NEW SOLUTION METHODS FOR SINGLE MACHINE BICRITERIA SCHEDULING PROBLEM: MINIMIZATION OF AVERAGE FLOWTIME AND NUMBER OF TARDY JOBS

A THESIS
SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Fatih Safa Erenay
July, 2006

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. İhsan Sabuncuoğlu (Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Ayşegül Toptal

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Erdal Erel

Approved for the Institute of Engineering and Science:

Prof. Mehmet Baray
Director of Institute of Engineering and Science

Abstract

NEW SOLUTION METHODS FOR SINGLE MACHINE BICRITERIA SCHEDULING PROBLEM: MINIMIZATION OF AVERAGE FLOWTIME AND NUMBER OF TARDY JOBS

Fatih Safa Erenay

M.S. in Industrial Engineering

Supervisor: Prof. İhsan Sabuncuoğlu

July 2006

In this thesis, we consider the bicriteria scheduling problem of minimizing number of tardy jobs and average flowtime on a single machine. This problem, which is known to be NP-hard, is important in practice as the former criterion conveys the customer's position and the latter reflects the manufacturer's perspective in the supply chain. We propose two new heuristics to solve this multiobjective scheduling problem. These two heuristics are constructive algorithms which are based on beam search methodology. We compare these proposed algorithms with three existing heuristics in the literature and two new meta-heuristics. Our computational experiments illustrate that proposed heuristics find efficient schedules optimally in most of the cases and perform better than the other heuristics.

Keywords: Bicriteria Scheduling, Average Flowtime, Number of Tardy Jobs, Beam Search.

Özet

TEK MAKİNEDA İKİ ÖLÇÜTLÜ ÇİZELGELEME PROBLEMİ İÇİN YENİ ÇÖZÜM METODLARI: ORTALAMA AKIŞ SÜRESİ VE TOPLAM GEÇ KALMIŞ İŞ SAYISINI ENKÜÇÜKLEME

Fatih Safa Erenay

Endüstri Mühendisliği Yüksek Lisans

Tez Yöneticisi: Prof. İhsan Sabuncuoğlu

Temmuz 2006

Bu tezde, ortalama iş akış süresini ve toplam geç kalmış iş sayısını enküçüklemeyi hedefleyen iki ölçütlü tek makina çizelgeleme problemini ele aldık. NP-zor olduğu bilinen bu problemın önemi ele aldığı ölçütlerden kaynaklanmaktadır. Zira, ele alınan birinci ölçüt tedarik zinciri içerisindeki bir üreticinin, ikincisi ise bir tüketicinin bakış açısını temsil eder. Bu çok ölçütlü problem için iki yapıcı sezgisel yöntem öneriyoruz. Bu iki yöntem ışın taraması algoritması esas alınarak geliştirilmiştir. Önerilen bu iki algoritma, üçü literatürde mevcut ikisi de yeni geliştirilmiş olan, 5 farklı sezgisel yöntem ile karşılaştırılmıştır. Yaptığımız sayısal testler sonucu, önerdiğimiz algoritmaların, çoğu zaman en iyi etkin çizelgelere ulaştığı ve karşılaştırıldıkları sezgisel yöntemlerden daha iyi sonuçlar verdikleri tesbit edilmiştir.

Anahtar Kelimeler: İki Ölçütlü Çizelgeleme, Ortalama İş Akış Süresi, Toplam Geç Kalmış İş Sayısı, Işın Taraması.

To my family,

Acknowledgement

I would like to express my sincere gratitude to Prof. İhsan Sabuncuoğlu and Asst. Prof. Ayşegül Toptal for their instructive comments and encouragements in this thesis work. I believe that their valuable suggestions in the supervision of the thesis will guide me throughout all my academic life.

I am indebted to Prof. Erdal Erel for accepting to review this thesis, and his useful comments and suggestions.

I would like to express my special thanks to Süleyman Kardaş and Mustafa Aydoğdu for their computational aids, sharing their technical knowledge with me and for their friendship.

I am also indebted to Prof. Selim Aktürk, Assoc. Prof. Oya Ekin Karaşan, Asst. Prof. Emre Alper Yıldırım, and Asst. Prof. Mehmet Rüştü Taner for their valuable time and feedbacks.

I would also like to thank to Selçuk Gören, Hakan Gültekin, Mehmet Mustafa Tanrıkulu, Muzaffer Mısırcı, Fazıl Paç, Gülay Samatlı, Ahmet Camcı, Cağdaş Büyükkaramıklı, Çagrı Latifoğlu, Sinan Gürel, Ayşegül Altın, Sıtkı Gülten and my other friends for their helps and morale support during my graduate study.

Finally, I would like to express my deepest gratitude to my family for their understanding and patience during my graduate life.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

In the literature most scheduling studies consider optimization of a single objective function. However, in practice, decision makers evaluate schedules according to more than one measure. Since using multiple criteria is more realistic, several multicriteria scheduling papers have appeared in the scheduling literature. Most of these papers are on single machine bicriteria scheduling problems. In the vein of this literature, this thesis study considers minimization of mean flowtime ($\overline{F}$) and number of tardy jobs (n_T) on a single machine. Our contribution lies in developing new heuristics that outperform the current approximate solution methodologies. Also, we characterize the effectiveness of these proposed heuristics in terms of problem parameters.

We propose two heuristics, which are constructive algorithms based on beam search method. In addition, two heuristics iteratively utilizing genetic algorithm and tabu-search are developed by Kardas and Sabuncuoglu (2006) and Aydogdu and Sabuncuoglu (2006). These new heuristics are designed to find the approximately efficient schedules. That is, they can estimate the pareto frontier solutions for the problem of minimizing mean flowtime ($\overline{F}$) and number of tardy jobs (n_T) on a single machine.

Efficient schedules are the set of schedules that cannot be dominated by any other feasible schedule according to the considered criteria. All other schedules, which are not in this set, are dominated by at least one of these efficient schedules. The reason for seeking efficient schedules instead of minimizing weighted sum of

n_T and $\overline{F}$ is that whatever the weights are, the optimum solution will be one of the efficient schedules. Specifically, given the efficient schedules for the bicriteria problem and the corresponding weights w_1 and w_2, the solution to the minimization of $w_1 \overline{F} + w_2 n_T$ can be found by evaluating all these finite number of efficient schedules.

Number of tardy jobs and average flowtime are quite significant criteria for characterizing the behavior of a manufacturer who wants to meet the due dates of his/her customers while minimizing own inventory holding costs. The solution to the single machine problem which is known to be NP-hard (Bulfin and Chen, 1993) can be used as an aggregate schedule for the manufacturer, or for generating a more detailed schedule for a factory based on a bottleneck resource. Thus, having an effective approximate solution methodology for finding efficient schedules to this problem is important both theoretically and in practical sense.

The organization of this thesis is as follows. In Section 2, we present a literature review on multicriteria scheduling. In Section 3, we formulate the problem of minimizing number of tardy jobs and average flowtime on a single machine. In Section 4, we describe Nelson et al. (1986)'s optimum solution method for this problem. The proposed beam search algorithms are presented in Section 5. Computational results are provided in Section 6. Finally, concluding remarks and future research directions are given in Section 7.

Chapter 2

LITERATURE REVIEW

In the scheduling literature, most of the studies consider bicriteria single machine scheduling problems that minimize couples of criteria such as maximum tardiness and flowtime (Smith, 1956; Heck and Robert, 1972; Sen and Gupta, 1983; Koksalan, 1999), maximum earliness and flowtime (Koksalan et al., 1998; Koktener and Koksalan, 2000; Keha and Koksalan, 2003), maximum earliness and number of tardy jobs (Erol et al., 1998; Kondakci et al., 2003). Extensive surveys of several bicriteria single machine scheduling studies are provided by Dileepan and Sen (1988), Fry et al. (1989) and Wan and Yen (2003). In addition to these survey papers, Nagar et al. (1995), Billaut and T'kindt (1999) and Hoogeveen (2005) review multicriteria scheduling literature including those papers that consider more than two criteria and more complex settings.

Bulfin and Chen (1993) analyze the complexity of the single machine multicriteria scheduling problems which consider maximum tardiness, flowtime, number of tardy jobs, tardiness and the weighted counterparts of the last three criteria. A more recent publication that reviews the complexity of the multicriteria scheduling problem is by T'kindt et al. (2005). The paper is mainly about the enumeration complexity theory. Nevertheless, the survey also reviews the complexity of several multicriteria scheduling problems as an application of the theorems presented in the paper.

Multicriteria scheduling studies can be grouped into three categories as: hierarchical optimization, weighted sum optimization and pareto optimization

(Wan and Yen, 2003). Hierarchical optimization approach tries to minimize some of the criteria while keeping the others at their optimal value. In weighted sum optimization approach, the decision makers assign weights to the criteria. Thus, the multiple criteria are reduced to a single performance measure. The last category, pareto optimization, minimizes corresponding criteria simultaneously by finding efficient schedules. The current study belongs to the last category.

For single machine case, the problem of minimizing n_T, while $\overline{F}$ is optimum, is solved in polynomial time (Chen and Bulfin, 1993) by an adjusted version of SPT order which applies Moore's Algorithm to break ties among the jobs with equal processing time. In the rest of the thesis, *SPT order* will refer to this adjusted version. In another study, Emmons (1975) develops an algorithm for minimizing $\overline{F}$ while n_T is optimum. Later, this problem is showed to be NP-Hard by Huo et al. (2005). Finally, Chen and Bulfin (1993) prove that simultaneously minimizing both criteria on a single machine via finding efficient schedules is NP-Hard.

Then, Nelson et al. (1986) develop a branch and bound procedure to find efficient schedules for minimizing n_T and $\overline{F}$ optimally on single machine. In addition, Nelson et al. (1986) develop a constructive heuristic for this problem. In another study, Kiran and Unal (1991) define several theorems about the characteristics of the efficient solutions. Kondakci and Bekiroglu (1997) present some dominancy rules on the efficient solutions, which they use to develop more effective optimal solution method. These dominancy rules are applied to the Nelson et al.'s branch and bound procedure. Consequently, the paper reports that the size of branch and bound tree is reduced considerably.

Recent studies on the problem propose some general purpose procedures. Koktener and Koksalan (2000), and Keha and Koksalan (2003) develop heuristic methods based on simulated annealing and genetic algorithms, respectively. The later study indicates that genetic algorithm generally performs better than the

simulating annealing; however, simulating annealing approach is faster than the genetic algorithm.

After reviewing these studies we observe that there are many multicriteria scheduling papers in the literature. However, only a few solution methodologies, (one exact and three heuristics) are proposed for the problem that the current study considers. Moreover, these solution methods are not compared with each other in detail. Thus their relative strengths are unknown. Only simulated annealing (Koktener and Koksalan, 2000) and genetic algorithm (Keha and Koksalan, 2003) approaches are compared with each other. Nevertheless, these two iterative methods are not properly compared with the optimum solution for problems with more than 20 jobs. Therefore, the current study presents two constructive and two iterative heuristic methods for this problem and compares these proposed heuristics with each other as well as with the other exact and heuristic solution methods available in the literature. Hence, the current study will illustrate the relative strengths of each solution method.

Chapter 3

PROBLEM FORMULATION

As discussed earlier, our approach aims at finding approximately efficient schedules for minimizing $\overline{F}$ and n_T. More formally, we are interested in finding a set of schedules where, if S is an element of this set, then there exists no schedule S' such that;

i) $n_T(S') \leq n_T(S)$

ii) $\overline{F}(S') \leq \overline{F}(S)$

iii) At least one of these constraints is strict.

Furthermore, our approach builds on the fact that optimizing either one of the objectives, n_T or $\overline{F}$, on a single machine is polynomially solvable. It is well known in the scheduling literature that shortest processing time (SPT) rule minimizes the average flowtime, and Moore's Algorithm (Moore, 1968) minimizes the number of tardy jobs. In the rest of the thesis, we will denote n_T(SPT) and n_T(Moore) as the number of tardy jobs in the sequence formed for a problem instance using SPT rule and Moore's Algorithm, respectively.

We assume that the processing times and due dates are constant and known at the beginning of the planning horizon. We also assume that there is no preemption or precedence relation between jobs. The delays that occur in machining process due to maintenance and unexpected failures are ignored. We define N as the total number of jobs and refer to a particular job by index j. P_j and d_j denote the

processing time and the due date of Job *j,* respectively. In single machine setting, a schedule is the sequence in which the jobs will start to be processed. Denoting *S* as a feasible schedule, $\overline{F}(S)$ represents the average flowtime of schedule *S* and $n_T(S)$ refers to the number of tardy jobs resulting from schedule *S***.**

Kiran and Unal (1991) show that for each number of tardy jobs between n_T(SPT) and n_T(Moore), there exists at least one corresponding efficient schedule. Therefore, the range between n_T(SPT) and n_T(Moore) is referred to as *efficient range* of number of tardy jobs. Since there exists at least one efficient schedule for every n_T value in this range, total number of efficient schedules for a given problem is at least n_T(SPT) - n_T(Moore) + 1. Therefore, for a problem with N jobs, we solve the following model for all n in the efficient range.

$$\underset{\forall S}{Min} \ \ \overline{F}(S)$$

st

$n_T(S) = n$ *where* $n_T(\text{SPT}) \geq n \geq n_T(\text{Moore})$

For the purpose of presenting a more detailed formulation of the above problem, let us define X_{ij} and Y_j as follows.

$$X_{ij} = \begin{cases} 1, & \text{if } i^{\text{th}} \text{ position is held by Job } j \\ 0, & \text{o.w.} \end{cases}$$

$$Y_j = \begin{cases} 1, & \text{if Job } j \text{ is tardy} \\ 0, & \text{o.w.} \end{cases}$$

Also, let M and ξ denote a very large and a very small number, respectively. The mathematical model is given below.

$$Min \ \frac{1}{N}\left(\sum_{i=1}^{N}\ \sum_{j=1}^{N}(N-i+1)X_{ij}P_j\right)$$

s.t.

