THE TOOL TRANSPORTER MOVEMENTS PROBLEM IN FLEXIBLE MANUFACTURING SYSTEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

FATMA KILINÇ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

APRIL 2005

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Çağlar Güven
Head of Department

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

Prof. Dr. Meral Azizoğlu
Supervisor

Examining Committee Members

Prof. Dr. Ömer Kırca (METU,IE)

Prof. Dr. Meral Azizoğlu (METU,IE)

Assoc. Prof. Dr. Canan Sepil (METU,IE)

Dr. Seçil Savaşaneril (METU,IE)

Asst. Prof. Dr. M. Rüştü Taner (Bilkent Univ., IE)

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Fatma KILINÇ

Signature :

ABSTRACT

THE TOOL TRANSPORTER MOVEMENTS PROBLEM IN FLEXIBLE MANUFACTURING SYSTEMS

KILINÇ, Fatma

M.Sc., Department of Industrial Engineering

Supervisor: Prof. Dr. Meral AZİZOĞLU

April 2005, 125 pages

In this study, we address job sequencing and tool switching problem arising in Flexible Manufacturing Systems. We consider a single machine with limited tool slots on its tool magazine. The available tool slots cannot accommodate all the tools required by all jobs, therefore tool switches between jobs are required. A single tool transporter with limited capacity is used in transporting the tools from the storage area to the machine. Our aim is to minimize the number of tool transporter movements.

We provide two mixed integer linear programming formulations of the problem, one of which is based on the traveling salesman problem. We develop a Branch-and-Bound algorithm powered with various lower and upper bounding techniques for optimal results. In order to obtain good solutions in reasonable times, we propose Beam Search algorithms.

Our computational results reveal the satisfactory performance of the B&B algorithm for moderate sized problems. Moreover, Beam Search techniques perform well for large-sized problems.

Keywords: Tool Transporter, Tool Switching, Branch and Bound, Beam Search

ÖZ

ESNEK İMALAT SİSTEMLERİNDE MAKİNA UCU TAŞIYICISI PROBLEMİ

KILINÇ, Fatma

Yüksek Lisans, Endüstri Mühendisliği Bölümü

Tez Yöneticisi : Prof. Dr. Meral AZİZOĞLU

Nisan 2005, 125 sayfa

Bu çalışmada, esnek imalat sistemlerinde, makine ucu taşıma ve iş çizelgeleme problemi ele alınmıştır. Sınırlı makine ucu barındırma kapasitesi olan tek bir makina için uçları depo alanından taşıyan uç taşıyıcının hareket sayısının en aza indirgenmesi amaçlanmaktadır.

Problemin, birisi Gezgin-Satıcı problemine dayanan, iki ayrı matematiksel modeli verilmiştir. En iyi çözüme ulaşmak için, alt ve üst sınırlama teknikleriyle iyileştirilmiş dal-sınır algoritması önerilmiştir. Kısa sürelerde kaliteli yaklaşık çözümler elde etmek için ise değişik Işın Araştırma sezgisel yöntemleri önerilmiştir.

Deneysel sonuçlar, dal-sınır algoritmasının orta-boyutlu problemler için makul sürelerde çözümler ürettiğini göstermektedir. Büyük boyutlu problemler için Işın Araştırma sezgisel yöntemlerinin başarılı sonuçlar verdiği görülmektedir.

Anahtar Kelimeler: Makine Ucu Taşıyıcısı, Makine Ucu Değiştirilmesi, Dal-Sınır Algoritması, Işın Araştırması

To my family and my love

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to my thesis supervisor, Prof. Dr. Meral Azizoğlu for her unlimited patience, encouragements, guidance and insight throughout this study. Studying with her was really a memorable experience for me.

I would like to express my deepest gratitude to my family who all mean a great deal for me. Thank you all for providing me your love and endless support.

For everything he added to my life, I would thank to my fiancée, Orhan Karzan. Without his understanding, care and moral support such a work will not exist.

I would like to thank all my friends who enriched my life and made me smile during this study and thanks go to especially Özlem Özcan for her valuable company.

I also express my thanks to METU IE assistants and my former assistant and dear friend, Tevhide Altekin for their understandings and kind encouragements. I am especially grateful to my roommate Tolga Han Seyhan for sharing all the burdens of this study and much more.

I would like to thank to my professors, my colleagues and departmental staff for providing me the best environment with their ever-smiling faces and kind encouragements.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Flexible Manufacturing Systems (FMS) are integrated systems of automated material handling devices and computer numerically controlled (CNC) machines that can simultaneously process medium-sized volumes of a variety of part types. They are designed to combine the productivity of the transfer lines and the flexibility of the job shops with the capability of efficiently interchanging tools in their magazines. The flexibility brings so many alternative decisions and as a result the management of an FMS is much more complicated than that of the conventional systems. This is mainly due to the fact that each machine is capable of performing different operations and each part can have a number of alternative routes in the system. Furthermore, high investment requirements necessitate more efficient utilization of the FMSs.

In FMSs the tool management becomes a vital issue for maintaining high productivity due to the limited tool magazine capacities of the machines, in particular if a large number of tools is required. In a typical FMS, the operations can be performed on very highly versatile CNC machines provided that the required tools are available on their tool magazines. As the total number of tools required to process a set of jobs is generally larger than the capacity of the tool magazine, it may be inevitable to switch tools between two successive job processing of the production sequence. Since tool switching operation is usually time consuming and may delay production, the tool switching problem has been recognized extensively in tool management literature. Veeramani et al. (1992) state that in

FMS environments about 16% of the total time is lost due to the unavailability of the tools, and their loading time.

Roh & Kim (1997) state two operating policies for FMSs as *Part Movement Policy* and *Tool Movement Policy*. In *Part Movement Policy*, jobs flow through the system from one machine to another according to their processing and tool requirements. In *Tool Movement Policy*, each job visits only one of the machines and the tools required by the job that are not in the tool magazine are transferred from another machine or from tool storage area and are loaded to the machine before its processing. For a given job sequence, the tool loadings can be determined while optimizing a function of tools loaded. However in FMSs operating with *Tool Movement Policy*, job sequencing becomes another important aspect of the system. Assume that the tools are available in front of the machines and can be loaded immediately when required. This will lead to a classical job sequencing problem. Nonetheless, in FMSs, the tools present in the magazine and the tools required by the job will have a direct effect on the time spent before processing. As the content of the tool magazine is defined by the previously processed jobs, one can conclude that job scheduling with tool considerations is equivalent to a scheduling problem with sequence dependent setup times. The traditional single machine problems aim to find the optimal job sequence. In FMS scheduling, we have additional complexities brought due to the tool management issues.

Majority of the research on FMS scheduling with tool considerations either aim to minimize the total number of tool switches or the tool switching instants or both. However, in practice, the tools are usually carried by an automatic tool transporter with limited capacity and hence minimizing the number of movements of the automatic tool transporter becomes another important concern as well. In these environments, where the tool transportation time between the storage area and tool magazine is significant relative to the processing times, the tool transporter can cause machine idle times. Also, as stated in Crama et al. (1994), if several machines utilize the same tool transporter then the tool transporter can be overloaded. Grieco et al. (1995) showed, in a real case study, that job sequencing considering the tools, reduces the saturation of the tool transporter. As a result, we can state that minimizing the frequency or the number of the tool transporter movements may reduce the time spent due to machine idle times and thus improve the productivity of the FMSs.

