq T_p - \left(\frac{n_i(q_{i+1} + t_2)}{n_{i+1}} - t_1 \right) \\ &= \frac{n_{i+1}T_p - n_iq_{i+1} - n_it_2 + n_{i+1}t_1}{n_{i+1}} \\ &= \frac{(n_{i+1} - n_i)T_{max} + n_i(T_{max} - q_{i+1}) - n_{i+1}T_a - n_it_2 + n_{i+1}t_1}{n_{i+1}} \\ &\geq \frac{(n_{i+1} - n_i)T_{max} - n_{i+1}T_a - n_it_2 + n_{i+1}t_1}{n_{i+1}} \\ &= \frac{(n_{i+1} - n_i)T_{max} - n_{i+1}(T_{max} - T_p) - n_it_2 + n_{i+1}t_1}{n_{i+1}} \end{aligned} \quad (4.9)$$

From Property 4.2, we know that in any batch of an optimal schedule none of the jobs from following batches could enter into the batch without violating feasibility of schedule. Therefore if $T_p - q_i$ is more than w , then we could transfer job from batch B_{i+1} to batch B_i without increasing the cost of schedule. For a schedule to be optimal, Property 4.2 must be satisfied. The expression $T - q_i$ must be less than shortest processing time of job in batch B_{i+1} , hence we can write.

$$\frac{n_{i+1}T_p + n_{i+1}t_1 - n_iT_{max} - n_it_2}{n_{i+1}} < w$$

$$\frac{n_{i+1}}{n_i} < \frac{T_{max} + t_2}{T_p + t_1 - w} \quad (4.10)$$

Property 4.6 *If there are two adjacent batches B_i and B_{i+1} such that t_2 immediately follows B_i and t_1 immediately follows B_{i+1} , then in optimal schedule $\frac{n_{i+1}}{n_i} < \frac{T_p + t_1}{T_{max} + t_2 - w}$.*

Proof

Let w be the shortest processing time in batch B_{i+1} , $w = \min\{p_j \mid j \in B_{i+1}\}$, let $w \leq \frac{T_p}{n_{i+1}}$, t_1 and t_2 are as defined in Section 4.2. Considering Figure 4.4 maintenance for time t_2 immediately follows batch B_i and t_1 immediately follows batch B_{i+1} . The summation of processing times of jobs in batch B_i exceeds T_p but the summation of

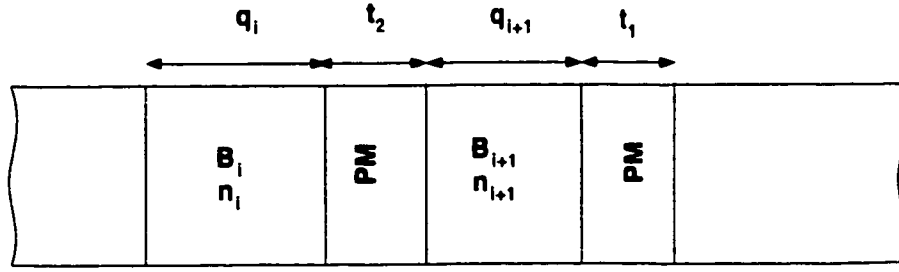


Figure 4.4: Batch B_i followed by t_2 and B_{i+1} followed by t_1 .

processing times of jobs does not exceed T_p . Again from Property 4.3 we know that

$$\frac{t_2 + q_i}{n_i} \leq \frac{t_1 + q_{i+1}}{n_{i+1}} \quad (4.11)$$

The equation 4.11 can be rewritten as following

$$q_i \leq \frac{n_i(t_1 + q_{i+1})}{n_{i+1}} - t_2 \quad (4.12)$$

Substituting the value of q_i from equation 4.12 in the following expression.

$$\begin{aligned} T_{max} - q_i &\geq T_{max} - \left(\frac{n_i(q_{i+1} + t_1) - n_{i+1}t_2}{n_{i+1}} \right) \\ &= \frac{(n_{i+1}T_{max} - n_iq_{i+1} - n_it_1 + n_{i+1}t_2)}{n_{i+1}} \\ &= \frac{(n_{i+1} - n_i)T_p + n_i(T_p - q_{i+1}) + n_{i+1}T_a - n_it_1 + n_{i+1}t_2}{n_{i+1}} \\ &\geq \frac{(n_{i+1} - n_i)T_p + n_{i+1}T_a - n_it_1 + n_{i+1}t_2}{n_{i+1}} \end{aligned} \quad (4.13)$$

Here we use the similar argument presented in Property 4.5, and according to that following condition must be satisfied

$$\begin{aligned}
 w &> \frac{(n_{i+1} - n_i)T_p + n_{i+1}T_a - n_i t_1 + n_{i+1}t_2}{n_{i+1}} \\
 n_i &> \frac{n_{i+1}(T_{max} + t_2 - w)}{T_p + t_1} \quad (\text{Since } T_a = T_{max} - T_p) \\
 \frac{n_{i+1}}{n_i} &< \frac{T_p + t_1}{T_{max} + t_2 - w}
 \end{aligned} \tag{4.14}$$

4.4 Proposed Heuristic

We propose a constructive heuristic (HSTC) based on the properties studied. The idea in HSTC is to schedule the jobs in non-decreasing order of their processing times and insert maintenance either at limit T_p or T_{max} whichever gives the least cost. This decision is made at T_p , either we perform preventive maintenance with time t_1 or we delay it till T_{max} with maintenance time t_2 . Detailed algorithm of HSTC is discussed in Figures. 4.5, 4.6, 4.7 and 4.8 respectively.

Algorithm HSTC

```

(* S = Schedule *)
(* U = Universal set of jobs. *)
(* SU = Set of unassign jobs. *)
(* SA = Set of assign jobs. *)
(* Bi = Set of jobs assigned to batch. i *)
(* pj = Processing time of job j. *)
(* Bi = Set of jobs assigned to batch i. *)
(* M1 = Maintenance in sequence for time duration t1. *)
(* M2 = Maintenance in sequence for time duration t2. *)
(* Cj = Completion time of job j. *)

i ← 1, j ← 1, Bi ← {0}, qi ← 0.0, SA ← {0};
Repeat
  If (qi + pj ≤ TP) Then
    Insert job Jj to sets SA, Bi, and S.
    Remove job Jj from and SU
    Cj ← Cj-1 + pj
    j ← j + 1
  Elseif (j ≤ n) Then
    S1 ← {0}, S2 ← {0}, R1 ← {0}, R2 ← {0}, D ← {0}.
    S1 ← SA, R1 ← SU, S2 ← S1, k ← j, D ← SU, d ← 0.0, h ← 1
    Repeat
      Insert job Jh h ∈ D to S2 set
      d ← d + ph, ∀ h ∈ D
      k ← k + 1, h ← h + 1
    Until (qi + ph + d ≤ Tmax & k ≤ |U| )
    R2 ← U - S2
    Call Option1 (S1, R1, Tp, Tmax, t1, t2, C, COST1)
    Call Option2 (S2, R2, Tp, Tmax, t1, t2, C, COST2)
    If (COST1 < COST2) Then
      Insert M1 in S
      i ← i + 1
      Insert job Jj in S
      j ← j + 1
    Elseif (COST2 ≤ COST1) Then
      Call Update_Batch (S1, R1, S2, R2, i, j, C, SA, SU, B, S)
    Endif
  Elseif (j = |U|) Then
    Insert M1 to the last position in S
    i ← i + 1
    Insert job j to the last position in Bi
    j ← j + 1
  Endif
Until(j ≤ |U|)
End Algorithm

```

Figure 4.5: Proposed Heuristic HSTC.

Algorithm Option1 ($S1, R1, T_p, T_{max}, t_1, t_2, C, COST1$)

(* a = Min number of batches if maintenance is performed in interval of T_p *)

(* b = Min number of batches if maintenance is performed in interval of T_{max} *)

(* na_i = Max number of jobs in within window T_p , with preemption allowed for last job in batch *)

(* nb_i = Max number of jobs in within window T_{max} , with preemption allowed for last job in batch *)

Begin

$$a = \left\lceil \frac{\sum_{\substack{i=1 \\ i \in R1}}^{|R1|} p_i}{T_p} \right\rceil$$

$$b = \left\lceil \frac{\sum_{\substack{i=1 \\ i \in R1}}^{|R1|} p_i}{T_{max}} \right\rceil$$

$$COSTa = \sum_{\substack{i=1 \\ i \in S1}}^{|S1|} C_i + |R1|C_{|S1|} + |R1|t_1 + \sum_{\substack{i=1 \\ i \in R1}}^{|R1|} (|R1| - i + 1)p_i + \sum_{i=2}^a na_i(i-1)t_1$$

$$COSTb = \sum_{\substack{i=1 \\ i \in S1}}^{|S1|} C_i + |R1|C_{|S1|} + |R1|t_1 + \sum_{\substack{i=1 \\ i \in R1}}^{|R1|} (|R1| - i + 1)p_i + \sum_{i=2}^b nb_i(i-1)t_2$$

$$COST1 = \max(COSTa, COSTb)$$

End

Figure 4.6: Cost of Option 1 (Contd. HSTC).

Algorithm Option2 ($S2, R2, T_p, T_{max}, t_1, t_2, C, COST2$)

(* a = Min number of batches if maintenance is performed in interval of T_p *)

(* b = Min number of batches if maintenance is performed in interval of T_{max} *)

(* na_i = Max number of jobs in within window T_p , with preemption allowed for last job in batch *)

(* nb_i = Max number of jobs in within window T_{max} , with preemption allowed for last job in batch *)

Begin

$$a = \left\lceil \frac{\sum_{\substack{i=1 \\ i \in R2}}^{|R2|} p_i}{T_p} \right\rceil$$

$$b = \left\lceil \frac{\sum_{\substack{i=1 \\ i \in R2}}^{|R2|} p_i}{T_{max}} \right\rceil$$

$$COSTa = \sum_{\substack{i=1 \\ i \in S2}}^{|S2|} C_i + |R2|C_{S2} + |R2|t_2 + \sum_{\substack{i=1 \\ i \in R2}}^{|R2|} (|R2| - i + 1)p_i + \sum_{i=2}^a na_i(i-1)t_1$$

$$COSTb = \sum_{\substack{i=1 \\ i \in S2}}^{|S2|} C_i + |R2|C_{S2} + |R2|t_2 + \sum_{\substack{i=1 \\ i \in R2}}^{|R2|} (|R2| - i + 1)p_i + \sum_{i=2}^b nb_i(i-1)t_2$$

$$COST2 = \max(COSTa, COSTb)$$

End

Figure 4.7: Cost of Option 2 (Contd. HSTC).

Algorithm Update_Batch ($S1, R1, S2, R2, i, j, C, SA, SU, B, S$)

Begin

$A \leftarrow S2 - S1.$

$h \leftarrow 1$

Repeat

delete job $J_h, \forall h \in A$ from SU

Add job J_h to S , and SA .

$C_j \leftarrow C_{j-1} + p_h.$

$j \leftarrow j + 1.$

$h \leftarrow h + 1.$

Until ($h \leq |A|$)

Insert $M2$ to S .

$i \leftarrow i + 1.$

End

Figure 4.8: Insert job in batch (Contd. HSTC).

4.5 Implementation of TS and SA

Implementation of Tabu search (TS) and Simulated annealing (SA) requires an efficient Neighborhood scheme. We proposed a new Neighborhood scheme, details of scheme are given in Figure 4.9. In order to tune TS and SA, initial experiments are performed with different problems, some values of t_1 , t_2 , T_p and T_{max} are used to determine the suitable parameters for TS and SA. For Tabu search Candidate list size (CLS) is 150 and Tabu list size (TLS) is 13. For Simulated annealing Cooling rate (α) is .99, acceptance rate at initial temperature is kept at 90 % at least and Metropolis loop size (M) is same as Candidate list size. Both the algorithms stops at 10,000 non improving iterations.

4.6 Proposed Lower Bound

The Lower Bound (LB 2) could be obtained by using the idea presented in Figure3.7 but here the continuous working time limit $T = T_{max}$ and maintenance time and $t = t_1$.