$$\sum_{j=1}^{N} X_{ij} = 1 \ \text{ for all } i \in \{1, 2, \ldots\ldots \text{N}\} \tag{1}$$

$$\sum_{i=1}^{N} X_{ij} = 1 \ \text{ for all } j \in \{1, 2, \ldots\ldots \text{N}\} \tag{2}$$

$$d_j - P_j - \sum_{r=2}^{N}\sum_{i=1}^{r-1}\sum_{k=1}^{N} X_{rj}X_{ik}P_k \geq -M \times Y_j \ \text{ for all } j \in \{1, 2, \ldots\ldots \text{N}\} \tag{3}$$

$$d_j - P_j - \sum_{r=2}^{N}\sum_{i=1}^{r-1}\sum_{k=1}^{N} X_{rj}X_{ik}P_k \leq M \times \left(1 - Y_j\right) - \xi \ \text{ for all } j \in \{1, 2, \ldots\ldots \text{N}\} \tag{4}$$

$$\sum_{j=1}^{N} Y_j = n \tag{5}$$

$i, j, k, r \in \{1,....N\}$;

Equation (1) assures that only one job can be assigned on each position in the schedule. Equation (2) makes sure that there is no unassigned job. Expressions (3) and (4) jointly identify whether Job j is tardy or not, i.e. $Y_j = 0$ or $Y_j = 1$. Finally, Equation (5) assures that only n jobs are tardy. In order to solve the problem of minimizing n_T and $\overline{F}$ on a single machine, this mathematical model should be solved for every n s.t. n_T(SPT) $\geq n \geq$ n_T(Moore). As seen in the model, Inequalities (3) and (4) are nonlinear due to the multiplication of X_{rj} and X_{ik}. However, since the both variables are binary, it is possible to linearise these inequalities by replacing X_{rj} X_{ik} with Z_{rjik} and adding the following expressions to the model.

i) $X_{rj} \geq Z_{rjik}$

ii) $X_{ik} \geq Z_{rjik}$

iii) $Z_{rjik} \geq X_{rj} + X_{ik} - 1$

for all $i, j, k, r \in \{1,....N\}$;

In a given problem, the efficient schedule that has n_T(SPT) tardy jobs is the schedule that is formed according to SPT order. For a given problem, other n_T(SPT) - n_T(Moore) efficient schedules need to be found. Nelson et al. (1984) proposed an efficient branch and bound algorithm to find all these schedules optimally. However, this algorithm works well only for small sized problems. Since the computationally efficient heuristics that we propose will use some insights from and will be compared with the optimum solution, let us present a brief summary of this algorithm in the next chapter.

Chapter 4

OPTIMAL SOLUTION METHOD FOR MINIMIZING n_T AND $\overline{F}$

In this section, we present a summary of the branch and bound method proposed by Nelson et al. (1986). This method finds an efficient schedule for each n in the efficient range for the problem of minimizing n_T and $\overline{F}$ on a single machine. Basically, it depends on two key points. The first one is the fact that, given N jobs and a subset of these N jobs, the schedule that gives minimum $\overline{F}$ while keeping the jobs in the given subset non-tardy is found using Smith's Algorithm (Smith, 1956; Kiran and Unal, 1991). The second one is presented in the following theorem.

Theorem 1: The jobs that are early in the SPT order are also early in at least in one of the efficient schedules with $n_T = n$ for all n s.t. $n_T(\text{SPT}) \geq n \geq n_T(\text{Moore})$ (Nelson et al., 1986).

This theorem indicates that, in order to find an efficient schedule with $N_T = n$, it is necessary to determine which other $n_T(\text{SPT}) - n$ jobs will be early besides the early jobs of SPT order. Therefore all subsets of SPT order's tardy jobs with cardinality $n_T(\text{SPT}) - n$ should be evaluated by using Smith's Algorithm to find the schedule with minimum F while having n tardy jobs. The schedule that is obtained through this evaluation is the efficient schedule for $n_T = n$.

The branch and bound method (B&B) is designed to determine one efficient schedule in every level of the branch and bound tree by finding which $n_T(\text{SPT}) - n$

jobs should be early. In the first level, the efficient schedule for $n_T = n_T(\text{SPT})$ is found and in the k^{th} level efficient schedule for $n_T = n_T(\text{SPT}) - k + 1$ is found. The tree continues in this manner such that at the lowest level an efficient schedule for $n_T = n_T(\text{Moore})$ is found. In this tree, each node stores the set of jobs that need to be kept nontardy. We refer to this set as *set of early jobs* in the remaining parts of the thesis. A set of early job at level k is a subset of N jobs with cardinality $N - n_T(\text{SPT}) + k - 1$. The nodes in level k cover all of the possible subsets with the specified cardinality. $N - n_T(\text{SPT})$ of these jobs in each set of early jobs are the early jobs of the SPT order and the remaining $k - 1$ are among the tardy jobs of the SPT order. For each node in level k, Smith's Algorithm is run, and the schedule that has the minimum F while keeping corresponding $N - n_T(\text{SPT}) + k - 1$ jobs non-tardy is found. The schedule that gives the least $\overline{F}$ in level k is the efficient schedule for $n_T = n_T(\text{SPT}) - k + 1$. This procedure is repeated for each level of the branch and bound tree. A sample question is presented in Appendix-A to show how Nelson et al.'s (1986) branch and bound tree is built.

Each node in level k of the branch and bound tree represents a set of early jobs. As stated above, we use Smith's Algorithm to evaluate the nodes of Nelson et al.'s B&B tree. Indeed, Smith's Algorithm minimizes $\overline{F}$ given that T_{max} is zero where T_{max} is the maximum tardiness. Equivalently, this algorithm finds the schedule that minimizes $\overline{F}$ given that $n_T = 0$. This implies that, in finding the minimum $\overline{F}$ corresponding to a node in the B&B tree, first, the due date of the jobs that are not in the set of early jobs are set to infinity, and then Smith's Algorithm are applied. Therefore, for each node k, we solve following problem by using Smith's Algorithm.

$$\underset{\forall S}{Min} \ \overline{F}(S)$$

st

$T_{max}(S) = 0;$

$d_j = \infty \ \ \forall j \notin E_k$ where E_k is the set of early jobs of node k.

The steps of Smith's Algorithm are described in the following pseudo algorithm. In this pseudo algorithm E is the set of jobs that are not scheduled yet and P_T is the sum of processing times of the unscheduled jobs. Moreover, k denotes the position of the sequence to which a job will be assigned by the algorithm.

Step 0: $E = \{1,2,3,\ldots\ldots,N\}$, $P_T = \sum_{j=1}^{N} p_j$, $k = N$.

Step 1: Record all the jobs j where $j \in E$ and $d_j \geq P_T$.

Step 2: Among the recorded jobs choose the one with the largest processing time. Assign that job to the k^{th} position in the schedule and record the processing time of the job to the variable P.

Step 3: Remove the assigned job from E. $P_T = P_T - P$, $k = k - 1$.

Step 4: If $E = \phi$, go to Step 5. Otherwise go to Step 1.

Step 5: The schedule is completed. Report the $\overline{F}$ value of the completed schedule.

Step 6: Terminate the algorithm.

The time that Smith's Algorithm requires to evaluate a node is increasing polynomially with respect to the number of the jobs to be scheduled. However, the number of the nodes that are needed to be evaluated increases exponentially as the number of the jobs increases. Therefore, Nelson et al.'s B&B Algorithm requires quite high CPU time to solve problems with more than 60 jobs.

Chapter 5

PROPOSED BEAM SEARCH ALGORITHMS AND OTHER NEW HEURISTICS

Since minimizing average flowtime and number of tardy jobs on a single machine is an NP-Hard problem, we develop two beam search based heuristic algorithms to find the approximately efficient schedules. Beam search is successfully applied to a variety of scheduling problems such as FMS scheduling (Sabuncuoglu and Karabuk, 1998), job-shop scheduling (Sabuncuoglu and Bayiz, 1999; Duarte et al., 2004), open shop scheduling (Blum, 2005), mixed-model assembly line scheduling (McMullen and Tarasewich, 2005), unrelated parallel machine scheduling (Ghirardi and Potts, 2005).

Beam search is a fast and approximate branch and bound algorithm. Instead of expanding every node to the next level in the classical branch and bound tree, beam search expands only a limited number of promising nodes to the next levels. Thus, rather than making all exhausting branch and bound tree operations, beam search efficiently operates only on a small portion of the tree and gets a quick and approximate solution.

Generally, at a level of beam search tree, the nodes are evaluated via a global evaluation function. The nodes with the highest scores are selected to be expanded to the next level. The number of these nodes is fixed and called *beam width* (*b*) in

the literature. In some beam search applications, a portion of the nodes to be expanded to the next level is chosen randomly in order to increase the quality of the solution. Some of the beam search algorithms use local evaluation functions to eliminate some of the nodes before evaluating them with global evaluation function. This approach is called as *filtered beam search*. In fact, in the literature, there are a number of other enhanced beam search algorithms.

In the literature, there are two types of beam search implementation with respect to the branching procedure; dependent and independent beam search. We applied both of these branching procedures to the problem of minimizing N_T and $\overline{F}$ on a single machine.

5.1 Independent Beam Search (BS-I)

As stated before, beam search is a quick and approximate branch and bound algorithm. It operates on a small portion of the Nelson et al.'s (1986) search tree in order to obtain a good solution quickly.

The first two levels of our beam search tree are the same as Nelson et al.'s search tree (see Figure 2 in Appendix-A and Figure 3 in Appendix-B). However, at level 2, only *b* number of the nodes are expanded to the next level. These *b* nodes are the ones with the *b* smallest $\overline{F}$ values obtained from applying Smith's Algorithm to the corresponding nodes. At the next levels, only one node among the nodes that are expanded from the same parent can be expanded to the next level. The schedule that is given by the node with minimum $\overline{F}$ among all the nodes at a level is chosen as the approximately efficient schedule for the corresponding level. The global evaluation function of BS-I is the average flowtime obtained by running Smith's Algorithm for the corresponding node. The example presented in Appendix-B shows how the proposed algorithm finds efficient schedules for each $n_T = n$ where n_T(SPT) $\geq n \geq$ n_T(Moore). Since the

solution tree has *b* independent branches (Figure 3 in Appendix-B) this algorithm is called independent beam search.

5.2 Dependent Beam Search (BS-D)

Dependent beam search algorithm is a slightly modified version of the independent beam search algorithm. In the independent beam search tree, after the second level only one node is expanded to the next level among the nodes that are expanded from the same parent. However, in the dependent beam search case, all the nodes at a level are evaluated together without considering their parent nodes and *b* nodes with the smallest $\overline{F}$ values are expanded to the next level. This implies that more than one node that have same parent node can be expanded to the next level. An example is given in Appendix-C.

5.3 Genetic and Tabu-search Algorithms (GA and TS)

As stated before, a genetic algorithm and a tabu-search algorithm are developed Kardas and Sabuncuoglu (2006) and Aydogdu and Sabuncuoglu (2006). GA and TS are explained in detail in Appendix D and E.

Chapter 6

COMPUTATIONAL EXPERIMENTS

In order to evaluate the performances of the proposed heuristics, we conducted experiments on several randomly generated problems with sizes of 20, 30, 40, 60, 80, 100, 150 jobs. The processing times are taken as uniformly distributed in the range [0,25] and [0,100] representing low and high processing time variability, respectively. The due dates are also distributed uniformly on the four different ranges as shown in Table 2. Here, SP denotes the sum of processing times of the N jobs. Note that, these due date and processing time distributions are used in Keha and Koksalan (2003).

Table 2: Due Date Ranges

Due Date Type	Due Date Range
I	[0,0.4SP]
II	[0.1SP, 0.3SP]
III	[0.25SP,.45SP]
IV	[0.3SP, 1.3SP]

Before performing an extensive numerical study, we solved about 50 sample problems with 20, 30, 40 and 60 jobs to gain some insights on the significant beam width values to use in our experiments with BS-I and BS-D. For this purpose, we consider the behavior of average percentage deviation of each heuristic from optimum with respect to increasing beam widths. In this context, we define the average percentage deviation of a heuristic from optimum as

$$\text{Average Percentage Deviation} = \frac{\sum_{m=1}^{M} \sum_{n=n_T(m,Moore)}^{n_T(m,SPT)} 100 \times \frac{\overline{F}(m,n) - \overline{F}_{OPT}(m,n)}{\overline{F}_{OPT}(m,n)}}{\sum_{m=1}^{M} \sum_{n=n_T(m,Moore)}^{n_T(m,SPT)} \varphi_{m,n}}.$$

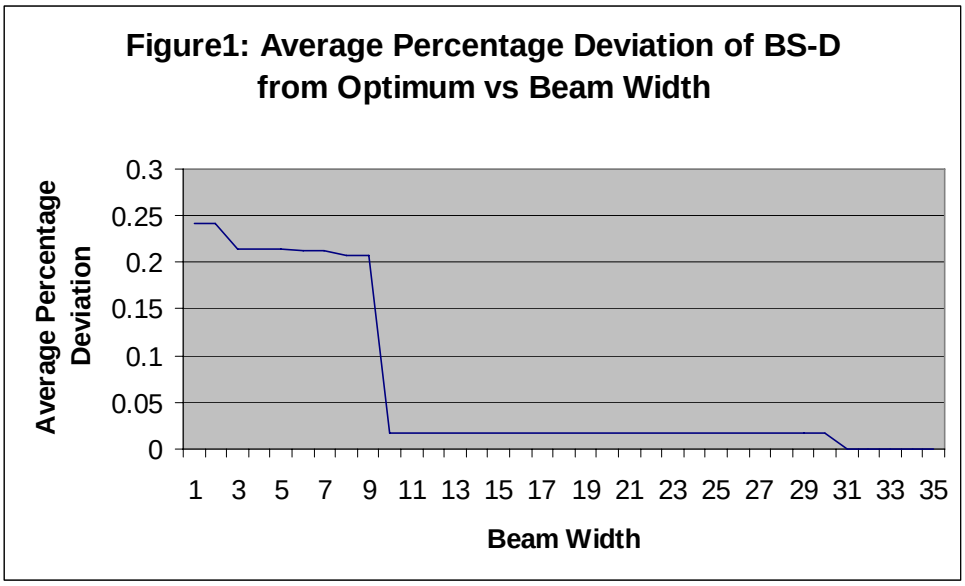
Here, M is the total number of problems, $\overline{F}(m,n)$ is the minimum mean flowtime resulting from the heuristic solution of the m^{th} problem for $n_T = n$, and $\overline{F}_{OPT}(m,n)$ is the corresponding optimal solution. $n_T(m,Moore)$ and $n_T(m,SPT)$ are the number of tardy jobs in sequences formed according to Moore's Algorithm and SPT order, respectively, for the m^{th} problem. Finally, $\varphi_{m,n}$ is defined as

$$\varphi_{m,n} = \begin{cases} 1, & \text{if } \overline{F}(m,n) > \overline{F}_{OPT}(m,n) \\ 0, & \text{o.w.} \end{cases}$$

Average percentage deviation illustrates the average gap between the heuristic and the optimal solution over all efficient schedules and test problems where this gap is positive. These cases will be referred to as *deviation instances* in the rest of the thesis. Figure 1 illustrates average percentage deviation of BS-D with respect to increasing beam width values in sample problems. As seen in Figure 1, the average percentage deviation is stabilized after a beam width value of 10. Therefore, in the rest of the experiments we use BS-I and BS-D with beam width value 10.