The relation of our problem with the FMS literature can be clearly stated as when the capacity of the tool transporter is one, the problem is equivalent to the minimization of the number of tool switches. When the capacity of tool transporter is equal to the tool magazine capacity, the problem reduces to the minimization of the tool switching instants problem. Despite its practical importance, there is only one reported study in the literature that considers the minimization of the frequency of the tool transporter movements. This study is due to Song & Hwang (2002) and presents a heuristic approach.

In this study, we consider the minimization of the tool transporter movements problem in FMS environments. We develop an optimization algorithm to produce exact solutions to our problem with the hope of filling an important gap in the FMS literature. We also design Beam Search techniques for finding approximate solutions for large-sized problem instances.

This thesis consists of seven chapters organized as follows:

In Chapter 2, the problem statement together with underlying assumptions, the notation used throughout the study and two mixed integer linear programming formulations of the problem are provided. The complexity status of the problem and some properties of the formulations are also given.

In Chapter 3, we present the literature on the tool switching problems. We classify the relevant works in the literature according to their objective functions.

In Chapter 4, the details of our solution approach are explained. Several properties of an optimal sequence and the lower and upper bounding mechanisms are presented.

In Chapter 5, we present various Beam Search algorithms to find near optimal solutions, for larger sized problem instances.

In Chapter 6, we discuss the experimental design, generation of the problem parameters and the results of our computational experiments for both Branch-and-Bound and Beam Search algorithms.

In Chapter 7, we discuss the conclusions of the study and directions for future research.

CHAPTER 2

PROBLEM DEFINITION

In this chapter, we present our problem together with its underlying assumptions. We provide two different mathematical programming formulations of the problem and finally discuss the complexity status of the problem.

2.1 PROBLEM STATEMENT

The problem consists of N jobs to be processed on a single flexible CNC machine equipped with a limited tool magazine of C tool slots. Each tool requires exactly one slot and the relative places of the tools in the tool magazine are not important. Each job j requires a set of tools T_j that should be on the magazine before its processing commences. The tool magazine is large enough to hold the tools of each job ($\left|T_j\right| \leq C$) but it is not large enough to hold all tools in T where $T = \bigcup_{j=1}^{N} T_j$. The tools are carried by a tool transporter having a capacity of D tools. Tools to be inserted are carried by the tool transporter whereas the tools taken out of the magazine are carried away with a conveyor or any other carrying mechanism having no restrictive capacity. Our objective is to minimize the number of tool transporter movements required to process all N jobs.

The other assumptions made throughout the study are as listed below:

- The list of the jobs and their tool requirements are completely known in advance, i.e. the system is deterministic.
- All jobs and tools are available at time zero, i.e. the system is static.
- Job preemptions are not allowed and there are no precedence relationships among the jobs.
- Tool magazine and tool transporter can accommodate any combination of tools.
- Tool magazine is initially empty.
- A tool switch occurs when a tool is inserted in the magazine.
- The tool switching process does not occur during the processing of jobs.
- The tool switching time is independent of whether a tool is inserted in an empty place or another tool is removed and the tool is inserted in the previous tool's place.
- The time needed to switch each tool is constant and identical for all tools. Tool switching time is independent of the next job scheduled for processing.
- Tools cannot be changed simultaneously.
- The CNC machine and the tool transporter are always available and never malfunction.
- The planning horizon is short compared to the tool lives; hence tools do not break down or wear out.

2.2 MATHEMATICAL FORMULATION

In this section, two alternative mathematical formulations for the problem are provided. The first formulation is a straightforward extension of the tool switching model developed by Tang and Denardo (1988a). The second one is more involved and based on the Traveling Salesman Problem. The model reduces to the one proposed by Laporte et al. (2004) when $D = 1$.

2.2.1 Mathematical Model 1

The indices, parameters and decision variables used in the first formulation are as follows:

Indices

i: job index

t: tool index

k: order index

Parameters

C: capacity of the tool magazine

D: capacity of the tool transporter

N: total number of jobs to be processed

T_i: set of tools required by job i

T: total number of tools required, i.e. $T = \left| \bigcup_{i=1}^{N} T_i \right|$

J_t: set of jobs requiring tool t

Decision Variables

$$x_{ik} = \begin{cases} 1, & \text{if job } i \text{ is scheduled at position } k \\ 0, & \text{otherwise} \end{cases}$$

$$w_{tk} = \begin{cases} 1, & \text{if tool } t \text{ is in the magazine at position } k \\ 0, & \text{otherwise} \end{cases}$$

$$y_{tk} = \begin{cases} 1, & \text{if tool } t \text{ is inserted in the magazine just before processing job at } k^{\text{th}} \text{position} \\ 0, & \text{otherwise} \end{cases}$$

z_k = total number of tool transporter movements made just before processing job at k^{th} position

Constraints

- Each job is assigned to exactly one position.

$$\sum_{k=1}^{N} x_{ik} = 1 \qquad i = 1, 2, ..., N \tag{2.1}$$

- At each position exactly one job is processed.

$$\sum_{i=1}^{N} x_{ik} = 1 \qquad k = 1, 2, ..., N \tag{2.2}$$

- All tools required by the job are loaded in the magazine just before its processing.

$$\sum_{i \in J_t} x_{ik} \le w_{tk} \qquad t = 1, 2, ..., T \text{ and } k = 1, 2, ..., N \qquad (2.3)$$

- If a tool is loaded in the magazine before processing job at k^{th} position, then a tool switch occurs.

$$w_{tk+1} - w_{tk} \le y_{tk+1} \qquad t = 1, 2, ..., T \text{ and } k = 0, 1, 2, ..., N-1 \qquad (2.4)$$

- Number of tools in the magazine cannot exceed the capacity of tool magazine.

$$\sum_{t=1}^{T} w_{tk} \le C \qquad k = 1, 2, ..., N \qquad (2.5)$$

- Tool transporter movements made before processing job at k^{th} position is determined by the capacity of tool transporter.

$$\sum_{t=1}^{T} y_{tk} \le D * z_k \qquad k = 1, 2, ..., N \qquad (2.6)$$

- Initially the tool magazine of the machine is empty.

$$w_{t0} = 0 \qquad t = 1, 2, ..., T \qquad (2.7)$$

- x_{ik} s, y_{tk} s and w_{tk} s are binary variables and z_k s are nonnegative integer variables.

$$x_{ik} = \{0,1\} \qquad i = 1, 2, ..., N \text{ and } k = 1, 2, ..., N \qquad (2.8)$$

$$y_{tk} = \{0,1\} \qquad t = 1, 2, ..., T \text{ and } k = 1, 2, ..., N \qquad (2.9)$$

$$w_{tk} = \{0,1\} \qquad t = 1, 2, ..., T \text{ and } k = 0, 1, 2, ..., N \qquad (2.10)$$

$$z_k \ge 0 \text{ and integer} \qquad k = 1, 2, ..., N \qquad (2.11)$$

The objective function is to minimize the total number of tool transporter movements required to process all jobs, and it is expressed as follows:

$$\text{Min } \sum_{k=1}^{N} z_k$$

Note that in the above formulation, the binary restrictions on variables y_{tk} and w_{tk} are redundant as they automatically lead to the integer values, once x_{ik} s are binary.