Algorithm Neighbor.Gen 2

```

(* S = Schedule *)
(* S' = Neighbor schedule *)
(* K = Job sequence *)
(* M = Maintenance in sequence i.e M1 or M2 *)
(* Si = Member of S (either job or maintenance) at position i *)
(* pj = Processing time of job j. *)

Begin
  S' ← {0}, Bj ← {0}, i ← 1, j ← 1.
  Repeat
    If (Si = M) Then
      j ← j + 1
    Else
      Insert job at Si at the end of batch Bj
    Endif
    i ← i + 1
  Until (i ≤ |S|)
  Generate two unique random numbers a, and b with U(1, |j|).
  Generate x and y with U(1, |Ba|) and U(1, |Bb|).
  S' ← S
  Swap job at position x in batch Ba with job at position y in batch Bb respectively in S'.
  Sort jobs in batches a and b in SPT order.
  If (S' is feasible) Then
    Return
  Else
    i ← 1, j ← 1
    Repeat
      If (S'i ≠ M1 or S'i ≠ M2) Then
        Kj ← Si
        j ← j + 1
      Endif
      i ← i + 1
    Until (i ≤ |S'|)
  Endif
  Repeat
    i ← 1, q ← 0.0, D ← {0}
    Repeat
      Add job ji at the end of D
      q ← q + pi, ∀ i ∈ D
      Until (q ≤ Tmax or |K| = 0)
      Generate z ~ U(1, |D|), and remove the jobs beyond position z in D.
      i ← 1, q ← 0.0, S' ← {0}
      Repeat
        q ← q + pi, ∀ i ∈ D
        i ← i + 1
      Until (i ≤ z)
      If (q ≤ Tp) Then
        Insert jobs in D at the end of S' followed by M1 at the end of S'.
      Else
        Insert jobs in D at the end of S' followed by M2 at the end of S'.
      Endif
      Remove the jobs in D from K
    Until (|K| ≤ 0)
  End
End

```

Figure 4.9: Proposed neighborhood scheme (NBR 2).

4.7 Results

4.7.1 Comparison of Heuristic HSTC with TS and SA

The proposed heuristic (HSTC) is compared with TS and SA. Average improvement (AI) and Relative improvement average (RIA) made by TS and SA over HSTC of 10 randomly generated problems for each combination of T_p , t_1 , $\frac{T_{max}}{T_p}$ and $\frac{t_2}{t_1}$ is reported in Tables 4.1 - 4.4.

				TS			SA		
T_p	t_1	$\frac{T_{max}}{T_p}$	$\frac{t_2}{t_1}$	AI	%RIA	CPU time	AI	%RIA	CPU time
				sec			sec		
40	10	1.5	2	64	2.42	111	85	3.07	150
40	10	1.9	2	3	0.16	80	8	0.37	121
40	10	1.5	3	117	3.89	126	123	4.15	119
40	10	1.9	3	116	4.02	97	143	4.96	141
60	15	1.5	2	65	2.42	88	76	2.89	117
60	15	1.9	2	14	0.48	68	14	0.48	72
60	15	1.5	3	243	7.19	70	259	7.68	107
60	15	1.9	3	95	3.43	73	91	3.53	88
70	20	1.25	2	45	1.85	69	52	2.18	101
70	20	1.5	2	99	3.70	83	106	3.91	106
70	20	1.25	3	15	0.53	70	15	0.53	93
70	20	1.5	3	240	8.08	75	252	8.45	108

Table 4.1: Improvement made by TS and SA over HSTC for $n = 20$.

				TS			SA		
T_P	t_1	$\frac{T_{max}}{T_p}$	$\frac{t_2}{t_1}$	AI	%RIA	CPU time	AI	%RIA	CPU time
						sec			sec
40	10	1.9	2	66	0.86	177	66	0.86	259
40	10	1.5	3	347	5.23	165	351	5.10	263
40	10	1.9	3	301	4.35	230	299	4.41	254
60	15	1.5	2	213	3.64	155	213	3.65	226
60	15	1.9	2	15	0.22	127	15	0.22	150
60	15	1.5	3	777	10.59	193	825	11.17	233
60	15	1.9	3	285	4.43	103	334	5.21	202
70	20	1.25	2	126	2.28	123	165	2.99	226
70	20	1.5	2	197	3.25	118	219	3.55	185
70	20	1.25	3	66	0.87	142	66	0.87	224
70	20	1.5	3	678	9.62	166	743	10.50	268

Table 4.2: Improvement made by TS and SA over HSTC for $n = 30$.

				TS			SA		
T_P	t_1	$\frac{T_{max}}{T_p}$	$\frac{t_2}{t_1}$	AI	%RIA	CPU time	AI	%RIA	CPU time
				sec			sec		
40	10	1.5	2	179	2.79	229	258	3.15	348
40	10	1.9	2	95	1.06	219	130	1.46	311
40	10	1.5	3	374	4.44	209	243	2.60	318
40	10	1.9	3	440	4.57	310	437	4.62	337
60	15	1.5	2	272	3.44	241	270	3.38	313
60	15	1.9	2	18	0.20	135	50	0.60	197
60	15	1.5	3	945	9.19	139	986	9.59	270
60	15	1.9	3	440	4.87	227	474	5.24	244
70	20	1.25	2	254	3.30	125	321	4.18	269
70	20	1.5	2	240	2.91	135	283	3.37	261
70	20	1.25	3	81	0.85	133	97	1.00	346
70	20	1.5	3	985	9.89	146	1061	10.59	256

Table 4.3: Improvement made by TS and SA over HSTC for $n = 35$.

				TS			SA		
T_p	t_1	$\frac{T_{max}}{T_p}$	$\frac{t_2}{t_1}$	AI	%RIA	CPU time sec	AI	%RIA	CPU time sec
40	10	1.5	2	311	2.74	367	352	3.09	448
40	10	1.9	2	166	1.39	483	177	1.48	378
40	10	1.5	3	499	4.25	298	505	4.30	496
40	10	1.9	3	465	3.78	353	501	4.07	376
60	15	1.5	2	288	2.66	290	302	2.85	401
60	15	1.9	2	127	1.10	198	171	1.44	304
60	15	1.5	3	1155	8.81	279	1249	9.53	376
60	15	1.9	3	713	5.87	286	677	5.57	327
70	20	1.25	2	360	3.53	230	371	3.63	359
70	20	1.5	2	391	3.65	271	448	4.15	344
70	20	1.25	3	440	3.08	173	352	2.44	266
70	20	1.5	3	1297	9.86	180	1456	10.93	310

Table 4.4: Improvement made by TS and SA over HSTC for $n = 40$.

4.7.2 Deviation of TS, SA and HSTC from Lower Bound

The average of percent relative deviation from lower bound (LB 2) is calculated for Simulated annealing (SA), Tabu search (TS) and proposed heuristic (HSTC).

For each combination of $T_p = 40, 60, 70$, $t_1 = 10, 15, 20$, $\frac{T_{max}}{T_p} = 1.25, 1.5, 1.9$ and $\frac{t_2}{t_1} = 2, 3$. 10 randomly generated problem are tested. The average percentage

relative deviation of each method is reported in tables from Tables 4.5 - 4.6.

n	T_p	t_1	$\frac{T_{max}}{T_p}$	$\frac{t_2}{t_1}$	HSTC	TS	SA
20	40	10	1.5	2	13.37	10.63	9.89
20	40	10	1.9	2	10.34	10.16	9.92
20	40	10	1.5	3	17.26	12.63	12.32
20	40	10	1.9	3	18.73	13.95	12.83
20	60	15	1.5	2	11.20	8.50	7.99
20	60	15	1.9	2	9.20	8.66	8.66
20	60	15	1.5	3	19.14	10.56	9.98
20	60	15	1.9	3	16.39	12.39	12.30
20	70	20	1.25	2	9.26	8.28	6.87
20	70	20	1.5	2	12.49	8.33	8.09
20	70	20	1.25	3	8.06	8.06	7.49
20	70	20	1.5	3	20.15	10.43	9.99
30	40	10	1.9	2	12.30	11.33	11.33
30	40	10	1.5	3	21.63	15.21	15.39
30	40	10	1.9	3	20.90	15.65	15.58
30	60	15	1.5	2	12.77	8.66	8.66
30	60	15	1.9	2	9.97	9.89	9.73
30	60	15	1.5	3	23.12	10.08	9.36
30	60	15	1.9	3	18.76	13.50	12.57
30	70	20	1.25	2	9.77	7.22	6.46
30	70	20	1.5	2	13.79	10.09	9.75
30	70	20	1.25	3	8.64	8.16	7.68
30	70	20	1.5	3	23.07	11.17	10.11

Table 4.5: Percentage relative deviation of HSTC, TS and SA from lower bound.

n	T_p	t_1	$\frac{T_{max}}{T_p}$	$\frac{t_2}{t_1}$	HSTC	TS	SA
35	40	10	1.5	2	15.62	12.05	11.44
35	40	10	1.9	2	13.17	11.98	11.52
35	40	10	1.5	3	20.96	15.50	17.80
35	40	10	1.9	3	22.39	16.79	16.74
35	60	15	1.5	2	12.68	8.80	8.86
35	60	15	1.9	2	10.80	10.59	10.13
35	60	15	1.5	3	23.15	11.79	11.29
35	60	15	1.9	3	20.00	14.16	13.71
35	70	20	1.25	2	11.84	8.15	7.16
35	70	20	1.5	2	13.85	10.53	10.01
35	70	20	1.25	3	9.21	8.83	8.12
35	70	20	1.5	3	23.95	11.66	10.78
40	40	10	1.5	2	15.43	12.26	11.85
40	40	10	1.9	2	13.56	11.97	11.87
40	40	10	1.5	3	20.95	17.80	15.68
40	40	10	1.9	3	22.84	19.26	17.84
40	60	15	1.5	2	13.38	10.35	10.14
40	60	15	1.9	2	11.53	10.30	9.93
40	60	15	1.5	3	23.42	12.56	11.65
40	60	15	1.9	3	21.19	14.08	14.43
40	70	20	1.25	2	12.68	8.70	8.58
40	70	20	1.5	2	15.18	10.97	10.39
40	70	20	1.25	3	13.57	9.97	11.74
40	70	20	1.5	3	26.41	13.88	12.57

Table 4.6: Percentage relative deviation of HSTC, TS and SA from lower bound.

4.8 Discussion

The percentage relative deviation of HSTC heuristic from lower bound (LB 2) increases with the increase in number of jobs. This is because of problem size. As the problem size increases, the number of possible solutions increases exponentially. HSTC heuristic sequences jobs in non-decreasing order of their processing times and schedules maintenance time on that sequence either at continuous working time limit of T_p or at T_{max} . If there is no maintenance requirement, then job sequence on non decreasing order of their processing times is optimal. The total cost of schedule is cost of total processing time of jobs and cost of total maintenance time. HSTC heuristic is optimal with respect to the total cost of processing, if no maintenance is considered. It may not be the optimal with respect to the total cost of maintenance.

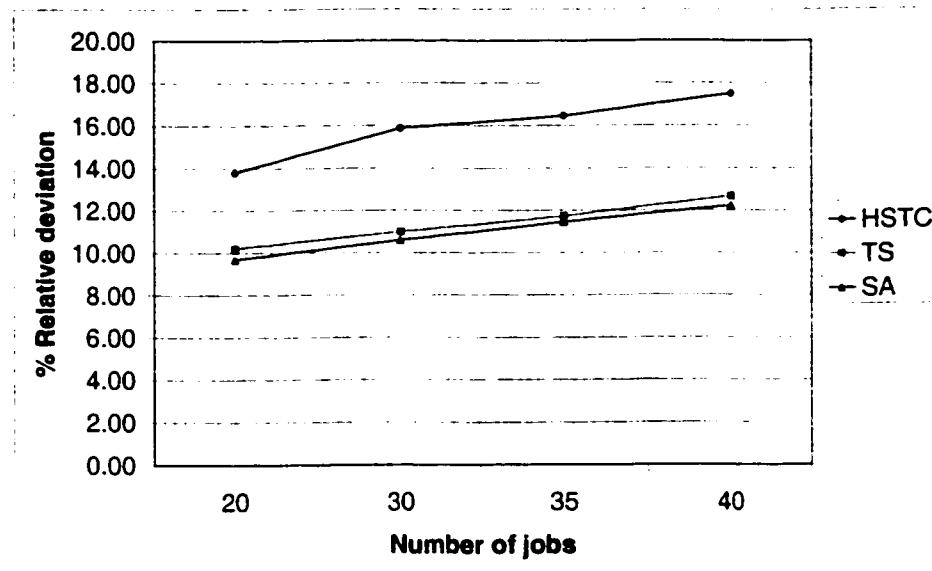


Figure 4.10: Effect of Number of jobs (n) on performance.