6.1 Comparison with the Optimal Solution

In this subsection, we report the results of our comparison of the proposed heuristics with the optimal solution considering several measures. Since Nelson's B&B Algorithm can solve problems with size up to 60 jobs within reasonable amount of time, comparing the heuristics with optimal solution on larger size



problems is not possible. Therefore we decided to solve problems with sizes of 20, 30, 40, 60 jobs. The processing time and due date values for these problems have been generated according to our discussion early in Chapter 6. For each job size, due date and processing time distribution, we solved 5 randomly generated problems.

Considering all possible combinations, we have solved 40 (2x4x5) problems for each job size which makes 160 in total. These 160 problems were solved with Nelson's B&B method, BS-I, BS-D, Keha and Koksalan (2003)'s genetic algorithm (GA(K&K)), Koktener and Koksalan (2000)'s simulated annealing (SA(K&K)), proposed tabu-search (TS) and proposed genetic algorithm (GA). Keha and Koksalan (2003) use tournament selection method to choose two parent schedules which are modified in order to build two new schedules. Tournament selection is choosing the best schedules with respect to a fitness function as parents among a number of randomly selected schedules. This number is referred as tournament size. For our experiments we take tournament size as 5. In addition, we also solved the test problems with a heuristic suggested by Nelson et al. (1986). This heuristic is based on expanding the node with minimum flowtime at

each level of a given B&B tree of Nelson et al.'s optimum solution. We recognize that this heuristic is nothing but a special version of our proposed beam search algorithms with beam width 1. In the rest of the thesis, we refer to this heuristic as *Nelson's Heuristic*.

In addition to the average percentage deviation, the following three measures were considered in our experiments.

i) *Maximum Percentage Deviation*: $\max_{(m,n)} (100 \times \frac{\overline{F}(m,n) - \overline{F}_{OPT}(m,n)}{\overline{F}_{OPT}(m,n)})$.

ii) ND/ N_{Total} where

Total Number of Deviation Instances (*ND*): $\sum_{m=1}^{M} \sum_{n=n_T(m,Moore)}^{n_T(m,SPT)} \varphi_{m,n}$

Total Number of Efficient Schedules (N_{Total}): $\sum_{m=1}^{M} \sum_{n=n_T(m,Moore)}^{n_T(m,SPT)} 1$

iii) *Average CPU Time*: The average computation time the heuristic spent in solving a test problem.

Table 3 illustrates the results of our experiments with 20, 30, 40, 60 jobs. The results indicate that both beam search based heuristics and Nelson's Heuristic perform better than GA(K&K) and SA(K&K) according to all performance measures. Only in the 60 jobs case, the average percentage deviation value of the BS-I seems to be larger than the GA(K&K). The reason behind this is that BS-I's average percentage deviation is calculated according to only 3 deviation instances. Since the deviation value in one of these few instances are high, average percentage deviation value of BS-I is higher than the GA(K&K). However, we conclude that both BS-I performs better than the GA(K&K) since the other performance measures favor beam search based algorithm.

Nelson's Heuristic, BS-I and BS-D find nearly all efficient schedules optimally. As expected, both algorithms perform a bit better than the Nelson's Heuristic since Nelson's Heuristic is equivalent to BS-I or BS-D with beam width

1. BS-D performs slightly better than BS-I for the problems with 60 jobs. Although the performance of the GA(K&K), SA(K&K) and Nelson's Heuristic worsens as the size of the problem increases, the performance of our proposed beam search based heuristics is quite stable with respect to problem size. Indeed, all the problems with job sizes 20, 30 and 40 were solved optimally by the proposed beam search algorithms. Only among the problems with 60 jobs, there are some instances where the $\overline{F}$ value found by BS-I and BS-D for a test problem deviate from optimum.

Both GA and TS perform better than the GA(K&K) and SA(K&K) but not as good as beam search based algorithms. GA performs a bit better than TS according to the average percentage deviation criterion. However, TS finds more efficient schedules optimally than the GA does, for the problems with 40 and 60 jobs. Their average deviation values seem to be stable according to the job size. However, as job sizes increases the rate ND/N_{Total} increases for both TS and GA algorithms. Therefore, performances of these heuristics are negatively affected by the increasing problem size.

As Table 3 shows, Nelson's Heuristic is the fastest of all 5 approximate solution methods that we tested in this experiment. Although GA(K&K)'s solution quality is better than SA(K&K)'s, GA(K&K) is much slower than SA(K&K). GA and TS are the two slowest heuristics. Indeed, Nelson's Heuristic performs much better than these four methods resulting with less CPU time. Both of our proposed beam search algorithms work slightly slower than the Nelson's Heuristic but faster than the others.

Tables 4 and 5 illustrate the average percentage deviation from optimum solution for each due date distribution type and for each problem size with low and high processing time distributions, respectively. These tables illustrate that BS-I, BS-D and Nelson's Heuristic provide better solutions than GA, SA, SA(K&K) and

Table 3: Comparison of the Heuristics with the Optimum Solution

Problem Size	Performance Measure	Nelson's Heuristic	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
20 Jobs								
	Average Deviation	0.2829%	0%	0%	0.9317%	5.2628%	0.1235%	0.7900%
	ND/N_{Total}	1/96	0/96	0/96	25/96	95/96	3/96	15/96
	Max. Deviation	0.2829%	0%	0%	7.1793%	35.6627%	0.1755%	4.5668
	CPU Time (millisecond)	13.15	14.09	20.1	1439.38	1513	1785.22	1913.46
30 Jobs								
	Average Deviation	1.2646%	0%	0%	0.4358%	3.7633%	0.2425%	0.2154%
	ND/N_{Total}	5/140	0/140	0/140	110/140	138/140	29/140	29/140
	Max. Deviation	3.0060%	0%	0%	3.6200%	20.1754%	0.8960%	0.9692%
	CPU Time (millisecond)	19.53	26.2	28.625	3222.3	4150.43	6641.41	6821.01
40 Jobs								
	Average Deviation	0.3135%	0%	0%	0.5675%	4.0801%	0.1753%	0.5190%
	ND/N_{Total}	3/176	0/176	0/176	125/176	172/176	51/176	41/176
	Max. Deviation	0.8793%	0%	0%	3.2100%	33.4914%	0.6431%	4.7802%
	CPU Time (millisecond)	25.87	36.5	41.83	8722.6	6382.05	17765.3	18535.47
60 Jobs								
	Average Deviation	0.6744%	0.8321%	0.0616%	0.4681%	3.3483%	0.2450%	0.3112%
	ND/N_{Total}	7/262	3/262	2/262	221/262	259/262	122/262	72/262
	Max. Deviation	2.2213%	2.2213%	0.0695%	2.8814%	26.9683%	3.3462%	4.2062%
	CPU Time (millisecond)	45.3	89.83	94.18	38748	12294.34	79296.9	83080.77

GA(K&K) with respect to each job size, processing time and due date distribution type. In fact, BS-I and BS-D deviate from the optimal solution only in the problems with 60 jobs, high processing times and Type 1 due dates.

Tables 4 and 5 also indicate that the problems generated by using Type IV due date distribution are solved quite effectively by the beam search based heuristics and Nelson's Heuristic. This distribution type represents problems with loose due dates, which implies that beam search based algorithms work well for the problems with loose due dates. Although these algorithms work also well for the problems with tighter due date distribution types (I, II, III), the most deviation instances occur in these problem types. Processing time distribution, on the other hand, does not affect the solution quality of BS-I and BS-D.

For Nelson's Heuristic, deviation from optimality mostly occurs for the problems with low processing times combined with Type 1 due dates and for problems with high processing times combined with Type 2 due dates. It can also be seen that BS-I and BS-D algorithms perform better in the problems with high processing time variability. The same situation is also valid for the GA and TS algorithms. In Appendix F, the performance measure given in Table 3 is presented for each processing time and due date distribution in detail for 20, 30, 40 and 60 jobs cases.

Although the quality of the solutions generated by beam search based heuristics is quite stable with respect to problem sizes, we observe that as the problem size increases Nelson's Heuristic, BS-I, BS-D, SA(K&K) and TS algorithms may fail to find a solution for some of the efficient schedules. As stated before, for a given problem, there are n_T(SPT) - n_T(Moore) + 1 efficient schedules and it is desired to find each of these schedules approximately. However, in some of the 160 test problems, beam search based heuristics, Nelson's Heuristics, SA(K&K) and TS fail to find an approximate solution

Table 4: Average Deviation from Optimum in the Problems with Low Processing Time Variability

Problem Size	Due Date Type	Nelson's Heuristic	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
20 Jobs								
	I	0.28%	0%	0%	0.49%	3.37%	0.14%	0.44%
	II	0%	0%	0%	1.02%	5.04%	0%	0.94%
	III	0%	0%	0%	1.57%	3.62%	0%	1.66%
	IV	0%	0%	0%	0.35%	7.25%	0%	0%
30 Jobs								
	I	0%	0%	0%	0.60%	3.50%	0.37%	0.12%
	II	0%	0%	0%	0.20%	5.23%	0.08%	0.13%
	III	0%	0%	0%	0.39%	2.87%	0%	0.64%
	IV	1.95%	0%	0%	0.62%	4.94%	0.31%	0.24%
40 Jobs								
	I	0.36%	0%	0%	1.00%	2.50%	0.29%	0.97%
	II	0%	0%	0%	0.17%	2.48%	0.05%	0.06%
	III	0.06%	0%	0%	0.19%	4.90%	0.06%	0.04%
	IV	0%	0%	0%	0.45%	7.67%	0%	0.52%
60 Jobs								
	I	0.82%	0.83%	0.06%	0.87%	2.21%	0.45%	0.36%
	II	0%	0%	0%	0.15%	2.59%	0.03%	0.04%
	III	0%	0%	0%	0.12%	5.49%	0.03%	0.01%
	IV	0%	0%	0%	0.07%	4.67%	0.05%	0.17%

Table 5: Average Deviation from Optimum in the Problems with High Processing Time Variability

Problem Size	Due Date Type	Nelson's Heuristic	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
20 Jobs								
	I	0%	0%	0%	1.55%	3.61%	0.11%	0.88%
	II	0%	0%	0%	0.12%	4.20%	0%	0%
	III	0%	0%	0%	0.26%	5.52%	0%	0%
	IV	0%	0%	0%	0.74%	6.76%	0%	0.05%
30 Jobs								
	I	0.19%	0%	0%	0.47%	2.29%	0.39%	0.08%
	II	2.39%	0%	0%	0.09%	3.28%	0%	0.20%
	III	0.005%	0%	0%	0.02%	4.80%	0%	0%
	IV	0%	0%	0%	0.40%	3.62%	0.14%	0.64%
40 Jobs								
	I	0%	0%	0%	0.79%	1.74%	0.18%	0.41%
	II	0.88%	0%	0%	0.23%	3.26%	0.13%	0%
	III	0%	0%	0%	0.24%	4.89%	0.02%	0.01%
	IV	0%	0%	0%	0.25%	7.19%	0%	0%
60 Jobs								
	I	0%	0%	0%	0.79%	2.16%	0.27%	1.05%
	II	0.62%	0%	0%	0.20%	3.21%	0.04%	0.005%
	III	0.001%	0%	0%	0.10%	3.76%	0.02%	0.005%
	IV	0%	0%	0%	0.37%	4.48%	0.11%	0.10%

specifically for the efficient schedule that have n_T(Moore) tardy jobs (Table 6). The number of the problems such a situation occurs is relatively small, and most of the cases that can not be solved by Nelson's Heuristic, are solved by BS-I and BS-D. Nevertheless, the number of these instances seems to be increasing as the problem size increases. In order to see whether this trend will continue for larger problem sizes and to better observe the performance of our heuristics, we performed some further experiments on problems with 80,100 and150 jobs.

Table 6: Number of Efficient Schedules for which No Solution is Found

Heuristic	20 Jobs	30 Jobs	40 Jobs	60 Jobs
Nelson's Heuristic	0/96	0/140	2/176	3/262
BS-I	0/96	0/140	1/176	0/262
BS-D	0/96	0/140	0/176	0/262
GA(K&K)	0/96	0/140	0/176	0/262
SA(K&K)	1/96	1/141	1/176	3/262
GA	0/96	0/141	0/176	0/262
TS	2/96	6/141	5/176	18/262

6.2 Experiments on Larger Problems

We generated larger size problems with 80, 100 and 150 jobs using the same processing time and due date distributions stated before. For each job size, processing time and due date distribution type, we generated 5 problems and obtained 120 problems in total. We compared Nelson's Heuristic with BS-I, BS-D, GA, TS, SA(K&K) and GA(K&K) algorithms and measured their relative performance. In our experiments with larger problems, we first consider average percentage difference of each heuristic's solution from Nelson's Heuristic. The average percentage difference is the arithmetic mean of the percentage differences

over all the efficient solutions and all the problems with the same size. In mathematical terms, it is defined as follows.

$$\text{Average Percentage Difference} = \frac{\sum_{m=1}^{M}\sum_{n=n_T(m,Moore)}^{n_T(m,SPT)} 100 \times \frac{\overline{F}_{Nelson}(m,n) - \overline{F}\ (m,n)}{\overline{F}_{Nelson}(m,n)}}{\sum_{m=1}^{M}\sum_{n=n_T(m.Moore)}^{n_T(m,SPT)} \psi_{m,n}}$$

where $\overline{F}\,(m,n)$ is the minimum flowtime provided by the considered heuristic for the m^{th} problem when $n_T = n$. $\overline{F}_{\text{Nelson}}(m,n)$ is the minimum flowtime resulting from Nelson's Heuristic. $n_T(m,Moore)$ and $n_T(m,SPT)$ are the number of tardy jobs in the sequences formed according to Moore's Algorithm and SPT order, respectively, for the m^{th} problem. Finally, $\psi_{m,n}$ is given below.

$$\psi_{m,n} = \begin{cases} 1, & \text{if } \overline{F}(m,n) \neq \overline{F}_{Nelson}(m,n) \\ 0, & \text{o.w.} \end{cases}$$

The other measures we consider in the experiments with larger problems are as follows.