The model is a Mixed Integer Linear Program (MILP). It requires n^2 binary variables for x_{ik} s, n general integer variables for z_k s and $2nt+t$ continuous variables for y_{tk} s and w_{tk} s. Moreover, in this formulation, there are $4n+2nt+t$ functional constraints (n constraints of type (2.1), n constraints of type (2.2), nt constraints of type (2.3), nt constraints of type (2.4), n constraints of type (2.5), n constraints of type (2.6), t constraints of type (2.7)) without the sign and integrality constraints.

We introduce some lower and upper bounds on the optimal values of z_k s, thereby on $\sum_{k=1}^{N} z_k$. Those bounds, when introduced as constraints may help to increase the speed of our MILP solution, by cutting the solution space.

Lower Bounds on Z_k:

Define $l_{ii'} = Max\{0, |T_i \cup T_{i'}| - C\}$ then the number of tool changes between job i and job i' will be at least $l_{ii'}$ and hence the number of tool transporter movements to be made between jobs i and i' will be at least $\left\lceil \frac{l_{ii'}}{D} \right\rceil$. The following constraint states this relationship:

$$z_{k+1} \geq \left\lceil \frac{l_{ii'} * (x_{i'k+1} + x_{ik} - 1)}{D} \right\rceil \qquad i, i' = 1, 2, ..., N \text{ and } k = 1, 2, ..., N-1 \qquad (2.12)$$

In an optimal solution, for a fixed k value, only one of the x_{ik} variables will be one, so the above inequality can be strengthened as follows:

$$z_{k+1} \geq \left\lceil \frac{\sum_{i=1}^{N} \sum_{\substack{i'=1 \\ i' \neq i}}^{N} l_{ii'} * (x_{i'k+1} + x_{ik} - 1)}{D} \right\rceil \qquad i, i' = 1, 2, ..., N \text{ and } k = 1, 2, ..., N-1 \quad (2.13)$$

After processing a job i where $|T_i| = C$, the number of tool switches to be made will be at least $|T_{i'} \setminus T_i|$. The constraint stating this relationship is as follows:

$$z_{k+1} \geq \left\lceil \frac{|T_{i'} \setminus T_i| * (x_{i'k+1} + x_{ik} - 1)}{D} \right\rceil \qquad i, i' = 1, 2, ..., N \text{ and } k = 1, 2, ..., N-1 \quad (2.14)$$

In an optimal solution, for a fixed k value, only one of the x_{ik} variables will be one, so the above inequality can be tightened as follows:

$$z_{k+1} \geq \left\lceil \frac{\sum_{i=1}^{N} \sum_{\substack{i'=1 \\ i' \neq i}}^{N} \left| T_{i'} \setminus T_i \right| * (x_{i'k+1} + x_{ik} - 1)}{D} \right\rceil \quad i, i' = 1, 2, ..., N \text{ and } k = 1, 2, ..., N-1 \quad (2.15)$$

Note that when $|T_i| = C$, then $l_{ii'} = Max\{0, |T_i \cup T_{i'}| - C\} = |T_i \cup T_{i'}| - C = |T_{i'} \setminus T_i|$, which is the same result presented in the second lower bound. Hence the second inequality is a special case of the first one, and therefore (2.15) becomes redundant when (2.13) is used.

<u>Upper Bounds on Zk:</u>

After processing a job i and before processing a job i', there can be at most $C - |T_i \cap T_{i'}|$ tool switches. The following relationship provides an upper bound on the tool transporter movements:

$$z_{k+1} \leq \left\lceil \frac{\left(C - |T_i \cap T_{i'}|\right) * (x_{i'k+1} + x_{ik} - 1)}{D} \right\rceil \quad i, i' = 1, 2, ..., N \text{ and } k = 1, 2, ..., N-1 \quad (2.16)$$

In our model tool transporter movements are done when necessary and no early tool transporter movements are allowed. Therefore, after processing a job i and before processing a job i', at most $|T_{i'} \setminus T_i|$ tool switches can be made. The following relationship provides an upper bound on the tool transporter movements:

$$z_{k+1} \leq \left\lceil \frac{|T_{i'} \setminus T_i| * (x_{i'k+1} + x_{ik} - 1)}{D} \right\rceil \quad i, i' = 1, 2, ..., N \text{ and } k = 1, 2, ..., N-1 \quad (2.17)$$

Note that $|T_{i'} \setminus T_i| = |T_{i'}| - |T_{i'} \cap T_i|$, since $|T_{i'}| \leq C$ then $C - |T_i \cap T_{i'}|$ values will be at least $|T_{i'}| - |T_i \cap T_{i'}|$. Hence the first upper bound inequality always gives bounds greater than or equal to the second one and (2.16) becomes redundant when (2.17) is used.

We also introduce two sets of valid inequalities so as to reduce the solution space bounded by the constraint set of our MILP.

Valid Inequalities:

At any instant either tool t is in the magazine or tool t is inserted in the magazine but both cannot happen at the same time.

$$w_{tk} + y_{tk} \leq 1 \qquad t = 1,2,...,T \text{ and } k = 1,2,...,N \qquad (2.18)$$

The initial tool magazine loading is also considered in the model. Each tool is required by at least one job; hence each tool should be loaded on the tool magazine at least once. This observation leads to the following inequality:

$$\sum_{k=1}^{N} y_{tk} \geq 1 \qquad t = 1,2,...,T \qquad (2.19)$$

Linear Relaxation of Model 1:

The LP-relaxation of this model will always give zero objective value. Clearly in the LP-relaxation the optimal values of the variables are as follows:

$$x_{ik} = \frac{1}{N} \qquad i,k = 1,2,...,N$$

$$w_{tk} = \frac{|J_t|}{N} \qquad t = 1,2,...,T \text{ and } k = 1,2,...,N$$

$$y_{tk} = 0 \qquad t = 1,2,...,T \text{ and } k = 1,2,...,N$$

$$z_k = 0 \qquad k = 1,2,...,N$$

Note that according to the optimal solution of the LP-Relaxation, all lower and upper bounds together with the first set of valid inequalities are redundant. However, when second valid inequality, (2.19) is introduced, objective function value will be positive as $y_{t1} = 1$ for $t = 1,2,...,T$ in the optimal solution. This will lead to a lower bound on the objective function value of $\left\lceil \frac{T}{D} \right\rceil$.