Effect of $\frac{t_2}{t_1}$				Effect of $\frac{T_{max}}{T_p}$			
$\frac{t_2}{t_1}$	HSTC	TS	SA	$\frac{T_{max}}{T_p}$	HSTC	TS	SA
2	10.97	9.09	8.57	1.5	15.60	10.18	9.71
3	16.62	11.34	10.82	1.9	13.66	11.29	10.93

Table 4.7: Effect of $\frac{t_2}{t_1}$ and $\frac{T_{max}}{T_p}$ on the percentage relative deviation from lower bound.

For a given ratio of T_{max} and T_p the relative error of SPT heuristic increases with the ratio of t_2 and t_1 , Table 4.7 shows the effect of $\frac{t_2}{t_1}$ on performance of HSTC, TS and SA heuristics. The comparison is made based on percentage relative deviation from lower bound (LB 2). Similarly Table 4.8 shows % relative error of HSTC when it is compared with TS and SA. This is due to the fact that in HSTC heuristic, total cost of maintenance times may not be optimal. Therefore, as the maintenance time t increases the contribution of total cost of maintenance times increases and its contribution to the total cost of resulting schedule becomes significant. Under this condition, the scheduling of maintenance becomes more important with sequencing of jobs. That is why the error of HSTC heuristic increases.

Effect of $\frac{t_2}{t_1}$			Effect of $\frac{T_{max}}{T_p}$		
$\frac{t_2}{t_1}$	TS	SA	$\frac{T_{max}}{T_p}$	TS	SA
2	2.20	2.50	1.5	4.49	4.71
3	5.49	5.71	1.9	2.82	3.10

Table 4.8: Effect of $\frac{t_2}{t_1}$ and $\frac{T_{max}}{T_p}$ on the percentage relative error of HSTC.

Similarly, for a given ratio of t_2 and t_1 , the error of HSTC heuristic increases with the decrease in $\frac{T_{max}}{T_p}$. This is because of the fact that as the maximum continuous working time limits T_{max} and T_p decreases, the maintenance constraint gets tighter (i.e., more requirement of maintenance). As discussed earlier, the total cost of schedule is total cost of processing times of jobs and the total cost of maintenance time. HSTC heuristic is optimal with respect to contribution of total processing times of jobs, but it may not be the optimal for total contributions from maintenance times. With reduction of $\frac{T_{max}}{T_p}$, the maintenance requirement increases and total contribution of maintenance time becomes significant and HSTC heuristic produces higher relative errors. Table 4.7 also shows the effect of $\frac{T_{max}}{T_p}$ on the performance of three heuristics. The comparison is made based on the percentage relative deviation from lower bound (LB 2). Similarly Table 4.8 shows the effect on error of HSTC when it is compared with TS and SA.

Chapter 5

Early-Tardy Minimization

In this chapter, we will study the problem of scheduling jobs on a single machine, that is subject to maintenance for another performance measure. The study is an extension to the problem studied by X. Qi et al. [4] for a different performance measure. The measure is to minimize total earliness and tardiness of jobs about a common due date.

5.1 Assumptions

All assumptions used in X. Qi et al. [4] are applicable. Other assumptions related to this problem are following:

1. All jobs have a common due date that is deterministic and known in advance.

2. The due date is large enough to be unrestrictive.

5.2 Notations

A_i	=	Batch i containing tardy jobs.
B_i	=	Batch i containing early jobs.
t	=	Time to perform preventive maintenance.
qa_i	=	Summation of processing times of jobs in batch A_i .
qb_i	=	Summation of processing times of jobs in batch B_i .
na_i	=	Number of job in batch A_i .
nb_i	=	Number of job in batch B_i .
La	=	Total number of batches of tardy jobs.
Lb	=	Total number of batches of early jobs.
T	=	allowable continuous working time limit.
SA	=	$\{A_1, A_2, A_3, \dots, A_{La}\}$.
SB	=	$\{B_1, B_2, B_3, \dots, B_{Lb}\}$.
SAT	=	$\{A_1, t, A_2, t, A_3, t, \dots, t, A_{La}\}$.
SBT	=	$\{B_1, t, B_2, t, B_3, t, \dots, t, B_{Lb}\}$.

5.3 Properties of Optimal Schedule

Some properties of an optimal schedule are developed in this section.

Property 5.1 *In an optimal schedule the due date d can not be during the maintenance period.*

Proof

We will prove it by contradiction argument. Suppose there is an optimal schedule S such that the maintenance is performed before due date d and continued after

d. Now consider the case where $|SB| > |SA|$ (Number of early jobs are more than number of tardy jobs). Consider another schedule S' such that maintenance is performed immediately after d , and the rest of the configuration in S' is the same as S . Let Z be the cost of schedule S and Z' be the cost of schedule S' . Let t_B and t_A be maintenance times before d and after d respectively. Due to the unrestrictive assumption of due date, changes could be made such that $t_A = t_B$ without violating any feasibility constraint. The change in the cost of the two schedules due to this shift is

$$Z' - Z = -|SB|t_B + |SA|t_B < 0$$

Since $|SB| > |SA|$, therefore the cost of the schedule improves.

Similarly if $|SB| \leq |SA|$, then a new schedule S' could be obtained by shifting maintenance such that it is finishing at d . In case $|SB| < |SA|$, it is obvious that the cost of new schedule S' will improve. In case $|SB| = |SA|$, an alternative schedule could be obtained.

$$Z' - Z = -|SB|t_A + |SA|t_A \leq 0$$

Corollary 5.1 *The problem of total tardiness minimization about $d = 0$ is equivalent to total completion time minimization.*

Proof

When the due date is zero no job could be processed earlier than the due date d , hence the problem is to minimize the average deviation from the completion time of all jobs about the a common due date d . Since all the deviations are measured from a common value d i.e, $d = 0$, therefore minimizing average completion time from zero ready time and minimizing average deviation of completion time minimization about $d = 0$ results in the same solution (Schedule).

Corollary 5.2 *The optimal schedule to a total completion time minimization problem could be transformed to obtain optimal schedule to a total earliness minimization problem about an infinitely large due date.*

Proof

Schedule the jobs for objective of total completion time minimization ignoring the due date d . Let the optimal schedule S be $\{B_1, t, B_2, t, \dots, B_{L-1}, t, B_L\}$, where B_i is the i th batch of jobs and L is the total number of batches. C_{max} is the resulting makespan of the schedule S . The schedule can be transformed to total earliness minimization about a common due date d . The optimal schedule to total earliness minimization will be $S' = \{B'_L, t, B'_{L-1}, t, \dots, t, B'_2, t, B'_1\}$, where batches B'_i and B_i , represents the same jobs, but in batch B'_i all jobs are arranged in non increasing order of their processing times. The schedule S' starts at $d - C_{max}$.

One may conclude from corollary 5.1 that the optimal schedule for the early-tardy scheduling problem with unrestrictive due date is not necessarily V- shape, since for above degenerate case X. Qi et al. [4] showed the optimal schedule is not necessarily SPT schedule for total completion time minimization.

Property 5.2 *Maintenance operations after the due date should be scheduled as late as possible.*

Proof

Consider $S = (SBT, SAT)$ with tardy set $SAT = \{A_1, t, \dots, t, A_i, t, A_{i+1}, t, \dots, A_{L\alpha}\}$, such that there is a batch A_i , with J_i at last position in batch A_i and J_k is the first job in the immediately following batch A_{i+1} , qa_i is the summation of the processing times in A_i , and $qa_i + p_k \leq T$. Another feasible schedule S' could be obtained by adding the job J_k at the last position of batch A_i . The tardiness of the job J_k will be decreased by an amount t , and the tardiness of rest of the jobs in the both schedules will remain unchanged, thus schedule S' is improved over schedule S . This procedure could be continued till none of the jobs in the batches following batch A_i could be inserted to A_i without delaying maintenance for a period more than T . This rule applicable to all batches of tardy jobs.

Property 5.3 *Maintenance operations before the due date should be scheduled as far as possible from the due date.*

Proof

The argument similar to proof of property 5.2 will be used. Consider a schedule $S = (SBT, SAT)$ with early set $SBT = \{B_1, t, \dots, B_{i-1}, t, B_i, t, \dots, t, B_{Lb}\}$, such that there is a batch B_{i-1} with J_k at the last position. Batch B_{i-1} is immediately followed by B_i . Let J_i is the first job in batch B_i and qb_i is the summation of the processing times of batch B_i such that $qb_i + p_k \leq T$. Another feasible schedule S' could be obtained by putting job J_k at the first position of batch B_i . The earliness of job J_k will decrease by t , and earliness of the rest of the jobs in schedule S' will remain same. This procedure could be continued till none of the jobs in the batches preceded by B_i could be inserted without delaying maintenance for a continuous working time more than T .

Property 5.4 *In an optimal schedule jobs in each of the tardy batches are sequenced in non decreasing order of their processing times (SPT order).*

Proof

We will prove this property by an interchanging argument of two adjacent jobs in a batch. Consider a schedule S such that there are two adjacent jobs J_k and J_i in batch B_i with $p_k > p_i$. Another schedule S' could be obtained by interchanging the positions of jobs J_i and J_k

$$\text{Tardiness of jobs } i \text{ and } k \text{ in } S = (a + p_k - d) + (a + p_k + p_i - d)$$

$$\text{Tardiness of jobs } i \text{ and } k \text{ in } S' = (a + p_i - d) + (a + p_k + p_i - d)$$

$$\text{Total change in cost of two schedule} = p_i - p_k < 0$$

The cost of the two schedules will be the same for all the remaining jobs. Hence S is superior to S' .

Property 5.5 *In an optimal schedule the jobs in each batch of early job set are sequenced in non increasing order of their processing times (LPT order).*

Proof

We will again use interchanging argument. Consider a schedule S such that there are two adjacent jobs J_i and J_k in batch B_i with $p_k > p_i$. Another schedule S' could be obtained by interchanging the jobs J_i and J_k in schedule S .

$$\text{Earliness of jobs } i \text{ and } k \text{ in } S = (d - a + p_i) + (d - a + p_k + p_i)$$

$$\text{Earliness of jobs } i \text{ and } k \text{ in } S' = (d - a + p_k) + (d - a + p_k + p_i)$$

$$\text{Total change in cost of two schedule} = p_i - p_k < 0$$

The cost of the two schedules will be the same for all the remaining jobs. Hence S' is superior to S .

Property 5.6 *In an optimal schedule, all the batches of the jobs in tardy set are sequenced in non decreasing order of $\frac{qa_i}{na_i}$.*

$$\frac{qa_i}{na_i} \leq \frac{qa_{i+1}}{na_{i+1}} \quad i = 1, 2, 3, \dots, La - 1 \quad (5.1)$$

Proof

This property is the extension of the SPT rule on batch level. Similar interchange argument can be used to prove this property.

Property 5.7 *In optimal schedule, all the early batches are sequenced in non increasing order of $\frac{qb_i}{nb_i}$.*

$$\frac{qb_i}{nb_i} \geq \frac{qb_{i+1}}{nb_{i+1}} \quad i = 1, 2, 3, \dots, Lb - 1 \quad (5.2)$$

Proof

The argument is similar to the one given in property 5.5 on batch level.

5.4 Proposed Heuristic

We have proposed a heuristic (HSET) based on the properties of optimal schedule developed in the previous section. The idea is to schedule the jobs in V-shape about the common due date d . The last job in the early set is scheduled on the due date. The maintenance is scheduled starting from first tardy job till the last tardy job as late as possible. Similarly, the maintenance is schedule starting from last early job till the first early job as late as possible. The effort is made to schedule

maintenance along with jobs such that total deviation from the due date of all the jobs scheduled is minimized. Details of HSET heuristic are given in Figures 5.1, 5.2 and 5.3 respectively.

Algorithm HSET

```

(* S = Schedule *)
(* SA = Tardy jobs set *)
(* SB = early jobs set *)
(* U = Universal job set in non increasing order of  $p_i$  *)
Begin
  SA ← {0}, SB ← {0}, i ← 1, i ∈ U.
  Repeat
    If ( $U_i \neq \emptyset$ ) Then
      Add job  $J_i$  to the end of SB
      i ← i + 1
    Endif
    If ( $U_i \neq \emptyset$ ) Then
      Add job  $J_i$  to the end of SA
      i ← i + 1
    Endif
  Until ( $i \leq |U|$ )
  Call Insert_SA2_M( $p, t, T, SA, SA'$ )
  Call Insert_SB2_M( $p, t, T, SB, SB'$ )
  Call Improve_SA'( $SA', SA''$ )
  Call Improve_SB'( $SB', SB''$ )
  S ← ( $SB'', SA''$ )
End

```

Figure 5.1: Proposed heuristic HSET.