N+: Number of cases where a specific heuristic performs better than the Nelson's Heuristic. $\text{N+} = \sum_{m=1}^{M}\sum_{n=n_T(m,Moore)}^{n_T(m,SPT)} \eta_{m,n}$ where

$$\eta_{m,n} = \begin{cases} 1, & \text{if } \overline{F}(m,n) > \overline{F}_{Nelson}(m,n) \\ 0, & \text{o.w.} \end{cases}$$

N-: Number of cases a specific heuristic performs worse than the Nelson's Heuristic. $\text{N-} = \sum_{m=1}^{M}\sum_{n=n_T(m,Moore)}^{n_T(m,SPT)} \mu_{m,n}$ where

$$\mu_{m,n} = \begin{cases} 1, & \text{if } \overline{F}(m,n) < \overline{F}_{Nelson}(m,n) \\ 0, & \text{o.w.} \end{cases}$$

Maximum Percentage Difference: $\max_{(m,n)} (100 \times \frac{\overline{F}_{Nelson}(m,n) - \overline{F}(m,n)}{\overline{F}_{Nelson}(m,n)})$.

Minimum Percentage Difference: $\min_{(m,n)} (100 \times \frac{\overline{F}_{Nelson}(m,n) - \overline{F}(m,n)}{\overline{F}_{Nelson}(m,n)})$.

The corresponding results are presented in Table 7. As seen in the table, proposed heuristics and Nelson's Heuristic perform better than the SA(K&K), GA(K&K), GA and TS, also in larger size problems with respect to all these measures. As it can be understood from the $N+/N_{Total}$ measure, in more than %90 of the cases Nelson's Heuristic performs better than or equal to these iterative algorithms.

Proposed beam search algorithms perform slightly better than the Nelson's Heuristic. As the job size increases, number of instances in which proposed heuristics perform better than the Nelson's Heuristic increases. We also observe that BS-D performs slightly better than the BS-I on the problems with larger job sizes. In most of the cases, however, their solution qualities are almost the same. As it can be seen in Table 7, BS-D outperforms Nelson's Heuristic in a few more instances than BS-I does.GA and TS perform better than the GA(K&K) and SA(K&K) almost for all measures presented in Table 7. TS and GA's performance are nearly same for the cases in which they both find a solution. In

Table 7: Comparison of the other Heuristics with Nelson's Heuristic

Problem Size	Performance Measure	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
80 Jobs							
	Average % Difference	0.143%	0.140%	-0.487%	-2.835%	-0.213%	-0.263%
	N+ / N_{Total}	14/394	15/394	4/394	0/394	5/394	5/394
	N- / N_{Total}	0/394	0/394	352/394	378/394	264/394	169/394
	Max % Difference	0.526%	0.526%	0.526%	-0.014%	0.526%	0.359%
	Min % Difference	0.000%	0.000%	-3.693%	-24.480%	-1.874%	-4.132%
100 Jobs							
	Average % Difference	0.032%	0.029%	-0.612%	-3.150%	-0.320%	-0.275%
	N+ / N_{Total}	15/484	17/484	8/484	0/484	12/484	7/484
	N- / N_{Total}	0/484	0/484	423/484	451/484	345/484	245/484
	Max % Difference	0.161%	0.161%	0.273%	-0.031%	0.273%	0.116%
	Min % Difference	0.000%	0.000%	-5.313%	-29.500%	-2.780%	-4.642%
150 Jobs							
	Average % Difference	0.085%	0.085%	-0.537%	-4.000%	-0.287%	-0.276%
	N+ / N_{Total}	30/796	31/796	3/796	0/796	4/796	7/796
	N- / N_{Total}	0/796	0/796	747/796	687/796	703/796	574/796
	Max % Difference	1.225%	1.225%	1.458%	-0.074%	1.450%	1.438%
	Min % Difference	0.000%	0.000%	-3.340%	-38.030%	-3.342%	-5.941%

these cases, overall average difference from Nelson's Heuristic is nearly same. GA's maximum deviation values are less than TS's, and number of instances that TS performs as well as Nelson's Heuristic is more than those that GA does. However, the real handicap of TS is that there are considerable number of instances in which it can not find an approximately efficient schedule for some N_T values (Table 8). GA, on the other hand, finds efficient schedules approximately for every instance.

In Appendix G, the performance measure given in Table 7 is presented for each processing time and due date distribution in detail for the 80, 100 and 150 jobs cases. According to the tables in Appendix G, most of the instances in which BS-I and BS-D perform better than the Nelson's Heuristic occur among the test problems with Type I due date. These problems also require much more CPU time than the others. In addition, GA performs better than the TS in the problems with Type I and IV due date distributions and TS performs better in the problems with Type II and III due date distributions.

We again observed the cases where compared heuristics fail to find approximately efficient schedules for some of the N_T values in the efficient range. The number of such instances is given in Table 8. This table illustrates that as the size of the problem increases such cases appear more frequently for Nelson's Heuristic. Problems, where feasible solutions cannot be found, frequently coincide with Type I due date distribution, and less frequently with Type II and III. BS-D and BS-I algorithms halved the number of these cases in the problems with 80 and 100 jobs. However, the experiments on the problems with 150 jobs demonstrate that the performance of our proposed algorithms on this issue worsens as the size of the problem increases.

Table 8: Number of Efficient Schedules for which No Solution is Found

Heuristic	80 Jobs	100 Jobs	150 Jobs
Nelson 's Heuristic	8/394	16/484	22/797
BS-I	5/394	7/484	18/797
BS-D	5/394	7/484	17/797
GA(K&K)	0/394	0/484	0/797
SA(K&K)	12/394	30/484	107/797
GA	0/394	0/484	0/797
TS	36/394	75/484	70/797

While the number of no solution cases is quite high for SA(K&K) and TS algorithms, GA and GA(K&K) find an approximate solution for every N_T value of the problems considered in our experiments. Nevertheless, as it can be seen in Table 9, the computation time requirements for GA and GA(K&K) are a lot more than that of the beam search based algorithms. Therefore, for large size problems, if the decision makers desire to find approximately efficient schedules for all N_T values in the efficient range, they should first use BS-D or BS-I algorithms in order to minimize the number of instances where no solution is found. Then genetic algorithm should be used to solve the remaining instances.

Table 9: Average CPU Time in Milliseconds

Heuristic	80 Jobs	100 Jobs	150 Jobs
Nelson's Heuristic	80.03	135.95	486.28
BS-I	276.13	570.7	2601.5
BS-D	258.15	565.25	2517.55
GA(K&K)	136836	26925.5	2057585.5
SA(K&K)	27836.76	48184.16	139499.3
GA	272333.6	681263.7	4094870.2
TS	272381.5	630742.8	3273617.3

Chapter 7

CONCLUSION

As a result of our experiments, we concluded that BS-D and BS-I perform quite well for the multicriteria scheduling problem of minimizing average flowtime and number of tardy jobs. In most of the cases, these two algorithms find the efficient schedules optimally. Even in the cases where BS-D or BS-I deviate from optimum, the deviation is quite small and the deviation is stable with respect to problem size. In addition, both BS-D and BS-I perform better than the other heuristics given in this thesis with respect to all problem types that we test. The only disadvantage of our proposed beam search heuristics is that, they, although rarely in some cases, fail to find approximately efficient solutions for some of the n_T values in the efficient ranges. For such cases, we propose that GA or GA(K&K) be used.

We believe that the good performance of our proposed approach is due to the beam search mechanism. In fact, GA(K&K) and SA(K&K), which are two existing heuristics in the literature, search among all possible sequences for the efficient schedules. However, BS-I and BS-D limit the search space by utilizing Theorem 1 and Smith's Algorithm. Hence, they find better solutions by searching a smaller space and more efficiently than GA(K&K) and SA(K&K) do.

Theorem 1 and Smith's Algorithm are also utilized by GA and TS. Therefore, GA and TS outperform GA(K&K) and SA(K&K). However, our

experiments show that in general the proposed beam search algorithms perform better than GA and TS. Since, this study shows that utilizing the characteristics of the efficient solutions in the approximate or optimal solution methods to limit the search space is quite effective; we believe that such a beam search mechanism can be also quite beneficial to solve the other multicriteria scheduling problems. Therefore, we strongly suggest using this technique in the future multicriteria scheduling studies.

As further studies, BS-I and BS-D can also be applied to the other bicriteria single machine problems such as minimizing weighted flowtime and number of tardy jobs, and minimizing weighted flowtime and weighted number of tardy jobs. With the insights gained from this study, we already extended our current research to consider the first problem. We consider the second problem which seems to be more challenging, as a future work.

We believe that beam search applications are quite promising to solve multicriteria scheduling problems in general. Therefore, another line of research may extend this work to more complex settings, such as parallel machine environments. As a final open area of possible investigation, we note the robustness of the solutions which is a fundamental application issue.

BIBLIOGRAPHY

Aydogdu, M., Sabuncuoglu, I., 2006. "A Tabu-searh algorithm for the single machine bicriteria scheduling problem: Minimization of average flowtime and number of tardy jobs". *Technical Report, Bilkent University IE Department*.

Billaut, J.-C., T_kindt, V., 1999. "Some guidelines to solvemulticriteria scheduling problems", *IEEE International Conference on Systems, Man and Cybernetics Proceeding.* 6, 463–468.

Blum, C., 2002. "ACO applied to group shop scheduling: A case study on intensification and diversification, in M. Dorigo, G. Di Caro and M. Sampels (eds)", *Proceedings of ANTS 2002 – From Ant Colonies to Artificial Ants: Third International Workshop on Ant Algorithms*, *Lecture Notes in Computer Science.* 2463, 14–27.

Chen, C.L., Bulfin, R.L., 1993. "Complexity of single machine multi-criteria scheduling problems", *European Journal of Operational Research.* 70, 115–125.

Dileepan. P, Sen. T., 1988. "Bicriterion static scheduling research for a single machine", *Omega.* 16-1, 53-59.

Duarte, R., Rego, C., Gamboa, D., 2004. "A Filter and Fan Approach for the Job Shop Scheduling Problem: A Preliminary Study", *Proceedings: International Conference on Knowledge Engineering and Decision Support.* 401-406.

Emmons, H., 1975. "One machine sequencing to minimize mean flowtime with minimum tardy", *Naval Research Logistics Quarterly.* 22-3, 585-592.

Erol, S., Guner, E., Tani, K., 1998. “One machine scheduling to minimize maximum earliness with minimum number of tardy jobs”, *International Journal of Production Economics*. 55, 213-219.

Fry, T., Armstrong, R., Lewis H., 1989. “A framework for single machine multiple objective scheduling research”, *Omega.* 17-6, 595 - 607.

Ghirardi, M., Potts, C. N., 2005. “Makespan minimization for scheduling unrelated parallel machines: A recovering beam search approach”, *European Journal of Operational Research*. 165-2, 457–467.

Heck, H., Roberts, S., 1992. “A note on the extension of a result on scheduling with secondary criteria”, *Naval Research Logistics Quarterly.* 19, 403-405.

Hoogeveen, J.A., 2005. “Multicriteria Scheduling”, *European Journal of Operational Research.* 167-3, 592-623.

Huo, Y., Leung, J. Y. T., Zhao, H., 2004. “Complexity of two-dual criteria scheduling problems”, *Submitted to Operations Research Letters*.

Kardas, S., Sabuncuoglu, I., 2006. “A Genetic algorithm for the single machine bicriteria scheduling problem: Minimization of average flowtime and number of tardy jobs”. *Technical Report, Bilkent University IE Department*.

Keha, A. B., Koksalan, M., 2003. “Using genetic algorithms for single-machine bicriteria scheduling problems”, *European Journal of Operational Research*. 145, 543–556.

Kiran, A. S., Unal A. T., 1991. “A single-Machine Problem with multiple Criteria”, *Naval Research Logistics*. 38, 721-727.

Koksalan, M., 1999. “A Heuristic Approach to Bicriteria Scheduling”, *Naval Research Logistics*. 46-7, 777 - 789.

Koksalan, M., Azizoglu, M., Kondakci, S., 1998. “Minimizing flowtime and maximum earliness on a single machine”, *IIE Transactions.* 30, 192–200.

Koktener, E.K., Koksalan, M., 2000. “A simulated annealing approach to bicriteria scheduling problems on a single machine”, *Journal of Heuristics.* 6, 311–327.

Kondakci, S., Azizoglu, M., Köksalan, M., 2003. “Scheduling with multiplecriteria”, *Computers & Industrial Engineering*. 45-2, 257-269.

Kondakci, S. K., Bekiroglu T., 2000. ”Scheduling with bicriteria: total flowtime and number of tardy jobs”, *International Journal of Production Economics.* 53, 91- 99.

McMullen, P., Tarasewich, P., Frazier, G., 2000, “Using genetic algorithms to solve the multi-product JIT sequencing problem with set-ups”, *International Journal of Production Research*. 38-12, 2653-2670.

Moore, J. M., 1968. “An n job, one machine sequencing algorithm for minimizing the number of late jobs”, *Management Science.* 15, 102-109.

Nagar, A., Haddock, J., Heragu, S., 1995. “Multiple and bicriteria scheduling: A literature survey”, *European Journal of Operations Research.* 81, 88 –104.

Nelson, R. T., Sarin, R. K., Daniels, R. L., 1986. “Scheduling with multiple performance measures: the one machine case”, *Management. Science* 32-4, 464-479.