Furthermore, it is known that the number of tool transporter movements required for the initial loading of the magazine is bounded with the following values:

$$\left\lfloor \frac{C}{D} \right\rfloor \le z_1 \le \left\lceil \frac{C}{D} \right\rceil$$

In addition to this, direct addition of lower bounds on the objective function and/or z_k will lead to the equality of the value of the objective function to the maximum of the proposed lower bounds for it. Therefore the lower bounds on the objective function will not provide gains as much as expected in the solution procedure.

2.2.2 Mathematical Model 2

The additional index, parameters and decision variables used in the second formulation are as follows:

Index

j: job index

Parameters

J: set of all jobs to be processed

S: subset of jobs to be processed and job 0

Decision Variables

$$x_{ij} = \begin{cases} 1, & \text{if job } i \text{ is immediately followed by job } j \\ 0, & \text{otherwise} \end{cases}$$

$$y_{it} = \begin{cases} 1, & \text{if tool } t \text{ is in the magazine while processing job } i \\ 0, & \text{otherwise} \end{cases}$$

$$z_{it} = \begin{cases} 1, & \text{if tool } t \text{ is inserted in the magazine just before processing job } i \\ 0, & \text{otherwise} \end{cases}$$

$w_i =$ the number of tool transporter movements made just before processing job i

Constraints

- Exactly one job succeeds each job including the dummy job 0.

$$\sum_{j \in J \cup \{0\} \setminus \{i\}} x_{ij} = 1 \qquad \forall i \in J \cup \{0\} \tag{2.20}$$

- Exactly one job precedes each job including the dummy job 0.

$$\sum_{i \in J \cup \{0\} \setminus \{j\}} x_{ij} = 1 \qquad \forall j \in J \cup \{0\} \tag{2.21}$$

- The connected arcs should not form any sub tours.

$$\sum_{i,j\in S} x_{ij} \le |S|-1 \qquad \forall S \subset J \cup \{0\} \text{ , } 2 \le |S| \le |J|-1 \qquad (2.22)$$

- The number of tools loaded to the tool magazine should not exceed the capacity of the tool magazine.

$$\sum_{t\in T} y_{it} \le C \qquad \forall i \in J \qquad (2.23)$$

- If a tool required by the job being processed is not on the magazine, then it must be inserted and hence a tool switch occurs.

$$x_{ij} + y_{jt} - y_{it} \le z_{jt} + 1 \qquad \forall i \in J \cup \{0\} \text{ , } \forall j \in J \text{ , } \forall t \in T \qquad (2.24)$$

- The total number of tool transporter movements made before processing the i^{th} job will be calculated considering the capacity of tool transporter.

$$\sum_{t\in T} z_{it} \le D * w_i \qquad \forall i \in J \qquad (2.25)$$

- Each tool required by a specific job should be in the magazine during its processing.

$$y_{it} = 1 \qquad \forall i \in J \text{ , } \forall t \in T_i \qquad (2.26)$$

- x_{ij} s and y_{it} s and z_{it} s are binary variables and w_i s are nonnegative integer variables.

$$x_{ij} = \{0,1\} \qquad \forall i \in J \cup \{0\} \text{ , } \qquad \forall j \in J \cup \{0\} \qquad (2.27)$$

$$y_{it} = \{0,1\} \qquad \forall i \in J \text{ , } \qquad \forall t \in T \qquad (2.28)$$

$$z_{it} = \{0,1\} \qquad \forall i \in J \text{ , } \qquad \forall t \in T \qquad (2.29)$$

$$w_i \ge 0 \text{ and integer} \qquad \forall i \in J \qquad (2.30)$$

The objective function minimizes the total number of tool transporter movements made required to process all jobs. It is expressed as follows:

$$\text{Min } \sum_{i\in J} w_i$$

This model requires $n^2 + 2nt + 2n + 1$ binary variables, and n integer variables. The number of constraints is ***exponential*** as there is exponential number of sub-tour elimination constraints.

Similar to the first formulation, we introduce some lower and upper bounds on the optimal values of w_i s, thereby on $\sum_{i \in J} w_i$. Those bounds, when introduced as constraints may help to increase the speed of our MILP solution, by cutting the solution space.

Lower Bounds on W_i :

Define $l_{ij} = Max\{0, |T_i \cup T_j| - C\}$, then the number of tool changes between job i and job j will be at least l_{ij} and hence the number of tool transporter movements to be made between jobs i and j will be at least $\left\lceil \frac{l_{ij}}{D} \right\rceil$. The following constraint states this relationship:

$$w_j \geq \left\lceil \frac{\sum_{i \neq j} l_{ij} * x_{ij}}{D} \right\rceil \qquad \forall j \in J \cup \{0\} \tag{2.31}$$

After processing a job i where $|T_i| = C$, the number of tool switches to be made will be at least $|T_j \setminus T_i|$. The constraint stating this relationship is as follows:

$$w_j \geq \left\lceil \frac{\sum_{i \neq j} |T_j \setminus T_i| * x_{ij}}{D} \right\rceil \qquad \forall i, j \in J \cup \{0\} \tag{2.32}$$

Note that when $|T_i| = C$, then $l_{ij} = Max\{0, |T_i \cup T_j| - C\} = |T_i \cup T_j| - C = |T_j \setminus T_i|$, which is the same result presented in the second lower bound. Hence the second inequality is a special case of the first one and (2.32) becomes redundant when (2.31) is used.

Upper Bounds on W_i :

After processing a job i and before processing a job j, at most $C - |T_i \cap T_j|$ tool switches can be made. The following relationship provides an upper bound on the tool transporter movements:

$$w_j \le \left\lceil \frac{(C - |T_i \cap T_j|) * x_{ij}}{D} \right\rceil \qquad \forall i, j \in J \cup \{0\} \qquad (2.33)$$

Clearly, before processing any job at most C tools can be switched and hence $\left\lceil \frac{C}{D} \right\rceil$ is a valid upper bound on the number of tool transporter movements made before each job, i.e.

$$w_j \le \left\lceil \frac{C}{D} \right\rceil \qquad \forall j \in J \cup \{0\} \qquad (2.34)$$

Considering the "no early tool transporter movement" assumption, the number of tool transporter movements can be bounded as:

$$w_j \le \left\lceil \frac{|T_j|}{D} \right\rceil \qquad \forall j \in J \cup \{0\} \qquad (2.35)$$

Moreover (2.35) can be tightened as follows:

$$w_j \le \left\lceil \frac{|T_j \setminus T_i| * x_{ij}}{D} \right\rceil \qquad \forall i, j \in J \cup \{0\} \qquad (2.36)$$

We also introduce two sets of valid inequalities so as to reduce the solution space bounded by the constraint set of our MILP.

Valid Inequalities:

Note that at any instant either tool t is in the magazine or tool t is inserted in the magazine but both cannot happen at the same time. The following constraint states this fact formally.

$$\sum_{i \in J \setminus \{j\}} x_{ij} + z_{jt} \le 1 \qquad \forall j \in J \text{ and } \forall t \in T \qquad (2.37)$$

Using the idea presented in the first lower bound, the following inequality can be also used in the model:

$$\sum_{t \in T_j} z_{jt} \ge \sum_{i \ne j} l_{ij} * x_{ij} \qquad \forall j \in J \qquad (2.38)$$

Linear Relaxation of Model 2:

The linear relaxation of the second model for a special case when the tool transporter can carry only one tool at a time, $D = 1$, is proved to give nonzero solutions in the article of Laporte et al. (2004). However the only constraint set forcing a nonzero objective in the LP-relaxation is (2.26).