5.5 Implementation of TS and SA

A neighborhood scheme (NBR 3) is proposed to search the solution space. Initial temperature was determined using the criteria discussed in Section 3.6.3. Initial experimentation was performed to chose the suitable parameters for Tabu search (TS) and Simulated annealing (SA). In our implementation of Simulated annealing (SA), the initial temperature is kept high enough such that at the initial temperature at-

Algorithm Insert_SA2_M(p, t, T, SA, SA')

(• S = Schedule •)
 (• SA = Tardy jobs set •)
 (• SA' = Tardy jobs set with maintenance inserted •)
 (• qa_i = Summation of processing times in batch i •)
 (• T = Max delay allowed •)
 (• M = Maintenance in sequence •)

Begin
 $SA' \leftarrow \{\emptyset\}, i \leftarrow 1, j \leftarrow 1 \forall j \in SA, T' \leftarrow \frac{T}{2}$
Repeat
 $qa_i \leftarrow 0.0$
 If ($qa_i + p_j \leq T'$) **Then**
 $qa_i \leftarrow qa_i + p_j$
 Insert J_j last position of SA'
 $j \leftarrow j + 1$
 Else
 $T' \leftarrow T$
 $i \leftarrow i + 1$
 $qa_i \leftarrow 0.0$
 Insert M last position of SA'
 Endif
Until ($j \leq |SA|$)
End

Figure 5.2: Insert maintenance in tardy set (Contd. HSET).

Algorithm Insert_SB2_M(p, t, T, SB, SB')

(• SB = early jobs set •)
 (• SB' = early jobs set with maintenance inserted •)
 (• qb_i = Summation of processing times in batch i •)
 (• T = Max delay allowed •)

Begin
 $SB' \leftarrow \{\emptyset\}, i \leftarrow 1, j \leftarrow |SB| \forall j \in SB, T' \leftarrow \frac{T}{2}, qb_i \leftarrow 0.0$
Repeat
 If ($qb_i + p_j \leq T'$) **Then**
 $qb_i \leftarrow qb_i + p_j$
 Insert J_j to the first position in SB'
 $j \leftarrow j - 1$
 Else
 $T' \leftarrow T$
 $i \leftarrow i + 1$
 $qb_i \leftarrow 0.0$
 Insert M to the first position in SB'
 Endif
Until ($j \geq 1$)
End

Figure 5.3: Insert maintenance to early set (Contd. HSET).

least 90% of the moves are accepted. The temperature is kept high to moderate the annealing process. It effects its ability to converge to a global minimum. Cooling rate is taken at 0.99, and Metropolis loop is called at least 100 times at each temperature level. In Tabu search implementation the candidate list size is taken as 100, and tabu list size is 13 for problems with number of jobs more than 20, otherwise it is 7. Both algorithms are allowed to run to 10,000 non improving iterations. The major issue in implementation of Tabu search and Simulated annealing is neighborhood scheme. Details of the neighborhood scheme is explained in Figures 5.4 - 5.6.

Algorithm Nbrhood_gen 3

```

(* S = Schedule *)
(* S' = Neighbor schedule *)
(* M = Maintenance in sequence *)
(* T = Max delay allowed *)
Begin
  A ← {0}, B ← {0}, i ← 1, j ← 1, v ← ⌈ $\frac{n}{2}$ ⌉
  Repeat
    Insert member (job or maintenance) at Si to B at position j.
    If (Si ≠ M) i ← i + 1
      j ← j + 1
    Until (i ≤ v)
    If (Si+1 = M) Then
      Bj+1 ← M, j ← 1
      Repeat
        Aj ← Si
        j ← j + 1
      Until (i ≤ |S|)
    Else
      j ← 1
      Repeat
        Aj ← Si
        j ← j + 1, i ← i + 1
      Until (i ≤ |S|)
    End if
  End if

  Generate x with U(0, 1)

  If (x ≤  $\frac{1}{2}$ )
    Call Scheme1 (S, S')
  Else
    Call Scheme2 (S, S')
  Endif
  If (S' infeasible) than
    Call insert_M (p, S, S')
  Endif
  Sort all Early batches in LPT and all Tardy batches in SPT order in S'.
End

```

Figure 5.4: Proposed neighborhood (NBR 3) for Early-Tardy minimization.

Algorithm Scheme1(S,S')

```

Repeat
  Generate two distinct integer random numbers a and b with  $U(1, |S|)$ .
  Do  $i \leftarrow a, b$ 
    If ( $S_i = M$ ) Break
  Enddo
Until(Break)
 $S' \leftarrow S$ 
Swap  $S'_a$  with  $S'_b$ 

```

Figure 5.5: Algorithm for scheme 1 (Contd. NBR 3).

Algorithm Scheme2(S,A,B,S')

```

Repeat
  Generate two distinct integer random number a and b with  $a \sim U(1, |B|)$ ,  $b \sim U(1, |A|)$ .
  If ( $S_a \neq M$  and  $S_b \neq M$ ) Break
Until(Break)
 $S' \leftarrow S$ 
Swap  $S'(a)$  with  $S'(b)$ 

```

Figure 5.6: Algorithm for scheme 2 (Contd. NBR 3).

5.6 Proposed Lower Bound for Early Tardy Minimization

The approach of relaxing maintenance constraint with the properties of the optimal schedule developed in Section 5.3 is used to construct the lower bound. The algorithm starts by considering the optimal schedule for the problem when no maintenance is required (Kanet's Algorithm). Maintenance operations are then inserted in schedule such that there is at most one job violating maintenance constraint T in each batch. Details of lower bound procedure is explained in Figure 5.7.

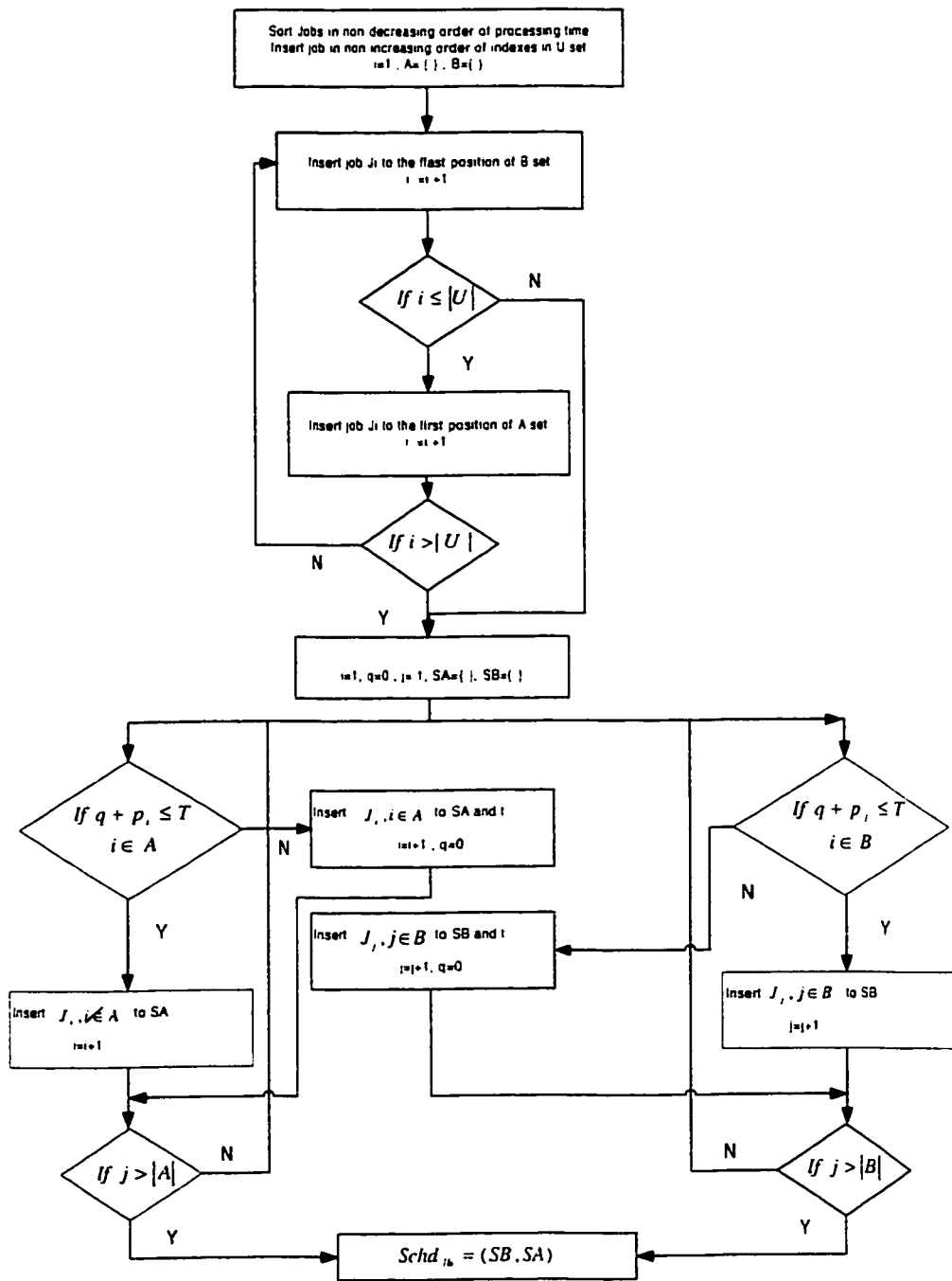


Figure 5.7: Proposed lower bound for Early -Tardy minimization.

5.7 Results

5.7.1 Comparison of Heuristic HSET with TS and SA

The proposed heuristic (HSET) is compared with the Tabu search (TS) and Simulated annealing (SA) in this section. The average percentage relative improvement of Tabu search (TS) and Simulated annealing (SA) over the proposed heuristic HSET is computed. This study will give an idea about the error of HSET heuristic and effect of maintenance parameters on performance. Tables 5.1 - 5.4 summarizes the results obtained. For each combination of T ranging from 50 to 80 and t ranging from 10 to 40, 10 problems are solved by each algorithm and Average Improvement (AI) and Relative Improvement Average (RIA) over HSET is computed based on the solution found by TS and SA respectively. In all the test problems the processing times are randomly generated with uniform distribution from 1 to 30 and the problem sizes are 10, 20, 35 and 50. The common due date d is taken by summing the processing times of all job and maintenance time inserted in the sequence if the preemption is allowed. The due date is determined by following equation.

$$d = \sum_{i=1}^n p_i + \left[\frac{\sum_{i=1}^n p_i}{T} \right] \quad (5.3)$$

		TS			SA		
T	t	AI	%RIA	CPU time sec	AI	%RIA	CPU time sec
50	10	15	3.31	33	20	5.63	43
50	20	20	3.38	33	27	6.17	44
50	30	38	5.81	30	40	6.09	47
50	40	65	14.02	38	40	14.41	43
60	10	7	1.79	39	7	1.79	40
60	20	8	2.03	37	8	2.03	42
60	30	28	4.86	30	6	1.16	42
60	40	80	13.08	30	79	12.90	42
70	10	7	1.94	28	7	1.94	41
70	20	7	2.64	33	7	2.64	40
70	30	30	6.95	36	20	5.34	40
70	40	12	1.78	33	12	1.78	40
80	10	1	0.45	36	1	0.45	40
80	20	8	1.50	37	8	1.50	41
80	30	10	3.68	23	10	3.68	41
80	40	13	3.30	36	13	2.08	42

Table 5.1: Average improvement made by TS and SA over HSET for $n = 10$.

		TS			SA		
<i>T</i>	<i>t</i>	AI	%RIA	CPU time sec	AI	%RIA	CPU time sec
50	10	38	2.92	83	38	2.92	110
50	20	60	4.30	93	60	4.30	115
50	30	130	5.65	90	133	5.75	114
50	40	111	5.60	81	112	5.65	123
60	10	33	2.76	76	25	2.05	95
60	20	80	5.17	77	80	5.17	98
60	30	92	4.75	81	80	4.15	119
60	40	95	5.39	79	95	5.39	83
70	10	3	0.33	74	9	0.92	89
70	20	46	3.14	74	52	3.55	86
70	30	78	4.02	75	87	4.55	88
70	40	78	4.09	76	88	4.88	94
80	10	17	1.27	74	17	1.27	79
80	20	60	3.76	73	60	3.76	99
80	30	29	2.08	71	40	2.78	81
80	40	27	1.55	69	35	2.02	89

Table 5.2: Average improvement made by TS and SA over HSET for $n = 20$.