Sabuncuoglu, I. Bayiz. M., 1999. “Job shop scheduling with beam search”, *European Journal of Operational Research.* 118-2, 390-412.

Sabuncuoglu, I., Karabuk, S., 1998. “A beam search algorithm and evaluation of scheduling approaches for FMSs”, *IIE Transactions.* 30-2, 179-191.

Sen, T., Gupta, S. K., 1983. “A branch and bound procedure to with multiple performance measures: the one machine case”, *Management Science.* 32, 464-479.

Smith, W. E., 1956. “Various Optimizers for Single Stage Production”, *Naval Research Logistics Quarterly* 3, 1-2.

T’kindt, V., Bouibede-Hocine, K., Esswein, C., 2005. “Counting and enumeration complexity with application to multicriteria scheduling”. *A quarterly Journal of Operations Research*. 3-1, 1-21.

Wan, G., Yen, B. P. C., 2003. “Single Machine Bicriteria Sceduling: A survey”, *International Journal of Industrial Engineering*. 10 -3 ,222-231.

APPENDIX

A. Example for Nelson et al.'s (1986) Branch and Bound Algorithm

The process times and the due dates of our example are given below.

Table 1: Sample Problem Parameters

Job(j)	Processing Time(Pj)	Due Date (dj)
1	1	40
2	2	3
3	3	5
4	5	7
5	10	20
6	15	32

For this problem n_T(Moore) = 1, $\overline{F}$ (Moore) = 19.83 , n_T(SPT) = 4 and $\overline{F}$ (SPT) = 13. Therefore, in this problem we need to find 4 efficient schedules. Note that, SPT order is the efficient schedule corresponding to n_T = 4. In the SPT order, Job 1 and Job 2 are early and the remaining jobs are tardy. The corresponding Branch and Bound tree has four levels as seen in Figure 2. The single starting node in level 1 represents the SPT order.

As seen in the Figure 2, each node *k* has a set of early jobs denoted as E_k. Since the starting node at level 1 represents SPT order, E_1: {1, 2} is the set of jobs that is early in the SPT order. In the efficient schedule for n_T = 3, besides Job 1 and Job 2 another job among 3,4,5 and 6 must be also early (See Theorem 1). Thus, at the second level, these four jobs are tried one by one by

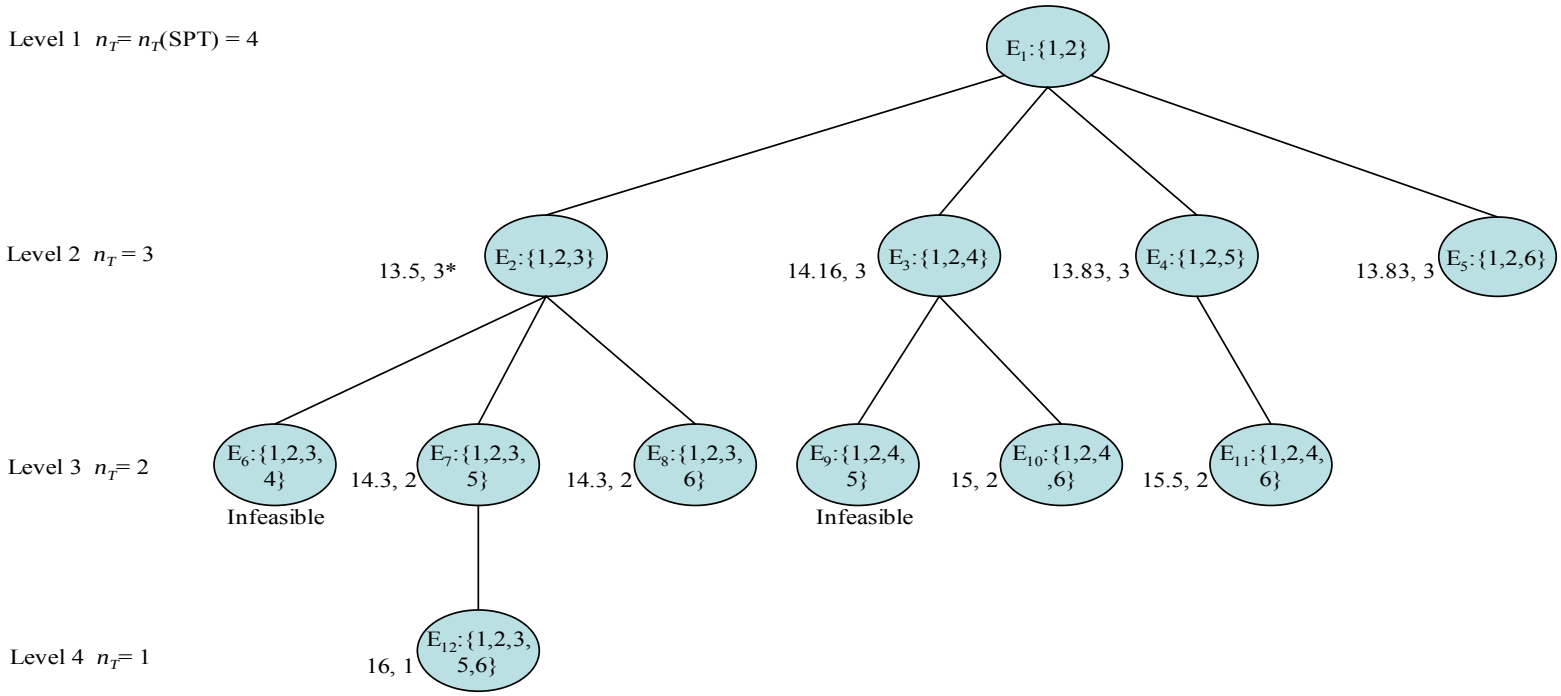


Figure 2: B&B Tree for the Sample Problem

* The first entry refers to the minimum flowtime when the jobs in set E_2 *are nontardy and the second entry is the number of tardy jobs.*

being added to the set of early jobs referred by 2nd, 3rd, 4th and 5th nodes. For example, in the 2nd node, Job 3 is added to E_3, and then, by using Smith's Algorithm, the schedule that gives minimum flowtime while keeping jobs 1, 2 and 3 non-tardy is found. The resulting flowtime is 13.5. In a similar fashion, other three nodes at the second level are constructed, and the minimum flowtime values are found as 14.16, 13.83 and 13.83. Since the smallest flowtime at this level is given by the second node, the efficient schedule for n_T = 3 corresponds to the sequence at this node.

At the third level, all possible subsets of {3,4,5,6} (tardy jobs in SPT order) with cardinality two is added to the early jobs of SPT order to find efficient schedule for n_T = 2. For this reason, each node at level 2 is further expanded by adding one more job. For example, node 2 (E_2: {1,2,3}) is expanded to the next level to form nodes 6, 7, and 8 by adding job 4, 5 and 6. Node 3 (E_3: {1,2,4}) is expanded to the next level to form nodes 9 and 10 by adding jobs 5 and 6. However, Job 4 is not added to E_3 to prevent repetition. After all nodes are expanded to level three, for each node, Smith's Algorithm is run. As a result, it appears that efficient schedule for n_T = 2 is the one that corresponds to E_7 = {1,2,3,5} with flowtime 14.3. The algorithm continues in this manner.

B. Example for Independent Beam Search Algorithm

As an example, we consider the problem given in Appendix-A. Let us solve the same problem also with independent beam search algorithm. As can be seen in Figures 2 and 3, the first two levels of both the B&B and the BS-I tree are the same. Both trees start with a node that represents SPT order. Then, four new nodes are expanded from this initial node by adding jobs 3,4,5 and 6 to E_2, E_3, E_4 and E_5, respectively. Using Smith's Algorithm, the schedules that give minimum flowtime while keeping the jobs in E_2, E_3, E_4 and E_5 nontardy are found. Since the schedules

found for nodes 2 and 4 have the smallest flowtime, they are selected to be expanded to the next level.

At level 3, the new nodes are generated by adding one more job to E_2 and E_4. Nodes 6, 7 and 8 are generated by adding jobs 4, 5 and 6 to E_2. Similarly, nodes 9, 10 and 11 are generated by adding jobs 3, 4 and 6 to E_4. As the schedules found for nodes 7 and 9 have the smallest *F* values, they are expanded to the next level. The algorithm continues in a similar way for the fourth level.

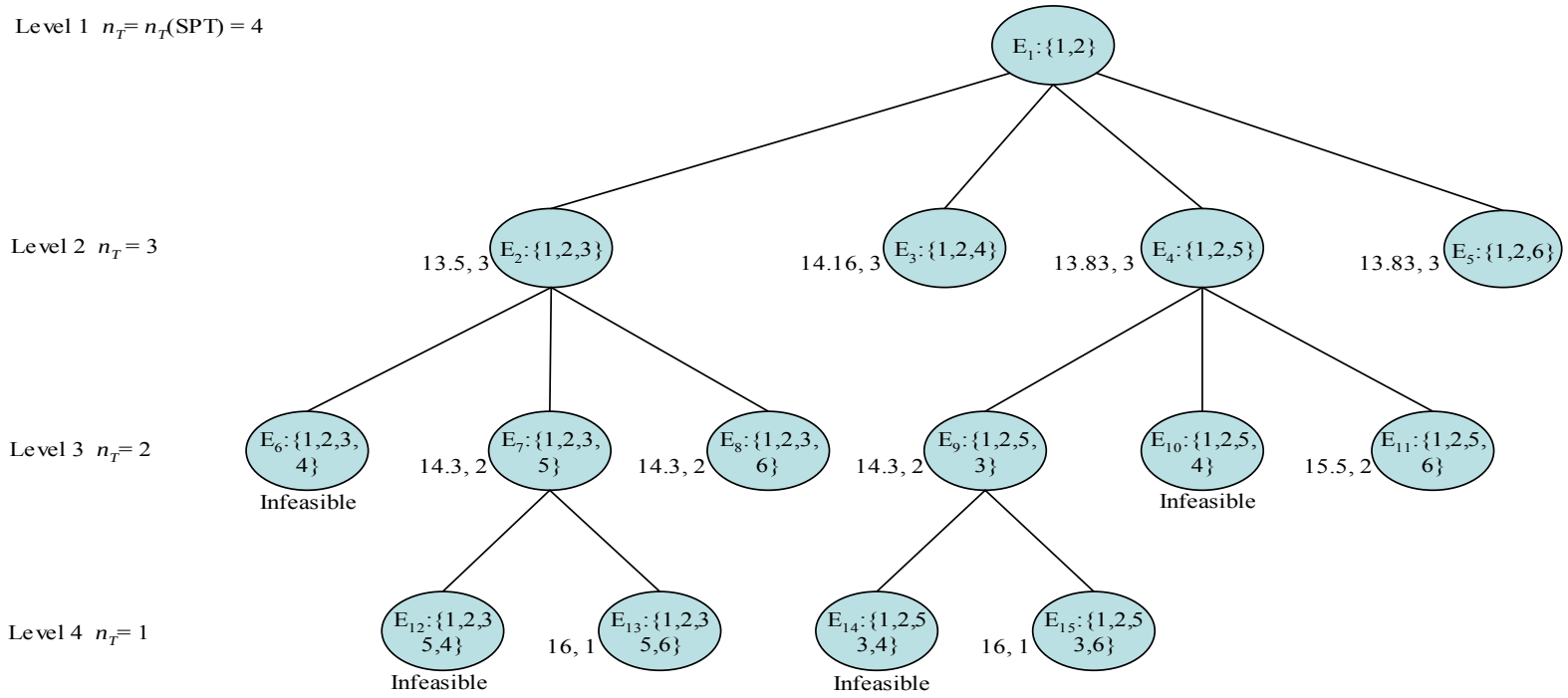


Figure 3: Independent Beam Search Tree for the Sample Problem where *b = 2*

C. Example for Dependent Beam Search Algorithm

Dependent beam search algorithm is also applied to the sample problem given in Appendix-A. The results are presented in Figure 4. As can be seen in Figure 4, at the third level, the two nodes with minimum F value are nodes 7 and 8. Without considering the fact that both these nodes are expanded from the node 2, they are expanded to the next level.

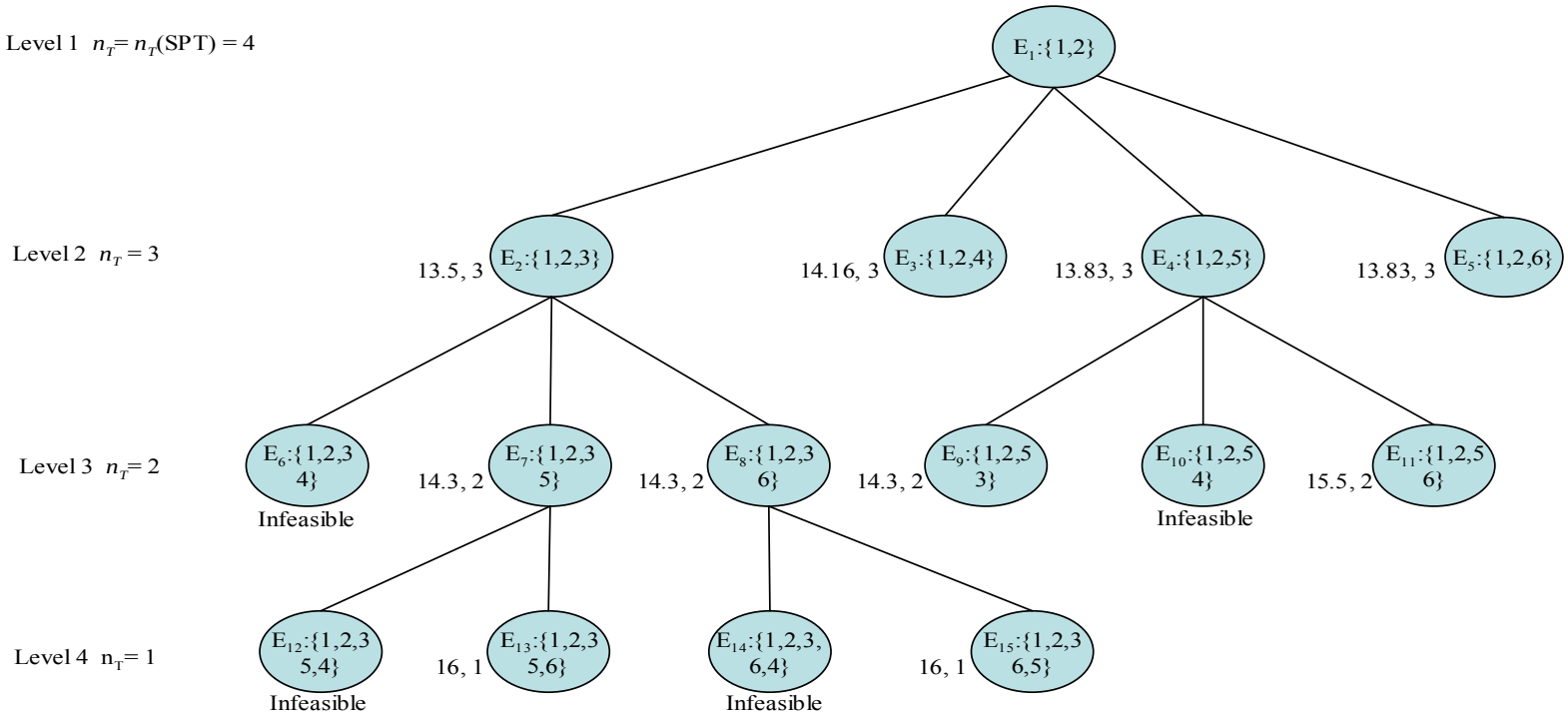


Figure 4: Dependent Beam Search Tree for the Sample Problem where *b = 2*

D. Tabu-search (TS)

Tabu search is a mathematical optimization method, belonging to the class of local search techniques. Tabu search (TS) is an iterative heuristic which avoids local optima by using memory structures called tabu list. Tabu lists temporarily record visited solutions and prevents algorithm to cycle around these solutions.