Note that according to the optimal solution of the LP-Relaxation, none of the lower and upper bounds together with the valid inequalities is redundant.

2.3 PROBLEM COMPLEXITY

When the capacity of tool transporter is $D = 1$, that is tool transporter can carry one tool at a time, our problem reduces to the minimization of tool switches. Tang & Denardo (1988a) state that minimization of tool switches on a single machine is NP-Hard in the strong sense through a reduction to well-known Hamiltonian path problem. A more formal NP-hardness proof is presented in Crama et al. (1994) by reduction from the decision problem of the Hamiltonian path in an edge-graph.

Moreover, when the capacity of tool transporter is $D = C$, our problem reduces to the minimization of tool switching instants. Tang & Denardo (1988b) shows that when each tool is required by only one job, the problem reduces to the well known strongly NP-Hard Bin Packing problem.

Note that, the minimization of tool switches ($D = 1$) and the minimization of tool switching instants ($D = C$) are special cases of our problem and they are strongly NP-Hard. So is our problem with arbitrary D.

Our problem can be decomposed into two parts: job sequencing and tool switching. The job sequencing subproblem finds a processing order for each job and the tool switching problem obtains the sequence of tool switches for a given job sequence.

Song and Hwang (2002) prove that for a fixed sequence of jobs, the tool switching problem can be solved optimally by applying ***G**eneralized **K**eep **T**ool **N**eeded **S**oonest (GKTNS)* policy. The stepwise description of the GKTNS policy is given below.

Phase 1 – Just Insertion

1. Set $n = 0$.
2. Select a tool i required by the $(n+1)^{th}$ part, but not considered yet, and a tool r that was last needed and that is on the magazine (at instant n). Break ties arbitrarily.
3. If tool i is not on the tool magazine, remove tool r and insert tool i. Otherwise keep tool i.
4. If all tools required by the $(n+1)^{th}$ part are not considered, go to step 2. Otherwise go to step 5.

Phase 2 – Early Insertion

5. Compute the minimum of empty spaces in the tool transporter and the difference between the tool magazine capacity and number of tools required by the $(n+1)^{th}$ part. Denote this minimum with K. If K is positive, then go to step 6, otherwise, go to step 8.
6. Select a tool e needed earliest after instant n, which is not on the tool magazine and a tool r that was last needed, and that is on the magazine respectively. If tool e is needed before tool r, then remove tool r, insert tool e, set $K = K - 1$ and repeat this step. Else keep tool r, set $K = 0$, and go to step 7.
7. If $n < N - 1$, set $n = n + 1$ and go to step 2.
8. Compute the total number of tool transporter movements and terminate.

CHAPTER 3

LITERATURE SURVEY

In this chapter, we discuss the literature on the minimization of tool transporter movements, tool switches and tool switching instants problems.

The literature on the minimization of the number of tool transporter movements is relatively new and scarce. Nevertheless, there are a number of papers associated with the problem of minimizing the number of tool switches and minimizing the number of tool switching instants on a flexible machine. Note that the first objective is adequate when tool switching time is proportional to the number of tools switched at that instant. Such a case usually occurs when fine tuning is essential during tool loading. The second objective is adequate when tool switching can be made in parallel. Note that when tool transporter capacity is 1, our problem reduces to the minimization of tool switches problem, and when tool transporter capacity is equal to the tool magazine capacity of the machine, C, our problem is equivalent to the minimization of the tool switching instants problem.

We review the literature on the problems of minimization of the tool transporter movements, minimization of the number of tool switches and minimization of the number of tool switching instants, in sections 3.1, 3.2 and 3.3, respectively.

3.1 MINIMIZATION OF THE NUMBER OF TOOL TRANSPORTER MOVEMENTS

To the best of our knowledge, the only published paper considering the minimization of tool transporter movements is the one by Song and Hwang (2002). In this paper, a nonlinear programming formulation of the problem is provided and, it is proven that given a fixed sequence of jobs, ***G**eneralized **K**eep **T**ool **N**eeded **S**oonest (GKTNS)* policy gives the optimal number of tool transporter movements. However, in order to obtain the optimal solution to the problem one needs to determine the optimal sequence of the jobs. In this paper, authors utilize ***M**ultiple **S**tart **G**reedy* heuristic presented in Crama et al. (1994) to find a relatively good job sequence. The results are then compared with the results obtained from the application of ***K**eep **T**ool **N**eeded **S**oonest (KTNS)* policy to the sequence obtained from ***M**ultiple **S**tart **G**reedy* heuristic. They conclude that GKTNS policy outperforms the KTNS rule in minimizing number of tool transporter movements, but the GKTNS policy results in more frequent tool switches. Moreover, the benefits obtained from GKTNS policy become more apparent as tool transporter capacity becomes larger.

3.2 MINIMIZATION OF THE NUMBER OF TOOL SWITCHES

There are several studies that address the number of tool switches problem on a single machine. In this section, we first review the papers proposing heuristic approaches, then the optimization paper and finally the extensions to the problem. The most noteworthy of studies that consider the model with these assumptions are due to Tang and Denardo (1988a), Bard (1988), Crama et al. (1994) and Laporte et al. (2004). Also some variants of the basic model are discussed by Privault and Finke (1995), Fathi and Barnette (2002), Hertz and Widmer (1996), Tzur and Altman (2004) and Rupe and Kuo (1997).

The common assumptions used in the papers addressing single machine are as follows:

- Tools are kept in a tool storage area and each machine has its own automatic tool interchanging device.
- The set of jobs to be processed and the tools required by each job are known in advance.

- Each tool occupies only one tool slot in the tool magazine.
- Start-up and shutdown times are taken as constant and hence ignored.
- It is assumed that no more time is needed to remove a tool t and insert a tool t' at the same instant then at different instants.

The first paper stating the minimization of tool switches on a flexible machine is written by Tang and Denardo (1988a). As a joint work, their second paper considers the minimization of the number of switching instants as another performance measure.

In Tang and Denardo (1988a), a linear programming formulation of the problem together with numerous tightening mechanisms is provided. They show that the solution of the problem consists of two decisions: finding the order in which the jobs are processed (job sequencing problem) and finding the tools to be switched before processing of each job (tool replacement problem). In their paper an NP-Hardness proof of the problem is also stated. Furthermore, they propose and prove that the KTNS policy gives the optimal solution to the tool replacement problem for a fixed job sequence. They propose a 3-step heuristic called ***G**reedy* ***P**erturbation* procedure for job scheduling problem and test their algorithm with four cases where there are $N = 10, 20, 30, 40$ jobs; $T = 10, 15, 25, 30$ tools with $C = 4, 8, 10, 15$ corresponding tool magazine capacities. Paper concludes that ***G**reedy* ***P**erturbation* procedure is efficient when $N = 10, 20, 30$, but suffers from the time requirements when N is large.