		TS			SA		
T	t	AI	%RIA	CPU time sec	AI	%RIA	CPU time sec
50	10	63	1.68	189	69	1.86	302
50	20	165	3.28	201	187	3.68	260
50	30	288	4.55	233	317	5.00	271
50	40	508	7.19	207	502	7.12	238
60	10	50	1.39	156	52	1.46	231
60	20	79	1.85	292	68	1.63	239
60	30	164	2.92	211	167	2.98	247
60	40	233	3.90	173	214	3.59	254
70	10	51	1.57	185	44	1.33	235
70	20	127	3.03	183	137	3.25	204
70	30	173	3.02	180	184	3.24	238
70	40	184	3.19	161	200	3.49	230
80	10	47	1.18	172	46	1.17	199
80	20	57	1.16	174	65	1.29	222
80	30	118	2.40	167	132	2.70	229
80	40	141	2.77	203	146	2.85	230

Table 5.3: Average improvement made by TS and SA over HSET for $n = 35$.

		TS			SA		
T	t	AI	%RIA	CPU time sec	AI	%RIA	CPU time sec
50	20	273	2.55	514	340	3.18	529
50	30	575	4.79	541	606	5.04	473
50	40	460	3.18	511	501	3.45	538
60	10	56	0.69	302	65	0.80	422
60	20	164	2.03	422	179	2.19	429
60	30	388	3.38	470	387	3.37	488
60	40	241	2.01	425	241	2.01	423
70	10	30	0.42	299	43	0.59	442
70	20	165	1.99	292	162	1.96	450
70	30	247	2.38	389	248	2.38	436
70	40	133	1.21	431	139	1.27	427
80	10	43	0.61	309	36	0.51	391
80	20	100	1.05	383	91	0.95	368
80	30	198	2.11	430	191	2.02	426
80	40	207	2.03	328	224	2.20	391

Table 5.4: Average improvement made by TS and SA over HSET for $n = 50$.

5.7.2 Deviation of TS, SA and HSET from Lower Bound

To test the performance of the three heuristics TS, SA and HSET, their average percentage relative deviation from the proposed lower bound (LB 3) is computed.

This study will give an idea about how far the solution methods are from their super

optimal values (Lower Bounds). In Tables 5.5 to 5.8 average percentage of relative deviation is presented for each combination of T ranging from 50 to 80 and t ranging from 10 to 40, 10 problems are solved and the average percentage relative deviation is reported for each method.

n	T	t	HSET	TS	SA
10	50	10	13.58	8.49	6.79
10	50	20	20.04	15.98	12.62
10	50	30	30.44	22.58	22.22
10	50	40	34.44	17.90	17.90
10	60	10	6.57	4.62	4.62
10	60	20	17.21	15.24	15.24
10	60	30	24.76	18.65	23.34
10	60	40	32.40	15.24	15.24
10	70	10	7.64	5.51	5.51
10	70	20	11.98	9.11	9.11
10	70	30	18.54	10.35	12.34
10	70	40	23.57	21.40	21.40
10	80	10	5.90	5.43	5.43
10	80	20	9.07	7.39	7.39
10	80	30	14.30	10.17	10.17
10	80	40	21.06	16.95	18.68

Table 5.5: Average deviation of TS, SA and HSET from lower bound for $n = 10$.

n	T	t	HSET	TS	SA
20	50	10	8.25	5.09	5.09
20	50	20	13.96	9.01	9.02
20	50	30	20.17	14.17	13.26
20	50	40	21.78	15.68	14.75
20	60	10	7.11	4.14	4.93
20	60	20	12.17	6.34	6.34
20	60	30	16.82	11.24	11.94
20	60	40	19.59	13.37	14.57
20	70	10	4.11	3.77	3.15
20	70	20	8.74	5.30	4.84
20	70	30	14.15	10.15	8.96
20	70	40	13.42	8.68	7.82
20	80	10	4.31	2.98	2.98
20	80	20	8.97	4.88	4.88
20	80	30	9.83	7.52	6.75
20	80	40	10.96	9.23	8.69

Table 5.6: Average deviation of TS, SA and HSET from lower bound for $n = 20$.

n	T	t	HSET	TS	SA
35	50	10	5.90	4.11	3.92
35	50	20	12.12	9.07	7.99
35	50	30	14.78	9.50	8.98
35	50	40	20.73	12.05	12.53
35	60	10	5.38	3.91	3.83
35	60	20	7.06	5.07	5.31
35	60	30	12.53	9.23	9.16
35	60	40	14.72	10.23	10.60
35	70	10	4.12	2.48	2.73
35	70	20	7.86	4.57	4.34
35	70	30	10.50	7.49	6.92
35	70	40	13.47	9.83	9.49
35	80	10	4.05	2.82	2.84
35	80	20	5.78	4.55	4.41
35	80	30	9.63	7.00	6.67
35	80	40	10.75	7.68	7.59

Table 5.7: Average deviation of TS, SA and HSET from lower bound for $n = 35$.

n	T	t	HSET	TS	SA
50	50	10	6.70	5.06	4.87
50	50	20	12.01	9.16	8.45
50	50	30	13.22	7.71	7.70
50	50	40	17.93	14.17	13.86
50	60	10	4.13	3.41	3.29
50	60	20	7.54	5.36	5.18
50	60	30	11.87	8.10	8.11
50	60	40	12.15	9.90	10.45
50	70	10	3.33	2.90	2.72
50	70	20	7.04	4.91	4.94
50	70	30	8.91	6.32	6.32
50	70	40	10.47	9.12	9.06
50	80	10	2.80	2.17	2.28
50	80	20	5.61	4.50	4.61
50	80	30	8.42	6.14	6.23
50	80	40	8.48	6.28	6.09

Table 5.8: Average deviation of TS, SA and HSET from lower bound for $n = 50$.

5.8 Discussion

Comparative study is made over the proposed heuristic HSET with Tabu search (TS) and Simulated annealing (SA). Tables 5.1- 5.4 show relative error of HSET heuristic when it is compared with TS and SA. Tables 5.5 - 5.8 extend the same idea but here, the comparison of three heuristics HSET, TS and SA is made with the proposed lower bound (LB 3). In Tables 5.1 - 5.4 the following observations could be made:

The error of heuristic HSET increases with the increase in the problem size. HSET

sequences jobs in V- shape about the common due date d and the maintenance is scheduled on that V-shape sequence. The total cost of schedule is the cost of total earliness and tardiness caused by the jobs and the cost by maintenance. The V-shape job arrangement is optimal if no maintenance is performed. HSET is optimal for early-tardy problem where machine is continuously available. It is not optimal when planned maintenance is involved. With the increase in number of jobs, maintenance requirement also increases and therefore contributions from maintenance increase and become significant. Figure 5.8 shows the variations in the average percentage relative deviation of the three methods with the number of jobs.

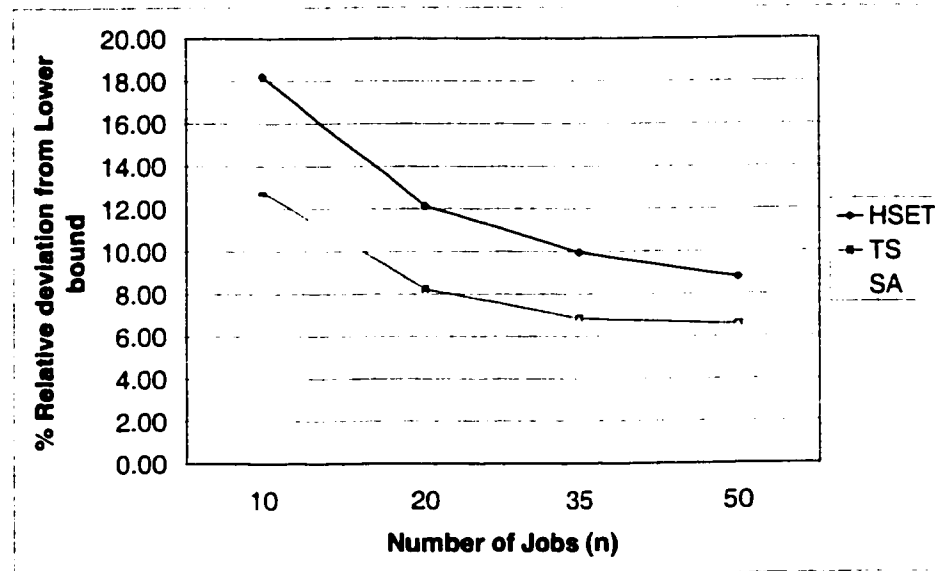


Figure 5.8: Effect of number of jobs on performance.

For a given value of maximum continuous working time T , the error of HSET heuristic increases with the maintenance time t . Figures 5.9 and 5.10 show the effect of

different maintenance time t on the error of HSET heuristic. As the maintenance time t increases the contribution of total cost of maintenance increases and its contribution to the total earliness and tardiness of resulting schedule becomes significant. Similarly for a given value of maintenance time t , the error of HSET heuristic increases with the decrease in maximum continuous working time T of machine. As

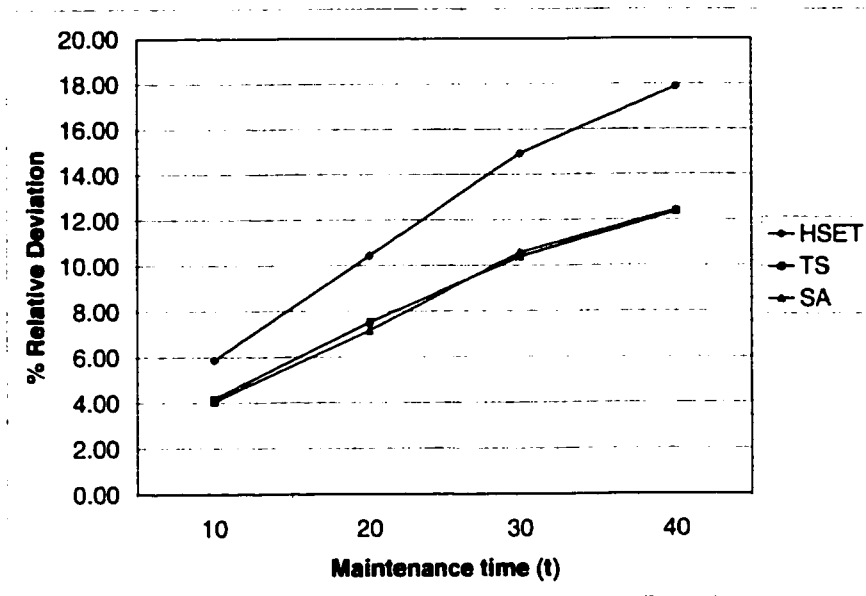


Figure 5.9: Effect of maintenance time t on performance.

the maximum continuous working time T decreases the maintenance constraint gets tighter i.e., more maintenance is required. With reduction of T , the maintenance requirement increases and total contribution of maintenance time becomes significant and HSET heuristic produces higher relative errors. Figure 5.11 the effect of max allowed time to do maintenance T on the performance of the three heuristics and Figure 5.12 shows the effect of T on the error of heuristic HSET when it is

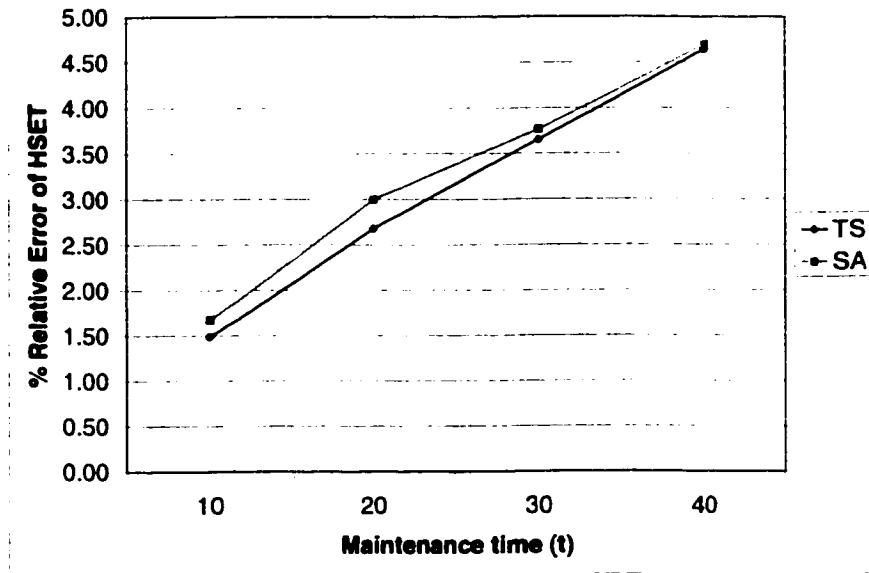


Figure 5.10: Effect of maintenance time t on HSET error.

compared with TS and SA. Tables 5.5 - 5.8 show the deviation of heuristics HSET, TS and SA from lower bound (LB 3). Lower bound (LB 3) is also sensitive to T and t . Since lower bound is obtained from relaxation of maintenance constraint, as the maintenance requirement increases (i.e., maximum continuous working time of machine T decreases), the lower bound gets weaker (results lower value). With the increase in maintenance time t , deviation to the maintenance constraint violation increases and hence the lower bound becomes weaker.