The jobs that are early in the SPT order must be also early in at least one of the efficient schedules with $n_T = n$ where $n_T(\text{SPT}) \geq n \geq n_T(\text{Moore})$ (Nelson et al., 1986). This theorem indicates that to find the efficient schedule for $n_T = n$, it is necessary to determine which other $n_T(\text{SPT}) - n$ jobs will be early besides the early jobs of SPT order. Therefore, subsets of SPT order's tardy jobs with cardinality $n_T(\text{SPT}) - n$ should be evaluated using Smith's Algorithm to find the schedule with minimum $\overline{F}$ while having n tardy jobs. The TS algorithm is based on searching these subsets by evaluating them with Smith's Algorithm to find approximately efficient schedules with $n_T = \text{n}$ where $n_T(\text{SPT}) \geq n \geq n_T(\text{Moore})$.

In other words, to find the efficient schedule with $n_T = n$, the subsets of the jobs that are tardy in the SPT order with cardinality $n_T(\text{SPT}) - n$ are searched. First, randomly a subset with cardinality $n_T(\text{SPT}) - n$ is selected and taken as current subset. Then, some neighbors of this current subset with cardinality $n_T(\text{SPT}) - n$ is generated. Next, these neighbors are evaluated by using Smith's Algorithm. The neighbor for which Smith's Algorithm gives the least $\overline{F}$ is accepted as the new current subset. After 100 iterations or every neighbor is appeared to be infeasible, the schedule that Smith's Algorithm finds for the current subset is accepted as approximately efficient schedule with $n_T = n$. Then, the same procedure is repeated for $n_T = n + 1$.

The procedure described above is a forward search starting from $n_T = n_T(\text{SPT}) - 1$ and continuing towards $n_T = n_T(\text{Moore})$. However, our initial runs indicate that

the forward search can not find an approximately efficient schedule for some $n_T = n$ where n_T(SPT) $\geq n \geq$ n_T(Moore) . Thus, a backward search is also performed by starting from n_T = n_T(Moore). In this backward search, the jobs that are tardy in Moore's Algorithm are allowed to be tardy at the every iteration. For each $n_T = n$, which other $n - n_T$(Moore) jobs will be also allowed to be tardy, is searched in the same manner as in forward search. After backward and forward search is completed, among the schedules that these searches find for $n_T = n$, the one with the smallest $\overline{F}$ is selected as the resulting approximately efficient schedule with $n_T = n$.

Neighborhood Selection

The neighbors of the current subset are generated by selecting a specific job from the current subset and replacing it with another job that is not an element of the current subset. Indeed, the selected job is replaced with every possible job one by one to generate all possible neighbors. A job is selected to be replaced with a probability that is inversely proportional to the number of times the job is selected before.

$$p_j = \frac{t_j}{\sum t_j}$$

$$t_j = \frac{1}{N_j \Big/ \sum_{j=1}^{N} N_j} \quad j \in \{1, 2, \ldots\ldots, N\}$$

p_j = Probability of Selecting Job j

N_j = Number of Times That Job j is Selected

Tabu List and Aspiration Criteria

The jobs from the current subset that are selected to be replaced are added to the tabu list. The jobs in the tabu list are not added to the current subset to generate its neighbors. Once a job is added to the Tabu list, it stays in the list for 5 iterations, since, our pilot experiments indicate that 5 iterations give good results for both small and large problem sizes. But if adding a specific job that is in the Tabu list to the current subset will give better results than the best result ever found, we use the job for that iteration, and then, we keep it in the Tabu list for the next 5 iterations.

Following pseudo algorithm is given to describe steps of forward and backward search of TS.

Forward Search

Step 0: Find the tardy jobs in the SPT sequence and define E_0 as the set of these jobs. Set n = 1, Iteration = 0.

Step 1: Take subset of E_0 with cardinality n randomly. Call this subset as $E_{current}$ and evaluate $E_{current}$ with Smith's Algorithm. Record the $\overline{F}$ value given by the schedule that Smith's Algorithm finds and refer to it as $\overline{F}_{\text{current}}$.

Step 2: Find the neighbors of $E_{current}$. Evaluate them using Smith's Algorithm and record the corresponding F value for each neighbor.

Step 3: If all the neighbors are infeasible according to Smith's Algorithm, then go to Step 6, other wise go to Step 4.

Step 4: The neighbor of $E_{current}$ with minimum $\overline{F}$ value is selected as the new $E_{current}$ and new $E_{current}$'s $\overline{F}$ value is assigned as new $\overline{F}_{\text{current}}$. Iteration = Iteration + 1.

Step 5: If Iteration <100 go to Step 2. Otherwise go to Step 6.

Step 6: The schedule that is found by evaluating $E_{current}$ with Smith's Algorithm is called as current approximately efficient schedule for $n_T = n$.

Step 7: $n = n+1$. If n_T(SPT) - $n < n_T$(Moore) go to Step 8. Otherwise go to Step 1.

Step 8: Terminate the algorithm.

Backward Search

Step 0: Find the non-tardy jobs in the sequence ordered according to Moore's Algorithm and define E_0 as the set of these jobs. Set $n = 1$, Iteration = 0.

Step 1: Take subset of E_0 with cardinality n randomly. Call this subset as $E_{current}$ and evaluate $E_{current}$ with Smith's Algorithm. Record the $\overline{F}$ value given by the schedule that Smith's Algorithm finds and refer to it as $\overline{F}_{\text{current}}$.

Step 2: Find the neighbors of $E_{current}$. Evaluate them with Smith's Algorithm and record the corresponding $\overline{F}$ value for each neighbor.

Step 3: If all the neighbors are infeasible according to Smith's Algorithm, then go to Step 6, other wise go to Step 4.

Step 4: The neighbor of $E_{current}$ with the minimum $\overline{F}$ value is selected as the new $E_{current}$ and new $E_{current}$'s $\overline{F}$ value is assigned as new $\overline{F}_{\text{current}}$. Iteration = Iteration + 1.

Step 5: If Iteration <100 go to Step 2. Otherwise go to Step 6.

Step 6: The schedule that is found by evaluating $E_{current}$ with Smith's Algorithm is called as current approximately efficient schedule for $n_T = n$.

Step 7: $n = n+1$. If n_T(Moore) + $n > n_T$(Moore) go to Step 8. Otherwise go to Step 1.

Step 8: Terminate the algorithm.

It is important to note that the $E_{current}$ in the forward search represents the jobs to be non-tardy with the early jobs of the SPT order. However, in backward search

$E_{current}$ represents the jobs to be tardy with the tardy jobs of sequence built by Moore's Algorithm.

E. Genetic Algorithm (GA)

Genetic algorithm finds which jobs should be tardy in the efficient schedule with $n_T = n$, for each n where n_T(SPT) $\geq n \geq n_T$(Moore). GA searches on the subset of the N jobs with cardinality n. GA algorithm uses binary representation, that is, each of these subsets is represented with chromosomes with N genes which have a value 1 or 0. Each gene represents the tardiness state of the corresponding job. For example, if the j^{th} gene has value 1, then the j^{th} job is allowed to be tardy, otherwise the j^{th} job should be non-tardy.

As known, the schedule which gives minimum $\overline{F}$ and keeps the jobs with gene value zero non-tardy, can be found using Smith's Algorithm. Therefore, finding the right chromosome is equivalent to finding the efficient schedule. GA searches on the chromosomes for efficient schedules in the following manner.

Step 0: Initialization of the parameters; $n = n_T$(SPT) – 1, $w = 0$.

Step 1: Establish the initial chromosome population for the efficient schedule with $n_T = n$.

Step 2: Select two chromosomes from the current chromosome population. One of the chromosomes will be chosen randomly while the other one will be determined according to a tournament.

Step 3: Apply crossing-over to the selected chromosomes and generate two new chromosomes. The new chromosomes are added to the population while the worst two existed chromosomes according to fitness function will be removed from the population.

Step 4: Apply mutation to the current population. Then evaluate the population with Smith's Algorithm and record the schedule with least fitness function value as the best schedule.

Step 5: If the best chromosome of the population does not change for 20 consecutive crossovers or after 100 mutations, go to Step 6. Otherwise repeat Steps 2, 3 and 4.

Step 6: State last best schedule that is recorded in Step 4 as the approximately efficient schedule for $n_T = n$.

Step 7: Compare the new best schedule's $\overline{F}$ value with the previous one. If new $\overline{F}$ value is higher than or equal to the old one, $w = w + 1/9$. Otherwise, calculate the percentage improvement and refer to it as α. Set $w = w + factor/9$ where $factor = 0.85^{\alpha}$. If $w \geq 1$, go to Step 8. Otherwise, initialize the statistics considered in Step 5 and go to Step 2.

Step 8: If $n = n_T$(Moore) go to Step 9. Otherwise set $n = n$ -1, $w = 0$ and go to Step 1.

Step 9: Terminate the algorithm.

The Fitness Function

$$Fitness\ Function = w\frac{|n_T(C) - n|}{n_T(SPT) - n_T(Moore)} + (1-w)\frac{F(C) - F(SPT)}{F(Moore) - F(SPT)}$$

This fitness function is quite similar to the one used in Keha and Koksalan (2003). Only difference is that $N_T(C)$ and $F(C)$ are N_T and F obtained by evaluating the chromosome C with Smith's Algorithm. This fitness function is used to determine the worst two chromosomes in the current population and to determine the second parent chromosome for crossing over operations via tournaments.

Initial Population

Keha and Koksalan (2003) presents an algorithm to find initial schedule with $n_T = n$. Their GA starts the search for the efficient schedule with $n_T = n$ at this initial schedule. They refer to this algorithm as *initial heuristic*. We also propose another initial heuristic. For a given problem having $n_T \leq n$ constraint, we assign a job to each position starting from the first position. Job *j* is eligible to be assigned to the current position if scheduling the remaining unassigned jobs according to Moore's Algorithm yields at most n tardy jobs in total. Among the eligible jobs, the one having shortest processing time will be placed to the current location in the schedule.

For creating the initial population of chromosomes, one schedule for each N_T value *n*, *n* -1, *n* +1, and *n* +2 are generated by using Köksalan and Keha(2003)'s initial heuristic and one schedule is generated by using our proposed initial heuristic for each of these n_T values. Eight chromosomes are created to represent tardy jobs of these schedules. Three other chromosomes are created to represent the tardy jobs of the schedules that are generated according to EDD order, SPT order, and Moore's Algorithms.

Five neighbor chromosomes are generated from the chromosome that represents SPT order. Five other neighbor chromosomes are also generated from Moore's Algorithm's representative chromosome. Neighbors are created by changing the values of some genes from 1 to 0 in SPT order case and from 0 to 1 in Moore's Algorithm case. In both cases the total gene values of the neighbor chromosomes will be equal to *n*. Lastly, nine solutions are generated randomly. The initial population consists of all these listed chromosomes.

Crossing Over Operation

Two points crossing over is used in the algorithm. Two genes on the parent chromosomes are selected randomly and the parts of the chromosomes between

these genes are interchanged. As a result of this operation, two new chromosomes are generated. After the crossing over, new offsprings' gene values are randomly increased to 1 or decreased to 0 in order to make the total gene value equal to *n*. This crossing over mechanism is quite similar to the one presented by Keha and Koksalan (2003).

Mutation

Mutation is applied to a randomly selected chromosome in the current population. The selected chromosome's two genes, one with value 1 and the other with value 0 is selected randomly and their gene values are interchanged.

F. Detailed Comparison Tables with Optimum Solution

Table 10-17 present the result of a detailed comparison of each heuristic with the optimum solutions on the test problems. The measures presented in these tables are the same measures that are presented in Table 3 in Chapter 6. The results are presented with respect to each problem size, processing distribution and due date distribution type.