The second paper dealing with the tool switches problem is due to Bard (1988). A nonlinear integer programming formulation of the problem is presented together with a dual-based relaxation heuristic. The heuristic utilizes Lagrangean relaxation to decompose the problem into two sub problems where one can be solved by Hungarian method and the other by Backward Dynamic Programming procedure. Due to the presence of a significant duality gap, the authors suggest a single pass heuristic ensuring a local optimum solution. Although the global convergence of the algorithm is not assured, in almost all cases tested, it finds the global optima. They also discuss a possible extension of the problem to the multiple machines where the job sequence is fixed among all machines with the objective of minimizing the maximum tool switches over all machines.

Crama et al. (1994) consider the same problem and prove that the problem is NP-Hard even when the tool magazine capacity is two through the reduction to the decision problem of the Hamiltonian path in an edge graph. Moreover, they provide a new proof of the correctness of KTNS policy by verifying that KTNS policy gives an optimal solution to the corresponding linear programming model of a tool loading problem for a fixed job sequence. The links between the minimization of the tool switches problem and some other well-known combinatorial optimization problems such as Matrix Permutation, Greedy Constraint Matrices Optimization, Block Minimization and Interval Matrix Recognition, are established in their paper. Due to the polynomial time solvability of the tooling problem, they focus on job sequencing problem and propose six heuristics falling into two main categories as construction and improvement. ***S**hortest **E**dge*, ***N**earest **N**eighbor*, ***F**arthest **I**nsertion* heuristics are applied to solve the corresponding Traveling Salesman Problem (TSP) problem with arc lengths corresponding to the lower bounds found as $l_{ij} = max\left(\left|T_i \cup T_j\right| - C, 0\right)$. They also propose and test two block minimization heuristics, namely, ***N**earest **N**eighbor **B**lock **M**inimization* and ***F**arthest **I**nsertion **B**lock **M**inimization* and a ***S**imple **G**reedy*, and ***M**ulti-**S**tart **G**reedy* heuristics together with ***I**nterval* heuristic. They further improve the performances of the construction heuristics by applying ***R**estricted* and/or ***G**lobal 2-opt* improvement procedures and ***L**oad-and-**O**ptimize* strategy. Their experimental results on several problem instances reveal that TSP-based heuristics perform well on dense instances, when the tool requirements are high; ***M**ulti-**S**tart **G**reedy* heuristic and ***G**lobal 2-opt* are the best despite their running time while ***S**imple **G**reedy* and ***F**arthest **I**nsertion **B**lock **M**inimization* are good competitors. TSP-based ***F**arthest **I**nsertion* is mentioned to perform noteworthy for dense instances.

Hertz et al. (1998) offer new heuristics that take into account the global view of the entire solution in addition to the interactions among two jobs at a time by redefinition of distance functions and a more holistic TSP-based approach. They provide four additional distance definitions over the well-known l_{ij} definition. Except the first one, which provides an upper bound, no definition states a real relationship between the tool switches among jobs, so they can neither be used as a lower bound nor as an upper bound. They have tested the effects of these distance functions with ***F**arthest **I**nsertion 1* and *2*, ***G**ENI,* and ***G**ENIUS*. To further improve the results, they propose to use the KTNS policy in the insertion decision. The extensive tests on the problem instances generated similar to the ones in Crama et al. (1994) suggest that some strategies yield very fast results and some others produce the best known results in the literature.

Djelab et al. (2000) formulate the problem using hypergraphs. Based on their formulation, they propose a new heuristic approach for the job sequencing problem, namely ***I**terative **B**est **I**nsertion* procedure, which assures a local optimum. They use the KTNS policy to obtain the corresponding tool sequence. They compare the new heuristic with Crama et al.'s (1994) heuristics, and conclude that their heuristic outperforms Crama et al. (1994)'s in terms of both computation time and solution quality. Moreover, their experimentational results show that the performance of their heuristic is not affected from the sparsity of the job-tool matrix.

Shirazi and Frizelle (2001) conduct an empirical study on the issue of tool switching. They gather 19 data sets from 7 different companies that employ high technology manufacturing processes. In the study, the efficiency of the currently employed methods in these companies is assessed. Then these methods are compared with the results of 6 different heuristics from the literature; ***S**imple **G**reedy*, ***M**ultiple **S**tart **G**reedy*, ***B**est **P**osition **I**nsertion*, ***S**hortest **E**dge*, ***F**arthest **I**nsertion with **l**ower **b**ounds*, and ***F**arthest **I**nsertion with **u**pper **b**ounds*. They conclude that the heuristics consistently outperform the current methods used within the companies. Their computational results also show that ***M**ultiple **S**tart **G**reedy* is the most robust heuristic and ***B**est **P**osition **I**nsertion* and ***F**arthest **I**nsertion with **u**pper **b**ounds* perform sound as well.

Al-Fawzan and Al-Sultan (2002) propose a ***T**abu **S**earch* algorithm. They test their algorithm with random data sets generated as in Tang and Denardo (1988a). The effects of the different strategies (random swapping and random block insertion methods combined in strategic/probabilistic oscillation in generating neighborhood structure, recency based memory for short term memory, frequency based memory for long term memory) used in ***T**abu **S**earch* are also compared. The authors state that the results obtained from the ***T**abu **S**earch* are very adequate and the effects of the long term memory structure and strategic/probabilistic oscillation are significant on the performance of the algorithm.

Zhou, Xi and Cao (2004) introduce a ***B**eam **S**earch* based algorithm equipped with simple priority rules. They compare their results with the ones obtained from Bard's (1988). Additional tests carried out on random instances indicate that their heuristic achieve sound results in terms of computational efficiency and solution quality.

Laporte et al. (2004) consider the same problem and suggest a new mixed integer linear programming formulation based on TSP. Examining the linear programming relaxations of the formulations lead to the fact that LP-relaxation of TSP-based formulation always dominates the model developed by Tang and Denardo (1988a). Consequently they propose several valid inequalities and lifting procedures for the TSP-based formulation. For optimal solutions, an LP-based Branch-and-Cut (B&C) algorithm and a Branch-and-Bound (B&B) algorithm equipped with two lower bounding and one upper bounding scheme are suggested. The lower bounds used in the B&B setting, include a simple lower bound and a Minimum Spanning Tree relaxation utilizing the usual lower bound definition for TSP. Their experimental results reveal that the B&B algorithm is more effective in terms of solution times and is capable of solving the problems with up to 25 jobs.

We categorize the studies addressing the extensions of the problem according to the machine environments. In the subsequent paragraphs, we first discuss studies addressing parallel machine environments and then the multi-stage environments.

Khan et al. (2000) and Fathi & Barnette (2002) consider the tool switching issues on parallel identical machines. The problem on identical machines consists of assigning jobs to machines, sequencing jobs on each machine and obtaining the tool switching plan for each machine with the objective of minimizing makespan. Furthermore, they assume no tool sharing among different machines.