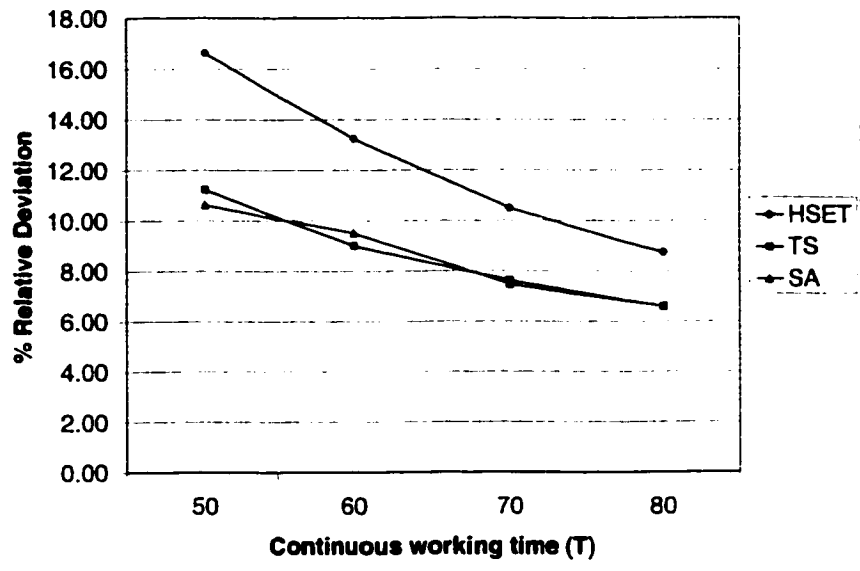


Figure 5.11: Effect of continuous working time T on performance.

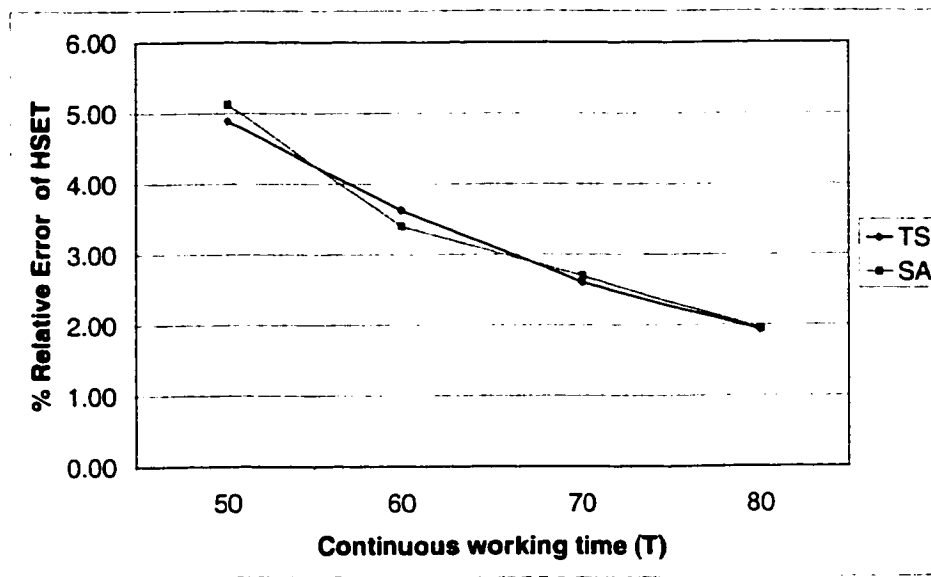


Figure 5.12: Effect of continuous working time T on HSET error.

Chapter 6

Summary, Conclusion and Future Work

In this thesis, jobs scheduling and maintenance operations on a single machine is studied. Tabu search (TS) and Simulated annealing (SA) approaches are developed to the problem proposed by X. Qi et al. [4], and to two other extensions. Extensions to the problem are analytically studied and heuristics are proposed. A brief summary of the work done, followed by conclusions and some future extensions of this work is provided in following sections.

6.1 Summary

The main contributions to the thesis are:

- Tabu search (TS), Simulated annealing (SA) approaches and a lower bound are developed to problem of X. Qi et al. [4]. The Branch and Bound (BNB) algorithm is implemented to evaluate the methods. The general factorial design of experiments are performed to determine the suitable parameters for Tabu search (TS) and Simulated annealing (SA). Solutions obtained by Tabu search (TS) and Simulated annealing (SA) are within 1 % of optimal solutions obtained by Branch and Bound (BNB) for the objective of total completion time minimization. In real time applications the proposed heuristic is more practical than the Tabu search (TS) and Simulated annealing (SA) when computation time is a significant factor.
- Comparisons are made between TS, SA and BNB for small size problems. For Large size problem, the average relative deviation from Lower Bound is studied for the heuristic (SPT) proposed by X. Qi et al. [4], Tabu search (TS) and Simulated annealing (SA) approaches.
- Two extensions to the problem are studied. In the first one the maintenance is allowed to be delayed with longer maintenance duration. The second extension studies the original problem with total earliness and tardiness as a performance measure. For each problem the following is done:
 - Properties of the optimal schedule are derived analytically.
 - A constructive heuristic is proposed based on the properties developed.

- Tabu search (TS) and Simulated annealing (SA) approaches are developed and comparative study is made between TS, SA and proposed heuristic.
- A lower bound is developed and the relative deviation of TS, SA and the proposed heuristic from the lower bound is studied.

In the situations where the maximum continuous working time limit is high the proposed heuristics to extensions find good solution for small values of maintenance time. As the maintenance time increases, the performance of heuristics becomes worse and Simulated annealing and Tabu search get good solutions. Therefore it is recommended to use heuristics when the maintenance time is low and maximum continuous working limit is high.

6.2 Future Work

Scientific research is an ongoing process and there is always some room for improvement. The following is a brief list of suggestions for possible future work in this area.

- The problem can be considered for scheduling maintenance activity on parallel machines.
- Error bound analysis of proposed heuristics can be carried out.

- The development of Hybrid evolutionary techniques can also be considered for the original problem and its proposed extensions.
- Other objectives related to the due date like total lateness, maximum lateness can be considered as future work.
- Unplanned maintenance requirement may also be considered.
- Multiple machine scheduling problem with maintenance requirement can also be considered for a future research.

Appendix A

DOE for Setting TS Parameters w.r.t Relative Improvement

Design of experiments for Tabu Search with Objective value = (initial cost- best cost observed)/initial cost

A = Number of Jobs (n) B = Candidate list size C = Tabu list size Levels of
A = 3 (10 50 100) Levels of B = 2 (10 20) Levels of C = 2 (7 11)

Dependent Variable	R-squared (percent)	Adjusted R-squared	Est. Std. Dev. of Model Error	Coefficient of Mean	Coefficient of Var. (percent)
Y	.000	.000	8.464	19.5	43.4

* * * Analysis of Variance * * *

Sum of	Mean	Prob. of
--------	------	----------

Source	DF	Squares	Square	Overall F	Larger F
Model	11	.0	.00	.000	1.0000
Error	204	14614.1	71.64		
Corrected Total	215	14614.1			

*** Variation Due to the Model ***

Source	DF	Sum of Squares	Prob. of F	Larger F
A	2	0	.000	1.0000
B	1	0	.000	1.0000
C	1	0	.000	1.0000
A*B	2	0	.000	1.0000
A*C	2	0	.000	1.0000
B*C	1	0	.000	1.0000
A*B*C	2	0	.000	1.0000

*** Subgroup Means ***

A Means (N=72)

1 19.5017

2 19.5017

3 19.5017

B Means (N=108)

1 19.5017

2 19.5017

C Means (N=108)

1 19.5017

2 19.5017

A*B Means (N=36)

1 1 19.5017

1 2 19.5017

2 1 19.5017

2 2 19.5017

3 1 19.5017

3 2 19.5017

A*C Means (N=36)

1 1 19.5017

1 2 19.5017

2 1 19.5017

2 2 19.5017

3 1 19.5017

3 2 19.5017

B*C Means (N=54)

1 1 19.5017

1 2 19.5017

2 1 19.5017

2 2 19.5017

A*B*C Means (N=18)

1 1 1 19.5017

1 1 2 19.5017

1 2 1 19.5017

1	2	2	19.5017
2	1	1	19.5017
2	1	2	19.5017
2	2	1	19.5017
2	2	2	19.5017
3	1	1	19.5017
3	1	2	19.5017
3	2	1	19.5017
3	2	2	19.5017

Appendix B

DOE for Setting TS Parameters w.r.t CPU Time

Design of experiment with Cpu time A = number of jobs B = Candidate list size (10 20) %
C = Tabu list size (7 11)

Dependent Variable	R-squared (percent)	Adjusted R-squared	Est. Std. Dev. of Model Error	Mean	Coefficient of Var. (percent)
Y	96.381	96.185	1.297	7.949	16.32

* * * Analysis of Variance * * *

Source	DF	Sum of Squares	Mean Square	Overall F	Prob. of Larger F
Model	11	9138.8	830.8	493.851	.0000
Error	204	343.2	1.7		

Corrected Total 215 9482.0

* * * Variation Due to the Model * * *

Source	DF	Sum of Squares	F	Prob. of Larger F
A	2	8345.96	2480.537	.0000
B	1	303.02	180.126	.0000
C	1	209.93	124.787	.0000
A*B	2	82.38	24.486	.0000
A*C	2	197.05	58.566	.0000
B*C	1	.13	.080	.7779
A*B*C	2	.32	.095	.9096

* * * Subgroup Means * * *

A Means (N=72)

1 1.4677

2 6.0466

3 16.3329

B Means (N=108)

1 6.7646

2 9.1335

C Means (N=108)

1 6.9632

2 8.9349

A*B Means (N=36)

1	1	.9991
---	---	-------

1	2	1.9363
---	---	--------

2	1	4.9377
---	---	--------

2	2	7.1556
---	---	--------

3	1	14.3572
---	---	---------

3	2	18.3086
---	---	---------

A*C Means (N=36)

1	1	1.4261
---	---	--------

1	2	1.5093
---	---	--------

2	1	5.4252
---	---	--------

2	2	6.6681
---	---	--------

3	1	14.0384
---	---	---------

3	2	18.6274
---	---	---------

B*C Means (N=54)

1	1	5.7539
---	---	--------

1	2	7.7754
---	---	--------

2	1	8.1726
---	---	--------

2	2	10.0944
---	---	---------

A*B*C Means (N=18)

1	1	1	.8782
---	---	---	-------

1	1	2	1.1199
---	---	---	--------

1	2	1	1.9740
---	---	---	--------

1	2	2	1.8987
---	---	---	--------

2	1	1	4.3181
---	---	---	--------

2	1	2	5.5573
---	---	---	--------

2	2	1	6.5322
2	2	2	7.7789
3	1	1	12.0653
3	1	2	16.6490
3	2	1	16.0116
3	2	2	20.6057

Appendix C

DOE for Setting SA Parameters w.r.t Relative Improvement

Design of Experiment for Simulated Annealing with

objective function = (initial solution cost -best cost)/initial cost,

A = number of jobs (n),

B = Number of initial trial solutions for initial temperature,

C = Alpha Cooling rate (Alpha),

Levels of A = (10 50 100),

Levels of B = (1*n 3*n),

Levels of C = (.90 .95 .99),

Dependent Variable	R-squared (percent)	Adjusted R-squared	Est. Std. Dev. of Model Error	Coefficient of Mean Var. (percent)
--------------------	---------------------	--------------------	-------------------------------	------------------------------------