Table 10: Comparison of Heuristics with Optimum Solution on the Test Problems with 20 Jobs and Low Processing Time Variability

Due Date Distribution Type	Performance Measure	Nelson's Heuristic	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
I								
	Average Deviation	0.28%	0%	0%	0.49%	3.37%	0.14%	0.44%
	ND/N_{Total}	1/13	0/13	0/13	3/13	13/13	1/13	5/13
	Max. Deviation	0.2829%	0%	0%	0.61%	10.59%	0.14%	1.13%
	Average CPU Time(millisecond)	19	23.75	37.4	915.4	1342.18	1718.6	1825.36
II								
	Average Deviation	0%	0%	0%	1.02%	5.04%	0%	0.94%
	ND/N_{Total}	0/9	0/9	0/9	2/9	9/9	0/9	3/9
	Max. Deviation	0%	0%	0%	1.55%	7.95%	0%	2.44%
	Average CPU Time(millisecond)	9.6	10	15.8	740.6	1395.86	1415.6 ms	1449.55
III								
	Average Deviation	0%	0%	0%	1.58%	3.62%	0%	1.66%
	ND/N_{Total}	0/6	0/6	0/6	1/6	6/6	0/6	3/6
	Max. Deviation	0%	0%	0%	1.58%	7.47%	0%	4.57%
	Average CPU Time(millisecond)	4	4.5	12.6	609.2	1449.55	1050	1127.43
IV								
	Average Deviation	0%	0%	0%	0.35%	7.25%	0%	0%
	ND/N_{Total}	0/21	0/21	0/21	5/21	21/21	0/21	0/21
	Max. Deviation	0%	0%	0%	0.76%	22.38%	0%	0%
	Average CPU Time(millisecond)	28	21.8	24.8	971.6	1664.29	1515.4	3328.6

Tablo 11: Comparison of Heuristics with Optimum Solution on the Test Problems with 20 Jobs and High Processing Time Variability

Due Date Distribution Type	Performance Measure	Nelson's Heuristic	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
I								
	Average Deviation	0%	0%	0%	1.56%	3.61%	0.11%	0.88%
	ND/N_{Total}	0/16	0/16	0/16	8/16	16/16	2/16	2/16
	Max. Deviation	0%	0%	0%	2.44%	14.26%	0.18%	1.12%
	Average CPU Time(millisecond)	15.6	19	28.2	1203	1181.11	1934.4	2147.48
II								
	Average Deviation	0%	0%	0%	0%	4.02%	0%	0%
	ND/N_{Total}	0/7	0/7	0/7	0/7	7/7	0/7	0/7
	Max. Deviation	0%	0%	0%	0%	15.24%	0%	0%
	Average CPU Time(millisecond)	4	4.75	6.8	653	1476.39	1237.4	1127.43
III								
	Average Deviation	0%	0%	0%	0.26%	5.52%	0%	0%
	ND/N_{Total}	0/7	0/7	0/7	1/7	6/7	0/7	0/7
	Max. Deviation	0%	0%	0%	0.26%	13.11%	0%	0%
	Average CPU Time(millisecond)	4.5	6	13	723	3310.70	1296.5	1275.06
IV								
	Average Deviation	0%	0%	0%	0.74%	6.75%	0%	0.05%
	ND/N_{Total}	0/18	0/18	0/18	5/18	18/18	0/18	2/18
	Max. Deviation	0%	0%	0%	2.51%	35.66%	0%	0.07%
	Average CPU Time(millisecond)	18.8	15.4	22.2	709.4	253374.59	1318.6	2899.1

Table 12: Comparison of Heuristics with Optimum Solution on the Test Problems with 30 Jobs and Low Processing Time Variability

Due Date Distribution Type	Performance Measure	Nelson's Heuristic	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
I								
	Average Deviation	0%	0%	0%	0.60%	3.50%	0.37%	0.12%
	ND/N_{Total}	0/17	0/17	0/17	13/17	16/17	5/17	6/17
	Max. Deviation	0%	0%	0%	1.20%	11.60%	0.84%	0.2817
	Average CPU Time(millisecond)	31.4	38	46.8	3528	4509.71	7262.6	6281.38
II								
	Average Deviation	0%	0%	0%	0.21%	5.24%	0.08%	0.13%
	ND/N_{Total}	0/10	0/10	0/10	8/10	9/10	2/10	2/10
	Max. Deviation	0%	0%	0%	0.35%	19.64%	0.11%	0.13%
	Average CPU Time(millisecond)	8.25	19.5	22	2996	3597.03	7675.5	4898.94
III								
	Average Deviation	0%	0%	0%	0.39%	2.87%	0%	0.64%
	ND/N_{Total}	0/9	0/9	0/9	2/9	8/9	0/9	1/9
	Max. Deviation	0%	0%	0%	0.52%	9.49%	0%	0.64%
	Average CPU Time(millisecond)	7	9.6	9.8	2368.8	4093.64	4722	3704.41
IV								
	Average Deviation	1.95%	0%	0%	0.65%	4.94%	0.31%	0.24%
	ND/N_{Total}	2/28	0/28	0/28	22/28	28/28	6/28	11/28
	Max. Deviation	3.01%	0%	0%	2.14%	15.19%	0.90%	0.60%
	Average CPU Time(millisecond)	34.2	34.4	40.6	3390.4	4509.71	6668.4	10468.98

Table 13: Comparison of Heuristics with Optimum Solution on the Test Problems with 30 Jobs and High Processing Time Variability

Due Date Distribution Type	Performance Measure	Nelson's Heuristic	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
I								
	Average Deviation	0.02%	0%	0%	0.67%	2.29%	0.39%	0.08%
	ND/N_{Total}	1/24	0/24	0/24	17/24	24/24	4/24	6/24
	Max. Deviation	0.02%	0%	0%	3.63%	15.46%	0.76%	0.38%
	Average CPU Time(millisecond)	19	34.6	37.2	4175	3972.84	8215.6	8589.93
II								
	Average Deviation	2.39%	0%	0%	0.09%	3.28%	0%	0.20%
	ND/N_{Total}	1/11	0/11	0/11	7/11	11/11	0/11	1/11
	Max. Deviation	2.39%	0%	0%	0.22%	9.27%	0%	0.20%
	Average CPU Time(millisecond)	10	12.8	12.8	2778.4	4026.53	5484.6	4402.34
III								
	Average Deviation	0.006%	0%	0%	0.017%	4.80%	0%	0%
	ND/N_{Total}	1/10	0/10	0/10	4/10	10/10	0/10	0/10
	Max. Deviation	0.006%	0%	0%	0.049%	18.38%	0%	0%
	Average CPU Time(millisecond)	12.6	12.6	12.8	2556	2791.73	5056.2	4133.9
IV								
	Average Deviation	0%	0%	0%	0.65%	4.94%	0.31%	0.64%
	ND/N_{Total}	0/31	0/31	0/31	27/31	31/31	12/31	2/31
	Max. Deviation	0%	0%	0%	2.14%	15.19%	0.90%	0.97%
	Average CPU Time(millisecond)	31.6	46.75	47	3940.6	5690.83	8253.2	11703.79

Table 14: Comparison of Heuristics with Optimum Solution on the Test Problems with 40 Jobs and Low Processing Time Variability

Due Date Distribution Type	Performance Measure	Nelson's Heuristic	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
I								
	Average Deviation	0.36%	0%	0%	1.00%	2.50%	0.29%	0.97%
	ND/N_{Total}	1/32	0/32	0/32	31/32	31/32	16/32	14/32
	Max. Deviation	0.36%	0%	0%	3.09%	9.49%	0.64%	4.78%
	Average CPU Time(millisecond)	59.4	75	87.2	12690.4	8965.74	29469	24051.82
II								
	Average Deviation	0%	0%	0%	0.18%	2.48%	0.05%	0.06%
	ND/N_{Total}	0/18	0/18	0/18	16/18	17/18	7/18	7/18
	Max. Deviation	0%	0%	0%	0.96%	13.59%	0.17%	0.20%
	Average CPU Time(millisecond)	21.8	28.6	34.8	9225	6871.94	17956.2	14388.14
III								
	Average Deviation	0.06%	0%	0%	0.19%	4.90%	0.06%	0.04%
	ND/N_{Total}	1/17	0/17	0/17	10/17	16/17	2/17	2/17
	Max. Deviation	0.06%	0%	0%	0.34%	15.22%	0.06%	0.06%
	Average CPU Time(millisecond)	12.8	22	37.4	8715.8	6871.94	16419	14173.39
IV								
	Average Deviation	0%	0%	0%	0.45%	7.69%	0%	0.52%
	ND/N_{Total} l	0/21	0/21	0/21	5/21	21/21	0/21	6/21
	Max. Deviation	0%	0%	0%	1.24%	27.12%	0%	1.48%
	Average CPU Time(millisecond)	19.2	22.2	28	4947	4133.90	9931.2	20401.09

Table 15: Comparison of Heuristics with Optimum Solution on the Test Problems with 40 Jobs and High Processing Time Variability

Due Date Distribution Type	Performance Measure	Nelson's Heuristic	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
I								
	Average Deviation	0%	0%	0%	0.79%	1.74%	0.18%	0.41%
	ND/N_{Total}	0/35	0/35	0/35	32/35	35/35	21/35	10/35
	Max. Deviation	0%	0%	0%	3.21%	9.59%	0.64%	1.49%
	Average CPU Time(millisecond)	40.8	71.8	68.8	14012.6	7301.44	27344	27219.36
II								
	Average Deviation	0.88%	0%	0%	0.23%	3.26%	0.13%	0%
	ND/N_{Total}	1/16	0/16	0/16	8/16	16/16	4/16	0/16
	Max. Deviation	0.88%	0%	0%	0.50%	12.59%	0.37%	0%
	Average CPU Time(millisecond)	15.6	25.4	28.2	8065.4	7408.81	16203.2	13421.77
III								
	Average Deviation	0%	0%	0%	0.24%	4.90%	0.02%	0.01%
	ND/N_{Total}	0/14	0/14	0/14	14/14	13/14	1/14	2/14
	Max. Deviation	0%	0%	0%	0.96%	17.69%	0.02%	0.22%
	Average CPU Time(millisecond)	15.6	21.8	22	7181.4	5798.20	14312.4	12079.6
IV								
	Average Deviation	0%	0%	0%	0.25%	7.19%	0%	0%
	ND/N_{Total}	0/23	0/23	0/23	9/23	23/23	0/23	0/23
	Max. Deviation	0%	0%	0%	0.74%	33.49%	0%	0%
	Average CPU Time(millisecond)	21.8	25.2	28.2	4943.6 ms	3704.40 ms	10487.4 ms	22548.5 ms

Table 16: Comparison of Heuristics with Optimum Solution on the Test Problems with 60 Jobs and Low Processing Time Variability

Due Date Distribution Type	Performance Measure	Nelson's Heuristic	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
I								
	Average Deviation	0.82%	0.83%	0.06%	0.87%	2.21%	0.45%	0.36%
	ND/N_{Total}	5/49	3/49	2/49	46/49	47/49	38/49	26/49
	Max. Deviation	2.22%	2.22%	0.07%	2.88%	11.09%	3.35%	4.20%
	Average CPU Time(millisecond)	87.6	187.4	193.8	61093.6	17609.37	123700.2	112098.65
II								
	Average Deviation	0%	0%	0%	0.16%	2.60%	0.05%	0.04%
	ND/N_{Total}	0/23	0/23	0/23	21/23	23/23	8/23	7/23
	Max. Deviation	0%	0%	0%	0.65%	10.23%	0.11%	0.19%
	Average CPU Time(millisecond)	28	65.4	78.2	37503.4	12938.59	75890.6	55727.2
III								
	Average Deviation	0%	0%	0%	0.14%	5.49%	0.03%	0.01%
	ND/N_{Total}	0/14	0/14	0/14	13/14	13/14	5/14	2/14
	Max. Deviation	0%	0%	0%	0.52%	13.89%	0.05%	0.01%
	Average CPU Time(millisecond)	18.8	34.4	31.2	25625	6925.63	51675	36453.54
IV								
	Average Deviation	0%	0%	0%	0.07%	4.67%	0.05%	0.17%
	ND/N_{Total}	0/39	0/39	0/39	20/39	39/39	6/39	14/39
	Max. Deviation	0%	0%	0%	0.31%	26.39%	0.08%	1.00%
	Average CPU Time(millisecond)	56.2	53	52.8	23878.4	9771.05	49984.4	108770.05

Table 17: Comparison of Heuristics with Optimum Solution on the Test Problems with 60 Jobs and High Processing Time Variability

Due Date Distribution Type	Performance Measure	Nelson's Heuristic	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
I								
	Average Deviation	0.36%	0%	0%	0.79%	2.16%	0.27%	1.05%
	ND/N_{Total}	0/52	0/52	0/52	50/52	52/52	40/52	9/52
	Max. Deviation	0.36%	0%	0%	2.52%	13.23%	1.16%	4.09%
	Average CPU Time(millisecond)	78.2	181.2	181.6	64871.8	19703.16	132490.6	119936.96
II								
	Average Deviation	0.62%	0%	0%	0.20%	3.21%	0.04%	0.005%
	ND/N_{Total}	1/21	0/21	0/21	18/21	21/21	9/21	2/21
	Max. Deviation	0.62%	0%	0%	0.65%	13.28%	0.09%	0.005%
	Average CPU Time(millisecond)	21.8	65.8	58.75	34934.2	10898.48	73297	52613.34
III								
	Average Deviation	0.002%	0%	0%	0.10%	3.76%	0.02%	0.005%
	ND/N_{Total}	1/23	0/23	0/23	20/23	23/23	9/23	1/23
	Max. Deviation	0.002%	0%	0%	0.48%	16.88%	0.05%	0.005%
	Average CPU Time(millisecond)	25	65.8	71.8	37443.6	11005.85	74503	60720.10
IV								
	Average Deviation	0%	0%	0%	0.37%	4.48%	0.11%	0.10%
	ND/N_{Total}	0/41	0/41	0/41	32/41	41/41	7/41	10/41
	Max. Deviation	0%	0%	0%	1.23%	26.97%	0.17%	0.48%
	Average CPU Time(millisecond)	46.8	65.6	75	24634.6	9502.61	52834.6	118326.34

G. Detailed Comparison Tables with Nelson's Heuristics

Table 18-23 present the result of a detailed comparison of each heuristic with the Nelson's Heuristic on the test problems with 80, 100 and 150 jobs. The measures that are presented in these tables are the same measures presented in Table 7 in Chapter 6. The results are given with respect to each problem size, processing distribution and due date distribution type.