Khan et al. (2000) consider the case of two parallel identical machines with the objective of minimizing makespan. They assume that each job requires its own set of tools and the tool switching time varies for different tools. They propose a heuristic procedure and test it on a single real-life industry data.

Fathi and Barnette (2002) address the problem of job scheduling with specified processing times and tool requirements on parallel identical machines. They assume that each machine has its complete set of tools, all machines have the same tool magazine capacity, and, tool switching time is identical for all tools on all machines. They demonstrate that the problem is NP-hard in strong sense. They utilize ***M****ulti*-***S****tart* ***G****reedy* heuristic for sequencing jobs on each machine and KTNS policy for obtaining respective tool switching plan on each machine. They propose 3 different heuristics for assigning jobs to machines: ***M****ulti*-***S****tart* ***L****ocal* ***I****mprovement* procedure, a variant of well-known list scheduling method, and a ***C**onstructive **A**pproach*, which is an adaptation of ***M**ulti-**S**tart **G**reedy* heuristic for k-TSP. They compare the results of these heuristics with a proposed lower bound on random instances generated as in Crama et al. (1994) and on special structured problems. Their experimental results reveal that ***M**ulti-**S**tart **L**ocal **I**mprovement* approach and ***C**onstructive **A**pproach* perform best.

Widmer (1991) and Hertz & Widmer (1996) study an extension of the problem to a job shop environment with the objective of minimizing makespan. They further include restrictions on the due date and limited production period in their models and propose ***T**abu **S**earch* approaches to their problems. Widmer (1991) tests the performance of the ***T**abu **S**earch* algorithm against various parameters. Hertz and Widmer (1996) present an adaptation of the ***T**abu **S**earch* algorithm together with its improvement procedures. They conclude that their algorithm is very robust and on the average it quickly produces solutions better than the best known solutions of the benchmark problems in Widmer (1991) and than on the problem sets used in the job-shop literature by modifying to include tooling constraints.

We finally discuss the literature on single machine with certain extensions including non-uniform tool sizes, and different tool and slot requirements.

Privault and Finke (1995) address the same problem under non-uniform tool switching times. They assume there are weights corresponding to the switching time required to remove tool i and insert tool j. They reduce the problem to the one of finding a minimum cost flow of maximum value in an acyclic network. They argue that their method is an alternative for the KTNS policy and runs in $O(n^2)$ time. Hence it is more efficient than the KTNS policy when the number of tools is much greater than the number of jobs, which is usually the case encountered in practice. In the second part of their paper, two groups of heuristics are presented for job sequencing. The first group heuristics first construct the job sequence and then find the tool requirements (including ***F**arthest **I**nsertion using* l_{ij} *values*, *"**S**uper **T**ask"* model and ***B**est **I**nsertion* method). Heuristics listed in the second group, construct sequence and manage tools simultaneously, and these include ***N**ext **B**est* method and ***P**art* method, an adaptation of an online partitioning algorithm to the ***N**ext **B**est* method. Their experimental results reveal that *"**S**uper **T**ask"* and ***P**art* method are promising for solution speed and quality respectively.

Rupe and Kuo (1997) consider a variation of the problem, where tool requirements of the jobs can be more than the tool magazine capacity and the jobs can be split into two or more pieces. A nonlinear mathematical formulation is provided and it is proven that given a job sequence, the KTNS policy with job splitting will result in optimal tool replacement policy. Moreover by allowing concurrent job and tool changing, a policy called "***G**et **T**ool **N**eeded **S**oonest*" is developed and compared with the KTNS policy using simulation. Their experimental results show that ***G**et **T**ool **N**eeded **S**oonest* policy results in less tool switches, however at an expense of greater computation times.

Tzur and Altman (2004) consider the extension where the slot requirement for each tool differs; each tool can occupy more than one slot of the tool magazine. To solve this problem, three decisions related with job sequencing, tool switching, and slot assignment have to be made. An integer programming formulation is presented and NP-Hardness of the problem is settled. Due to the discouraging results obtained from the general optimization software in solving the model, they focus on heuristic methods. Their heuristics use ***K**eep **S**maller **T**ools **N**eeded **S**oonest (KSTNS)* policy in place of the KTNS policy in finding the tool loading for a given job sequence. In constructing a job sequence, they employ modified versions of ***M**ultiple **S**tart **G**reedy* heuristic, G*lobal 2-opt* and *GENIUS*. For the physical placement of tools in the magazine, they develop ***B**lock **S**ubmersion **P**rocedure*, which tries to minimize the slot interchanges of tools. Finally, they propose a heuristic named ***A**laddin*, which simultaneously considers three decisions. The performances of the algorithms are tested on similar problem sets of Hertz et al. (1998) and ***A**laddin* turns out to produce the best results over the modified heuristics from the literature.

3.3 MINIMIZATION OF THE TOOL SWITCHING INSTANTS

Minimization of the tool switching instants on a flexible machine is first considered by Tang & Denardo (1988b), in the second part of their companion paper. The manufacturing environment is identical to the environment described in the first part of their paper, but the performance criterion is set to the minimization of the total number of tool switching instants. When each tool is required by at most one job, they show that the problem reduces to the classical bin packing problem. The classical bin packing problem is strongly NP-Hard (see Garey and Johnson, 1979), so is their problem with additional complexity of arbitrary tool requirements. They provide the properties of the optimal solution and develop a Branch-and-Bound scheme for solving the problem. Additionally, based on the set partitioning idea, they generate ***S**weeping **P**rocedure* as a lower bound on the objective value of the partial solution. ***M**aximal **I**ntersection **M**inimal **U**nion* procedure, analogous to ***F**irst **F**it **D**ecreasing* rule for bin packing, is developed to compute an upper bound at each node of the tree. Their computation results indicate that B&B procedure is efficient for problems up to 30 jobs. They discuss an extension of the scheme, to a series of flexible machines with different tool magazine capacities.

Denizel (2003) considers the same problem and provides an integer programming formulation of the problem. She proposes a B&B scheme that utilizes a lower bounding procedure based on Lagrangean decomposition. The lower bounding scheme is based on the idea for determining infeasibility of the problem given a fixed number of switching instants. She discusses a multiplier adjustment scheme for LR-decomposition. Furthermore, she obtains upper bounds at each node with heuristically modifying the partial solution. She compares performance of the developed procedure against the ones available in the literature and concludes that her procedure performs better on the average and is effective in solving large size problems, with up to 30 jobs and 30 tools.

3.4 MULTI-OBJECTIVE APPROACHES

A few papers address the tool switching problem with two objectives. The objectives are the minimization of the number of tool switches and the minimization of the number of tool switching instants. Keung, Ip and Lee (2001) suggest a genetic algorithm for a real-life FMS under investigation, Denford system, with the aim of simultaneous minimization of both objectives. The objectives are represented in a nonlinear form and are simply added. Their genetic algorithm is tested on a specific instance and effects of initial population size and number of generations are assessed. The results are then compared with random search strategy and finally, authors conclude that for this single instance, their algorithm produces high quality solutions very quickly.