Y	41.289	38.027	.07203	.2348	30.68
---	--------	--------	--------	-------	-------

*** Analysis of Variance ***

Source	DF	Sum of Squares	Mean Square	Overall F	Prob. of Larger F
Model	17	1.117	.06569	12.659	.0000
Error	306	1.588	.00519		
Corrected Total	323	2.705			

*** Variation Due to the Model ***

Source	DF	Sum of Squares	F	Prob. of Larger F
A	2	1.09669	105.674	.0000
B	1	.00124	.239	.6251
C	2	.00253	.243	.7841
A*B	2	.00177	.171	.8433
A*C	4	.01159	.558	.6930
B*C	2	.00125	.121	.8864
A*B*C	4	.00159	.077	.9893

*** Subgroup Means ***

A Means (N=108)

1	.3046
2	.2374
3	.1622

B Means (N=162)

1 .2328

2 .2367

C Means (N=108)

1 .2387

2 .2329

3 .2326

A*B Means (N=54)

1 1 .2995

1 2 .3098

2 1 .2379

2 2 .2369

3 1 .1610

3 2 .1634

A*C Means (N=36)

1 1 .3197

1 2 .2937

1 3 .3006

2 1 .2347

2 2 .2412

2 3 .2364

3 1 .1617

3 2 .1640

3 3 .1610

B*C Means (N=54)

1	1	.2373
1	2	.2330
2	1	.2280
2	2	.2401
3	1	.2328
3	2	.2372

A*B*C Means (N=18)

1	1	1	.3178
1	1	2	.2898
1	1	3	.2910
1	2	1	.3217
1	2	2	.2975
1	2	3	.3101
2	1	1	.2344
2	1	2	.2468
2	1	3	.2324
2	2	1	.2350
2	2	2	.2355
2	2	3	.2403
3	1	1	.1598
3	1	2	.1625
3	1	3	.1606
3	2	1	.1635
3	2	2	.1655
3	2	3	.1613

Appendix D

DOE for Setting TS Parameters w.r.t CPU Time

Design of Experiment for annealing with objective of cpu time) A = Number of
Jobs (n) B = initial temperature trial solution C = Alpha

Dependent Variable	R-squared (percent)	Adjusted R-squared	Est. Std. Dev. of Model Error	Mean	Coefficient of Var. (percent)
Y	58.985	56.707	5.688	6.408	88.75

* * * Analysis of Variance * * *

Source	DF	Sum of Squares	Mean Square	Overall F	Prob. of Larger F
Model	17	14236.1	837.4	25.887	.0000

Error	306	9898.9	32.3
Corrected Total	323	24135.0	

* * * Variation Due to the Model * * *

Source	DF	Sum of Squares	F	Prob. of Larger F
A	2	9801.71	151.498	.0000
B	1	51.09	1.579	.2098
C	2	135.04	2.087	.1258
A*B	2	104.77	1.619	.1997
A*C	4	633.19	4.893	.0008
B*C	2	1129.86	17.463	.0000
A*B*C	4	2380.44	18.396	.0000

* * * Subgroup Means * * *

A Means (N=108)

1 .7882

2 4.5613

3 13.8755

B Means (N=162)

1 6.0112

2 6.8054

C Means (N=108)

1 7.1859

2 6.4340

3		5.6051
A*B Means (N=54)		
1	1	.7419
1	2	.8344
2	1	4.6155
2	2	4.5071
3	1	12.6763
3	2	15.0747
A*C Means (N=36)		
1	1	.6714
1	2	.6984
1	3	.9946
2	1	4.4224
2	2	3.8049
2	3	5.4567
3	1	16.4638
3	2	14.7988
3	3	10.3640
B*C Means (N=54)		
1	1	4.3735
1	2	8.1696
2	1	5.4906
2	2	9.9982
3	1	4.6984
3	2	5.7196

A*B*C Means (N=18)			
1	1	1	.6142
1	1	2	.6624
1	1	3	.9491
1	2	1	.7287
1	2	2	.7344
1	2	3	1.0402
2	1	1	4.5402
2	1	2	3.6372
2	1	3	5.6690
2	2	1	4.3045
2	2	2	3.9726
2	2	3	5.2444
3	1	1	7.9661
3	1	2	20.2092
3	1	3	9.8538
3	2	1	24.9615
3	2	2	9.3884
3	2	3	10.8742

Appendix E

DOE for Setting TS Parameters w.r.t Relative Improvement

Objective value is Relative improvement over proposed heuristic HSTC

A = Candidate list size (60 100 150)

B = Tabu list size (7 11 20)

C = stopping criteria iteration (10000 20000)

Dependent Variable	R-squared (percent)	Adjusted R-squared	Est. Std. Dev. of Model Error	Coefficient of Mean	of Var. (percent)
Y	.426	.000	3.332	4.344	76.69

* * * Analysis of Variance * * *

Source	DF	Sum of Squares	Mean Square	Overall F	Prob. of Larger F
--------	----	----------------	-------------	-----------	-------------------

Model	17	16.2	.96	.086	1.0000
Error	342	3796.0	11.10		
Corrected Total	359	3812.3			

* * * Variation Due to the Model * * *

Source	DF	Sum of Squares	F	Prob. of Larger F
A	2	1.78717	.081	.9227
B	2	.14380	.006	.9935
C	1	2.27052	.205	.6514
A*B	4	3.65189	.082	.9878
A*C	2	2.59456	.117	.8897
B*C	2	2.25061	.101	.9036
A*B*C	4	3.54504	.080	.9885

* * * Subgroup Means * * *

A Means (N=120)

1 4.3435
2 4.2582
3 4.4307

B Means (N=120)

1 4.3506
2 4.3647
3 4.3171

C Means (N=180)

1 4.2647

2 4.4236

A*B Means (N=40)

1 1 4.4445

1 2 4.2253

1 3 4.3608

2 1 4.1975

2 2 4.4640

2 3 4.1130

3 1 4.4097

3 2 4.4050

3 3 4.4775

A*C Means (N=60)

1 1 4.3700

1 2 4.3170

1 3 4.0768

2 1 4.4395

2 2 4.3473

2 3 4.5142

B*C Means (N=60)

1 1 4.2327

1 2 4.4685

1 3 4.3955

2 1 4.3340

2 2 4.1660

2 3 4.4682

A*B*C Means (N=20)

1	1	1	4.3325
1	1	2	4.5565
1	2	1	4.4460
1	2	2	4.0045
1	3	1	4.3315
1	3	2	4.3900
2	1	1	4.0535
2	1	2	4.3415
2	2	1	4.4610
2	2	2	4.4670
2	3	1	3.7160
2	3	2	4.5100
3	1	1	4.3120
3	1	2	4.5075
3	2	1	4.2795
3	2	2	4.5305
3	3	1	4.4505
3	3	2	4.5045

Appendix F

Tables

t	TS		SA		BNB	
	ΣC	CPU Time sec	ΣC	CPU Time sec	ΣC	CPU Time sec
20	582.3	2.0	582.3	0.2	582.3	3
20	1016.2	1.3	1016.2	0.6	1016.2	148
30	1181.9	1.5	1181.9	1.2	1181.9	148
30	951.8	1.6	955.0	0.1	951.8	14
40	932.2	1.6	933.7	0.3	932.2	59
40	1149.1	2.0	1148.7	0.4	1148.7	242
50	714.3	1.2	715.6	0.4	656.7	11
50	1249.1	1.7	1257.2	0.9	1249.1	12

Table F.1: Comparison of TS, SA & BNB for $n = 9$, $T = 50$

	TS		SA		BNB	
t	ΣC	CPU Time sec	ΣC	CPU Time sec	ΣC	CPU Time sec
50	540.1	1.7	540.8	0.48	540.1	22
50	1560.5	1.4	1560.5	0.38	1560.5	185
50	435.4	1.6	435.4	0.12	435.4	13
50	918.4	1.8	918.4	1.15	918.4	2
50	572.8	1.5	572.8	0.72	572.8	11
50	1137.6	1.7	1137.6	1.40	1137.6	15
50	794.2	2.0	794.2	0.75	794.2	6
50	729.5	1.3	729.5	0.69	729.5	20

Table F.2: Comparison of TS, SA & BNB for $n = 9$, $T = 60$

	TS		SA		BNB	
t	ΣC	CPU Time sec	ΣC	CPU Time sec	ΣC	CPU Time sec
30	510.4	2.0	510.4	1.7	510.4	17
30	309.0	1.5	309.0	1.2	279.0	2
40	1027.3	2.1	1026.7	1.5	1026.7	94
40	614.8	1.2	614.8	1.0	614.8	13
50	1063.3	1.3	1063.2	0.8	1063.1	46
50	483.9	0.9	483.9	0.6	483.9	20
60	1167.2	1.8	1167.2	0.6	1167.2	80
60	634.1	2.2	634.1	1.5	634.1	21

Table F.3: Comparison of TS, SA & BNB for $n = 9$, $T = 70$

	TS		SA		BNB	
t	ΣC	CPU Time sec	ΣC	CPU Time sec	ΣC	CPU Time sec
40	931.5	1.7	931.5	0.8	931.5	6
40	586.5	1.1	586.5	1.2	586.5	17
50	619.1	1.7	619.1	0.6	619.1	18
50	930.1	1.7	930.1	1.5	930.1	25
60	922.7	1.3	922.7	0.9	922.7	27
60	521.0	1.3	521.0	1.8	521.0	23
70	1030.3	1.5	1030.3	1.8	1030.3	20
70	1299.6	1.6	1299.7	1.3	1299.6	153

Table F.4: Comparison of TS, SA & BNB for $n = 9$, $T = 80$

	TS		SA		BNB	
t	ΣC	CPU Time sec	ΣC	CPU Time sec	ΣC	CPU Time sec
50	796.4	1.3	796.4	1.1	796.4	15
50	376.6	0.9	376.6	1.8	376.6	21
60	576.7	1.4	576.7	1.7	576.7	21
60	1380.5	1.2	1380.5	1.4	1380.5	118
70	403.2	1.3	403.1	1.7	403.1	5
70	549.9	1.4	550.1	1.4	549.9	21
80	1089.3	1.3	1089.3	0.9	1089.3	40
80	690.5	1.5	690.5	0.8	690.5	24

Table F.5: Comparison of TS, SA & BNB for $n = 9$, $T = 90$

	TS		SA		BNB	
t	ΣC	CPU Time sec	ΣC	CPU Time sec	ΣC	CPU Time sec
10	1095.3	0.8	1093.2	0.7	1093.2	48
10	1350.9	1.3	1350.9	1.1	1350.5	44
20	940.7	3.1	940.7	1.4	940.7	98
20	1588.6	1.1	1588.6	0.7	1588.6	896
30	1448.5	2.0	1448.5	1.5	1448.5	402
30	1461.5	0.2	1461.5	1.1	1461.5	157
40	1874.9	1.6	1880.5	0.8	1874.9	896
40	1207.4	3.0	1207.4	1.3	1207.4	190

Table F.6: Comparison of TS, SA & BNB for $n = 10$, $T = 40$

	TS		SA		BNB	
t	ΣC	CPU Time sec	ΣC	CPU Time sec	ΣC	CPU Time sec
10	1000.4	0.3	1000.3	1.1	1000.3	8
10	1426.9	0.7	1424.5	0.9	1424.5	58
20	1130.4	1.2	1130.4	1.3	1130.3	67
20	2055.0	0.7	2055.0	1.2	2055.0	2560
30	2096.6	0.4	2095.8	1.5	2095.8	1641
30	2345.0	0.6	2345.0	1.7	2345.0	4437
40	2336.7	2.2	2335.8	1.0	2335.8	2387
40	2635.0	1.6	2635.0	1.6	2635.0	5777

Table F.7: Comparison of TS, SA & BNB for $n = 10$, $T = 50$

	TS		SA		BNB	
t	ΣC	CPU Time sec	ΣC	CPU Time sec	ΣC	CPU Time sec
10	1885.4	0.0	1885.4	0.5	1885.4	119
10	1658.5	0.6	1658.5	1.8	1658.5	44
20	1715.9	0.6	1713.3	0.6	1713.3	30
20	1856.6	0.4	1856.6	0.6	1856.6	350
30	2174.6	0.8	2174.7	1.6	2174.6	295
30	1841.8	1.9	1838.1	1.4	1838.1	113
40	2076.2	0.9	2073.3	0.4	2073.3	127
40	2613.2	1.1	2611.9	0.2	2611.9	881