Table 18: Comparison of Heuristics with Nelson's Heuristic on the Test Problems with 80 Jobs and Low Processing Time Variability

Due Date Distribution Type	Performance Measure	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
I	Average % Difference	0.336%	0.342%	-0.858%	-2.197%	-0.372%	-0.592%
	N+ / N_{Total}	3/64	3/64	1/64	0/64	1/64	2/64
	N- / N_{Total}	0/64	0/64	60/64	59/64	57/64	31/64
	Max % Difference	0.526%	0.526%	0.526%	-0.217%	0.526%	0.359%
	Min % Difference	0%	0%	-2.595%	-10.174%	-1.874%	-4.134%
	Avg. CPU Time(ms)	418.6	450	194484.4	33769.18	382415.8	814379.5
II	Average % Difference	0.006%	0.006%	-0.273%	-1.905%	-0.118%	-0.052%
	N+ / N_{Total}	1/46	1/46	0/46	0/46	0/46	0/46
	N- / N_{Total}	0/46	0/46	44/46	44/46	36/46	31/46
	Max % Difference	0.006%	0.006%	0%	-0.032%	0%	0%
	Min % Difference	0%	0%	-0.693%	-7.815%	-0.460%	-0.467%
	Avg. CPU Time(ms)	290.6	293.6	159650	29474.21	314424.8	427134.5
III	Average % Difference	0.161%	0.161%	-0.225%	-2.041%	-0.795%	0.008%
	N+ / N_{Total}	2/33	2/33	1/33	0/33	2/33	2/33
	N- / N_{Total}	0/33	0/33	29/33	32/33	22/33	15/33
	Max % Difference	0.197%	0.197%	0.151%	-0.055%	0.197%	0.197%
	Min % Difference	0%	0%	-0.777%	-8.513%	-0.540%	-0.065%
	Avg. CPU Time(ms)	215.6	222	127509.6	24964.49	252050.2	273159.9
IV	Average % Difference	0%	0.072%	-0.316%	-5.219%	-0.104%	-0.149%
	N+ / N_{Total}	0/59	1/59	0/59	0/59	0/59	0/59
	N- / N_{Total}	0/59	0/59	46/59	59/59	32/59	36/59
	Max % Difference	0%	0.072%	0.004%	-0.147%	0%	0%
	Min % Difference	0%	0%	-1.013%	-24.485%	-0.276%	-0.881%
	Avg. CPU Time(ms)	134.4	147	77465.4	23407.57	161034.4	706253.6

Table 19: Comparison of Heuristics with Nelson's Heuristic on the Test Problems with 80 Jobs and High Processing Time Variability

Due Date Distribution Type	Performance Measure	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
I	Average % Difference	0.08%	0.08%	-0.983%	-1.731%	-0.371%	-0.753%
	N+ / N_{Total}	5/77	5/77	0/77	0/77	0/77	0/77
	N- / N_{Total}	0/77	0/77	71/77	71/77	65/77	25/77
	Max % Difference	0.24%	0.24%	-0.002%	-0.032%	0%	0%
	Min % Difference	0%	0%	-3.694%	-7.978%	-1.735%	-3.400%
	Avg. CPU Time(ms)	524.8	562.4	226759.4	49392.12	449968.8	1036858.8
II	Average % Difference	0.320%	0.320%	-0.171%	-1.947%	-0.053%	-0.047%
	N+ / N_{Total}	2/35	2/35	1/35	0/35	1/35	0/35
	N- / N_{Total}	0/35	0/35	33/35	33/35	20/35	7/35
	Max % Difference	0.062%	0.062%	0.062%	0%	0.062%	0%
	Min % Difference	0%	0%	-0.510%	-10.282%	-0.232%	-0.124%
	Avg. CPU Time(ms)	231.2	240.4	132050.2	24642.37	261750	688966.4
III	Average % Difference	0.216%	0.216%	-0.087%	-3.089%	0.002%	0.009%
	N+ / N_{Total}	1/27	1/27	1/27	1/27	1/27	1/27
	N- / N_{Total}	0/27	0/27	29/33	26/27	11/27	8/27
	Max % Difference	0.216%	0.216%	0.207%	-0.014%	0.216%	0.216%
	Min % Difference	0%	0%	-0.470%	-12.655%	-0.050%	-0.045%
	Avg. CPU Time(ms)	131.2	153.2	106853	19166.29	212168.8	362012.06
IV	Average % Difference	0%	0%	-0.231%	-4.049%	-0.049%	-0.100%
	N+ / N_{Total}	0/53	0/53	0/53	0/53	0/53	0/53
	N- / N_{Total}	0/53	0/53	44/53	53/53	22/53	18/53
	Max % Difference	0%	0%	-0.002%	-0.067%	0%	0%
	Min % Difference	0%	0%	-0.887%	-23.523%	-0.136%	-0.323%
	Avg. CPU Time(ms)	118.8	140.4	69922	17877.8	144856.2	737177.4

Table 20: Comparison of Heuristics with Nelson's Heuristic on the Test Problems with 100 Jobs and Low Processing Time Variability

Due Date Distribution Type	Performance Measure	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
I	Average % Difference	0.032%	0.032%	-1.068%	-2.468%	-0.555%	-0.647%
	N+ / N_{Total}	6/84	6/84	0/84	0/84	0/84	1/84
	N- / N_{Total}	0/84	0/84	75/84	74/84	73/84	48/84
	Max % Difference	0.070%	0.070%	0%	-0.042%	-0.002%	0.025%
	Min % Difference	0%	0%	-3.163%	-11.341%	-2.377%	-4.456%
	Avg. CPU Time(ms)	906.2	990.4	518509.2	69524.78	1008134.4	814379.4
II	Average % Difference	0.016%	0.016%	-0.305%	-2.151%	-0.091%	-0.055%
	N+ / N_{Total}	2/43	2/43	0/43	0/43	0/43	1/43
	N- / N_{Total}	0/43	0/43	40/43	42/43	35/43	13/45
	Max % Difference	0.025%	0.006%	0%	-11.541%	-0.002%	0.007%
	Min % Difference	0%	0%	-0.002%	-0.071%	-0.341%	-0.313%
	Avg. CPU Time(ms)	471.8	543.8	327906.2	40641.13	640590.6	427134.5
III	Average % Difference	0.014%	0.014%	-0.122%	-2.783%	0.007%	-0.024%
	N+ / N_{Total}	1/26	1/26	1/26	0/26	1/26	0/26
	N- / N_{Total}	0/26	0/26	22/26	25/26	12/26	7/26
	Max % Difference	1.039%	1.039%	0%	-0.047%	0.274%	0%
	Min % Difference	0%	0%	-0.496%	-13.771%	0%	-0.097%
	Avg. CPU Time(ms)	203.2	228	227290.8	18146.23	441028.2	273160
IV	Average % Difference	0.002%	0.002%	-0.143%	-4.268%	-0.033%	-0.089%
	N+ / N_{Total}	1/62	1/62	6/62	0/62	9/62	4/62
	N- / N_{Total}	0/62	0/62	42/62	62/62	25/62	37/62
	Max % Difference	0.002%	0.002%	0.094%	-0.233%	0.149%	0.052%
	Min % Difference	0%	0%	-0.823%	-24.317%	-0.221%	-0.370%
	Avg. CPU Time(ms)	175.2	187.4	161859.4	32641.75	318774.8	706253.6

Table 21: Comparison of Heuristics with Nelson's Heuristic on the Test Problems with 100 Jobs and High Processing Time Variability

Due Date Distribution Type	Performance Measure	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
I	Average % Difference	0.021%	0.017%	-1.164%	-2.659%	-0.580%	-0.519%
	N+ / N_{Total}	4/105	5/105	0/105	0/105	0/105	0/105
	N- / N_{Total}	0/105	0/105	100/105	92/105	96/105	57/105
	Max % Difference	0.053%	0.053%	-0.006%	-0.031%	0%	0%
	Min % Difference	0%	0%	-5.313%	-14.100%	-2.780%	-4.642%
	Avg. CPU Time(ms)	1381.2	1303.2	651368.6	98462.13	1254365.8	1036858.8
II	Average % Difference	0.161%	0.087%	-0.449%	-2.574%	-0.202%	-0.047%
	N+ / N_{Total}	1/69	2/69	1/69	0/69	2/69	1/69
	N- / N_{Total}	0/69	0/69	63/69	62/69	57/69	39/57
	Max % Difference	0.161%	0.161%	0.062%	-0.059%	0.012%	0.116%
	Min % Difference	0%	0%	-1.287%	-10.064%	-0.976%	-0.207%
	Avg. CPU Time(ms)	803.4	772	471637.6	56908.31	906313	688966.4
III	Average % Difference	0%	0%	-0.148%	-3.152%	-0.009%	-0.029%
	N+ / N_{Total}	0/34	0/34	0/34	0/34	0/34	0/34
	N- / N_{Total}	0/34	0/34	33/34	33/34	16/34	14/34
	Max % Difference	0%	0%	-0.005%	-0.069%	0%	0.000%
	Min % Difference	0%	0%	-0.552%	-14.437%	-0.031%	-0.074%
	Avg. CPU Time(ms)	375	321.8	280056.4	32749.13	590328.25	362012
IV	Average % Difference	0%	0%	-0.247%	-4.970%	-0.060%	-0.107%
	N+ / N_{Total}	0/62	0/62	0/62	0/62	0/62	0/62
	N- / N_{Total}	0/62	0/62	48/62	62/62	32/62	35/62
	Max % Difference	0%	0%	0%	-0.076%	0%	0.000%
	Min % Difference	0%	0%	-0.247%	-29.496%	-0.179%	0.458%
	Avg. CPU Time(ms)	206	219	160556.2	36399.85	342664	737177.4

Table 22: Comparison of Heuristics with Nelson's Heuristic on the Test Problems with 150 Jobs and Low Processing Time Variability

Due Date Distribution Type	Performance Measure	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
I	Average % Difference	0.023%	0.023%	-0.129%	-2.968%	-0.621%	-1.084%
	N+ / N_{Total}	2/132	2/132	0/64	0/64	0/132	0/132
	N- / N_{Total}	0/132	0/132	123/132	109/132	125/132	95/132
	Max % Difference	0.023%	0.023%	0%	-0.137%	0%	0%
	Min % Difference	0%	0%	-3.342%	-10.783%	-2.450%	-5.941%
	Avg. CPU Time(ms)	4056.2	4296.8	2978565.4	200199.1	5871646.8	4016009.18
II	Average % Difference	0.013%	0.003%	-0.199%	-2.568%	-0.080%	-0.060%
	N+ / N_{Total}	3/64	3/64	0/64	0/64	0/64	0/64
	N- / N_{Total}	0/64	0/64	63/64	60/64	54/64	43/64
	Max % Difference	0.020%	0.006%	0%	-0.074%	0%	0%
	Min % Difference	0%	0%	-0.497%	-13.894%	-0.485%	-0.346%
	Avg. CPU Time(ms)	2096.8	2234.6	1874315.6	96905.2	3716753	2015413.4
III	Average % Difference	0.335%	0.335%	0.210%	-2.620%	-0.069%	-0.016%
	N+ / N_{Total}	2/59	2/59	1/59	0/59	1/59	1/59
	N- / N_{Total}	0/33	0/33	54/59	57/59	49/59	43/59
	Max % Difference	0.664%	0.664%	0.449%	-0.149%	0.566%	0.614%
	Min % Difference	0%	0%	-0.558%	-12.804%	-0.481%	-0.143%
	Avg. CPU Time(ms)	1531.4	1618.6	1775550	81067.5	3524253.2	1900415.63
IV	Average % Difference	0%	0%	-0.171%	-6.620%	-0.112%	0.126%
	N+ / N_{Total}	1/98	1/98	0/98	0/98	0/98	0/98
	N- / N_{Total}	0/98	0/98	89/98	95/98	81/98	92/98
	Max % Difference	0%	0%	0.000%	-0.322%	0%	0%
	Min % Difference	0%	0%	-0.631%	-34.808%	-0.339%	-0.504%
	Avg. CPU Time(ms)	593.8	712.4	775106	101146.4	1609331.2	3481929.98

Table 23 Comparison of Heuristics with Nelson's Heuristic on the Test Problems with 150 Jobs and High Processing Time Variability

Due Date Distribution Type	Performance Measure	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
I	Average % Difference	0.02%	0.01%	-0.914%	-2.765%	-0.498%	-0.317%
	N+ / N_{Total}	11/149	0/149	0/149	0/149	0/149	3/149
	N- / N_{Total}	11/149	0/149	141/149	104/149	138/149	73/139
	Max % Difference	0.07%	0.03%	0.000%	-0.109%	0%	0.015%
	Min % Difference	0%	0%	-2.749%	-10.508%	-2.811%	-4.249%
	Avg. CPU Time(ms)	5218.6	5187.4	3381065.6	239229.6	6633975.2	4692090.72
II	Average % Difference	0.141%	0.131%	-0.473%	-2.878%	-0.231%	-0.051%
	N+ / N_{Total}	9/106	10/106	1/106	0/106	2/106	2/106
	N- / N_{Total}	0/106	0/106	101/106	89/106	98/106	82/106
	Max % Difference	1.225%	1.225%	1.458%	0%	1.458%	1.438%
	Min % Difference	0%	0%	-1.304%	-11.393%	-0.985%	-0.455%
	Avg. CPU Time(ms)	3487.6	3512.4	2742934.2	173141	5444396.6	3430658.84
III	Average % Difference	0.099%	0.116%	-0.270%	-2.542%	-0.093%	-0.031%
	N+ / N_{Total}	4/74	4/74	1/74	0/74	1/74	1/74
	N- / N_{Total}	0/74	0/74	71/74	69/74	66/74	53/74
	Max % Difference	0.388%	0.455%	0.442%	-0.080%	0.442%	0.232%
	Min % Difference	0%	0%	-0.788%	-12.832%	-0.585%	-0.204%
	Avg. CPU Time(ms)	2071.8	2103	2041987.4	106890.9	4079712.4	2445661.79
IV	Average % Difference	0%	0%	-0.231%	-7.111%	-0.110%	0.130%
	N+ / N_{Total}	0/53	0/53	0/115	0/115	0/115	0/115
	N- / N_{Total}	0/53	0/53	106/115	113/115	92/115	93/115
	Max % Difference	0%	0%	0.000%	-0.146%	0%	0%
	Min % Difference	0%	0%	-0.916%	-38.034%	-0.726%	-0.607%
	Avg. CPU Time(ms)	1084.2	1146.8	891159.4	109414.3	1878893.8	4206759.32