Table F.8: Comparison of TS, SA & BNB for $n = 10$, $T = 60$

	TS		SA		BNB	
t	ΣC	CPU Time sec	ΣC	CPU Time sec	ΣC	CPU Time sec
10	1796.4	0.6	1790.5	1.7	1790.5	18
20	2048.9	1.7	2048.9	0.5	2048.9	28
20	2010.5	0.2	2010.5	0.7	2010.5	217
30	2104.6	0.5	2103.9	1.3	2103.9	207
30	2230.5	2.3	2230.5	0.7	2230.5	547
40	1279.5	1.6	1279.1	1.0	1279.1	36
40	1500.1	0.7	1503.0	0.3	1500.1	180

Table F.9: Comparison of TS, SA & BNB for $n = 10$, $T = 70$

	TS		SA		BNB	
t	ΣC	CPU Time sec	ΣC	CPU Time sec	ΣC	CPU Time sec
10	2228.5	1.0	2226.7	1.2	2226.7	118
10	2128.2	1.3	2128.2	0.9	2128.2	17
20	2648.1	1.0	2648.1	1.1	2648.1	103
20	2924.4	2.2	2924.4	0.8	2924.4	816
30	2358.6	0.0	2374.7	0.9	2354.4	173
30	1759.6	1.0	1759.5	0.6	1759.5	67
40	3504.4	0.5	3504.4	1.2	3214.4	2158
40	2948.5	1.2	2946.7	1.5	2946.7	3296

Table F.10: Comparison of TS, SA & BNB for $n = 10$, $T = 80$

	TS		SA		BNB	
t	ΣC	CPU Time sec	ΣC	CPU Time sec	ΣC	CPU Time sec
10	2413.7	1.5	2413.1	1.1	2411.8	1543
10	2388.2	0.8	2388.7	1.3	2386.6	8
20	2102.4	1.5	2098.2	0.6	2098.2	10
20	2651.8	0.4	2651.8	0.8	2651.8	15780
30	2872.7	0.1	2874.9	1.5	2826.6	240
30	2451.6	1.2	2451.6	1.4	2451.6	22
40	3084.9	1.0	3084.8	1.0	3080.8	22805
40	1772.4	0.6	1772.9	1.2	1761.9	35

Table F.11: Comparison of TS, SA & BNB for $n = 10$, $T = 90$

	TS		SA		BNB	
t	$\sum C$	CPU Time sec	$\sum C$	CPU Time sec	$\sum C$	CPU Time sec
10	933.2	1.6	934.6	1.7	933.2	13
	714.5	2.0	712.2	2.0	712.2	14
	647.8	1.7	650.0	2.0	647.5	76
	1033.8	0.8	1037.4	1.7	1032.4	5002
	804.0	0.8	811.2	1.2	804.0	9
	1070.1	1.4	1075.6	1.2	1070.1	500
	1562.3	0.7	1568.8	0.8	1562.3	21539
	939.5	1.6	943.6	1.5	939.5	207

Table F.12: Comparison of TS, SA & BNB for $n = 11$, $T = 40$

	TS		SA		BNB	
t	$\sum C$	CPU Time sec	$\sum C$	CPU Time sec	$\sum C$	CPU Time sec
20	795.4	2.3	795.4	2.0	795.4	512
	757.9	0.9	757.9	0.8	757.9	251
	1033.3	0.8	1027.4	1.5	1026.8	2760
	1562.3	2.1	1568.8	2.2	1562.3	21539
	939.5	2.1	943.6	2.4	939.5	207

Table F.13: Comparison of TS, SA & BNB for $n = 11$, $T = 40$

	TS		SA		BNB	
t	ΣC	CPU Time sec	ΣC	CPU Time sec	ΣC	CPU Time sec
30	1391.9	0.7	1387.3	2.3	1387.3	10912
	1212.1	0.7	1212.1	0.9	1212.1	3623
	1330.5	1.1	1330.5	1.7	1330.5	19787
	1273.0	1.4	1279.2	1.3	1273.0	384
	959.7	1.6	961.8	1.5	959.7	1776

Table F.14: Comparison of TS, SA & BNB for $n = 11$, $T = 40$

	TS		SA		BNB	
t	ΣC	CPU Time sec	ΣC	CPU Time sec	ΣC	CPU Time sec
40	2039.0	1.7	2049.6	2.4	2038.8	50518
	917.2	1.5	916.3	1.8	916.3	185
	1243.8	1.1	1239.8	2.0	1239.8	1381
	1797.9	2.3	1797.9	2.8	1797.9	139244
	1181.4	2.3	1178.4	2.2	1178.4	20011

Table F.15: Comparison of TS, SA & BNB for $n = 11$, $T = 40$

	TS		SA		BNB	
t	ΣC	CPU Time sec	ΣC	CPU Time sec	ΣC	CPU Time sec
20	1503.0	1.0	1502.4	1.0	1502.4	2662
	936.6	1.7	936.6	1.5	936.6	20
	1196.2	2.3	1196.2	1.4	1196.2	507
	912.0	2.8	912.0	1.5	912.0	25
	1342.5	1.3	1342.5	1.2	1342.5	2014

Table F.16: Comparison of TS, SA & BNB for $n = 11$, $T = 50$

	TS		SA		BNB	
t	ΣC	CPU Time sec	ΣC	CPU Time sec	ΣC	CPU Time sec
30	1173.9	1.3	1173.9	1.6	1173.9	3049
	932.8	2.7	932.8	2.1	932.8	68
	1256.9	1.2	1258.1	1.0	1256.9	4595
	1130.6	2.6	1130.6	0.8	1130.6	2349
	1268.1	2.8	1268.1	1.8	1268.1	7332

Table F.17: Comparison of TS, SA & BNB for $n = 11$, $T = 50$

	TS		SA		BNB	
t	ΣC	CPU Time sec	ΣC	CPU Time sec	ΣC	CPU Time sec
20	1503.0	1.0	1502.4	1.0	1502.4	2662
	936.6	1.7	936.6	1.5	936.6	20
	1196.2	2.3	1196.2	1.4	1196.2	507
	912.0	2.8	912.0	1.5	912.0	25
	1342.5	1.3	1342.5	1.2	1342.5	2014

Table F.18: Comparison of TS, SA & BNB for $n = 11$, $T = 50$

Bibliography

- [1] M. Sait Sadiq and Habib Youssef. *Iterative Computer Algorithm with Applications in Engineering*. IEEE Computer society, 1999.
- [2] S. French. *Sequencing and Scheduling: An introduction to the mathematics of job-shop*. Jhon wiley and sons, 1981.
- [3] M. Pinedo. *Scheduling: Theory, algorithms, and systems*. Prentice Hall, Englewood Cliffs NJ., 1995.
- [4] X. Qi, T. Chen, and F. Tu. Scheduling the maintenance on a single machine. *Journal of the Operational Research Society*, 50:1071–1078, 1999.
- [5] A. C. Yao. New algorithms in bin packing. *J ACM*, 27:207–227, 1980.
- [6] G. Schmidt. Scheduling independent tasks with deadlines on semi-identical processors. *Journal of Operational Research Society*, 39:271–277, 1988.
- [7] R. Kenneth and G. D. Scudder. Sequencing with earliness and tardiness penalties: A review. *Operations Research*, 38:22–36, 1990.

- [8] I. Adiri. Single machine flow-time scheduling with single breakdown. *ACTA Informatica*, 26:679–696, 1992.
- [9] C. Y. Lee and S. D. Liman. Single machine flow-time scheduling with scheduled maintenance. *ACTA Informatica*, 29:375–382, 1992.
- [10] R. M. Narashima and J. B. Addagatla. Heuristic algorithms for solving Earliness-Tardiness scheduling problem with machine vacations. *Computers and Industrial Engineering*, 25,1-4:255–258, 1993.
- [11] C. Y. Lee. Machine scheduling with an availability constraint. *Journal of Global Optimization*, 9:395–416, 1996.
- [12] Renata Mazzini and V. A. Amentano. A heuristic for single machine scheduling with early and tardy costs. *European Journal of Operational Research*, 29:129–146, 2001.
- [13] C. Y. Lee and Zhi-Long Chen. Scheduling jobs and maintenance activities on parallel machine. *Naval Research Logistics*, 47:145–165, 2000.
- [14] J. Wesley Barnes and B. Chambers John. Solving the job shop scheduling problem with tabu search. *IIE Transactions*, 27:257–263, 1995.
- [15] M. Barglia and R. Melloni. Tabu search for single machine sequencing problem with ready time. *IEEE Symposium on Emerging Technologies*, 80(2):397–403, 1995.

- [16] Colin Reeves. Heuristics for scheduling a single machine subject to unequal job release times. *European Journal of Operational Research*, 80(2):397–403, 1995.
- [17] C. Tsallis and D. A. Stariolo. Generalized simulated annealing. *Statistical and Theoretical Physics*, 233(1-2):395–406, 1996.
- [18] M. Ben-Daya and M.A. Al-Fawzan. A simulated annealing approach for the one-machine mean tardiness scheduling problem. *European Journal of Operational Research*, 93:61–67, 1996.
- [19] M.A. Al-Fawzan and K.S. Al-Sultan. Tabu search algorithm for minimizing the makespan in a job shop scheduling. *Industrial Engineering Research- Conference Proceeding*, pages 115–119, 1996.
- [20] J. Lyu, A. Guneskaren, and J. H. Ding. Simulated annealing algorithm for solving the single machine early/tardy problem. *International Journal of Systems Science*, 27(7):605–610, 1996.
- [21] Huang Xiaoyuan and Liu Hailong. Multiobjective scheduling on a single machine. *Proceedings of SPIE- The International Society of optical society*, 2620, 1995.
- [22] Ruiz-Torres, J. Alex, and E. Emory Enscore. Simulated annealing heuristics for average flow-time and the number of tardy jobs bi-criteria parallel machine problem. *Computers and Industrial Engineering*, 33(1-2):257–260, 1997.

- [23] Marangos, A. Charalambos, and Govanda Vinay. Algorithms to minimize completion time variance in a two machine flowshop. *Computers and Industrial Engineering*, 35(1-2):101–104, 1998.
- [24] A. Armento Vinicius and P. Ronconi Debora. Tabu search for total tardiness minimization in flow shop problems. *Computers and Operations Research*, 26(3):219–235, 1999.
- [25] A. Islam and M. Eksioğlu. Tabu search approach for single machine mean tardiness problem. *Journal of Operational Research Society*, 48(7):751–755, 1997.
- [26] Y. Yi and D. Wang. Tabu search for scheduling grouped jobs on parallel machines. *journal of Northeastern University, Shenyang, China*, 22(2):188–191, 2001.
- [27] John Kanet. Minimizing the average deviation of the job completion times about a common due date. *Naval Research Logistics*, 28(4):643–651, 1981.
- [28] John Kanet. Minimizing variation of flow time in single machine systems. *Management Science*, 27:1453–1459, 1981.
- [29] P. S. Sundararaghvan and U. Ahmed Mesbah. Minimizing the sum of absolute lateness in single machine and multi-machine scheduling. *Naval Research Logistics*, 31:325–333, 1984.

- [30] Wlodzimierz Szwarc. Single-machine scheduling to minimize absolute deviation of completion times from a common due date. *Naval Research Logistics*, 36:663–673, 1989.
- [31] M. C. Gupta, Y. P. Gupta, and C. R. Bector. Minimizing the flow-time variance in single machine systems. *Journal of Operational Research Society*, 41(8):767–779, 1990.
- [32] V. Jorge leon and S. David Wu. On scheduling with ready-times, due dates and vacations. *Naval Research Logistics*, 39:53–65, 1992.
- [33] Fred Glover. Tabu search and adaptive memory programming- advances, applications and challenges. Technical Report, 1996.
- [34] Glover and Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.
- [35] Fred Glover. Tabu search; a tutorial. technical report. Technical report, University of Colorado, Boulder, 1990.
- [36] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Management Science*, 220(4598):671–680, 1983.
- [37] R. W. Eglese. Simulated annealing: a tool for operational research. *European Journal of Operational Research*, 46:271–281, 1990.

- [38] Junaid Asim Khan. Performance driven, low power standard VLSI cell placement using simulated evolution. Master's thesis, King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia, December 2000.

Vitae

- **SYED ASIF RAZA**

- Born in Karachi, Pakistan.
- Received Bachelor of Engineering Degree in Mechanical Engineering from N.E.D University of Engineering and Technology, Karachi, Pakistan, in Dec 1998.
- Joined Systems Engineering Department, KFUPM, as a Research Assistant in September 1999.
- Received Master of Science Degree in Systems Engineering from KFUPM, Dhahran, Saudi Arabia in May 2002.