In another multi-objective study by Keung, Ip and Lee (2001), the environment is extended to multiple parallel machines, where the tool magazine capacity of each machine is different and tool sharing is allowed. A nonlinear integer programming formulation of the problem is stated and similar to their companion study, a genetic algorithm is proposed to find local optimum solutions. They test their algorithm with simulation and conclude that their algorithm produces high quality robust solutions.

CHAPTER 4

SOLUTION APPROACH

Recall that our problem is strongly NP-hard. This fact leads one to focus on implicit enumeration techniques such as Branch and Bound (B&B) and Dynamic Programming to find optimal solutions in reasonable times. In this study, we propose a B&B approach that employs some optimality properties and several bounding schemes. In this chapter, we first present some properties of the optimal solution and then various lower and upper bounds. We illustrate our bounding schemes on numerical examples and finally describe our B&B scheme.

4.1 PROPERTIES OF THE OPTIMAL SOLUTION

In this subsection, we present our optimality properties that are utilized to reduce the size of the search. We also give the extensions of the properties to the partial schedules.

Theorem 1:

If job i requires completely different tools than all other jobs, then it can be processed at the last position.

***Proof*:**

Suppose S is an optimal schedule in which job $i \left| T_i \cap \left(\bigcup_{j \neq i} T_j \right) = \emptyset \right.$ is processed between jobs $j-1$ and j, if $\frac{|T_i|}{D}$ is integer then, we can remove job i from its current position and put it to the last position of the schedule without increasing the number of tool transporter movements (changing optimality). Assume that $\frac{|T_i|}{D}$ is not integer. If we reschedule job i to the last position, the tool transporter movements made before processing job $j-1$ will remain same. In this case, the empty slots in the tool transporter of its movement before processing job $j-1$, if any, can be filled with the tool requirements of job j not job i. Likewise, all of the early tool insertions can be shifted and an equal amount of tool slots can be made available at the last position for job i. Thus removing job i from its current position and inserting to the last position will not increase the tool transporter movements. □

Assume i_p is the last scheduled job in a partial schedule. Let Q be the set of unscheduled jobs.

We now discuss the extension of ***Theorem 1*** to a partial schedule with $T_{i_p} = C$. In such a case, if there exists job j in Q, that requires completely different tools than all other jobs in Q and job i_p, i.e., ($T_j \cap \left(T_{i_p} \cup \left(\bigcup_{\substack{i \in Q \\ i \neq j}} T_i \right) \right) = \emptyset$), then there exists an optimal schedule that schedules job j as the last job among all jobs in Q.

There can be a case where a single job satisfying the conditions of ***Theorem 1*** may not exists, however a subset of jobs, say SS, where $SS = \left\{ j \left| T_j \cap \left(T_{i_p} \cup \left(\bigcup_{\substack{i \in Q \\ i \notin SS}} T_i \right) \right) = \emptyset \right. \right\}$ that satisfies the conditions may exist. If $T_{i_p} = C$ and a subset, SS, exists then the jobs in SS can be processed at the last positions consecutively without increasing the number of the tool transporter movements.

Theorem 2:

Assume the jobs can be partitioned into r subsets according to their tool requirements in such a way that $\left(\bigcup_{i\in r_v} T_i\right)\cap\left(\bigcup_{i\in r_y} T_i\right)=\varnothing$ for all subsets r_v and r_y. Then there exists an optimal schedule in which the jobs in each subset are processed consecutively.

Proof:

Suppose S is a schedule which satisfies the conditions of the above theorem, i.e. the jobs in set r_v are processed together before all jobs in set r_y which are processed together as well. Let x' denote the additional number of tools loaded to the magazine before processing of job $i \in r_v$ and x be the number of tools required by job i which are present in the magazine just before its processing. Similarly define y and y' for job $j \in r_y$. Let S' be the schedule obtained by interchanging jobs i and j without altering any other job sequence. The jobs that are sequenced before job i in S will not be affected from this interchange and the jobs that are sequenced after job j in S will never lead to higher number of tool switches as they may share common tools with job j but not job i. Moreover such an interchange cannot decrease the total number of tool switches associated with jobs i and j and the jobs sequenced between them. Note that the processing of job j between the jobs in set r_v will require exactly $y+y'$ additional tool switches before job j as all the tools of job j will be introduced first. Furthermore, the processing of job i between the jobs of set r_y will require at most $x+x'$ additional tool switches before job i. Some tools of job i may be introduced during the processing of jobs sequenced between j and i in schedule S' leading to less additional tool switches before job i. So the total number of tool switches for jobs i and j, and the jobs sequenced between them in schedule S' will never be smaller than that of schedule S. Hence schedule S satisfying the conditions of ***Theorem 2*** can never lead to higher tool switches, thereby tool transporter movements, than those of any other schedule violating the conditions. □

Once job i_p using exactly C tools is scheduled, then the distinct subsets in ***Theorem 2*** should be redefined only considering the unscheduled jobs and job i_p.

Theorem 3:

Let $A_j = \left\{ i \in Q \mid T_i \subseteq T_j \right\}$*, then there exists an optimal solution that processes all jobs in* A_j *consecutively, immediately after job* j*.*

Proof:

Suppose S is an optimal schedule in which any job i in A_j is processed before job j. Remove job i from its position in S and replace it immediately after job j without altering any other job or tool sequence. Note that such an interchange does not increase the number of tool switches as job i uses a subset of tools that are used by job j. Hence the new schedule that sequences job i just after job j is optimal as well. □

We can extend ***Theorem 3*** to a partial schedule as follows:

Let SP be a subset of Q that contains all jobs that can be processed after job i_p without any tool switches, i.e. $SP = \left\{ j \mid j \in Q, T_j \subseteq T_{i_p} \right\}$. Then all jobs in SP can be processed just after job i_p in any order.

For any partial sequence, as the order of unscheduled jobs is not known, we cannot know the content of the tool magazine exactly. In such a case, the inserted tools are known, but the tools removed from the magazine are not known. Now assume that we know which tools are in the magazine. Such a case can only occur either before the tool magazine gets full at the first positions of the sequence, or the last scheduled job requires exactly C tools. The following theorem generalizes ***Theorem 3*** for the known tool magazine content case.

Theorem 4:

If all the tools required by job j *are already on the magazine, then job* j *can be added just after job* i_p*.*

Proof:

Suppose S is an optimal schedule in which job j is sequenced after job i_p and we know the tools on the magazine when i_p is completed. Assume job j is removed from its current position and inserted just after job i_p without altering any other job or tool sequence. Note that such an interchange cannot increase the number of tool switches as all tools required by job j are already in the magazine. (Hence the new schedule that sequences job j just after job i_p is optimal as well.) □

4.2 LOWER BOUNDING PROCEDURES

Four lower bounding schemes to be used in our B & B approach are discussed in this section. The first two lower bounds are based on the special structure of the problem whereas the last two are based on Traveling Salesman Problem (TSP) analogy. Traveling Salesman problem is finding the shortest Hamiltonian cycle in a given graph with fixed arc lengths. Hence, our problem is related to finding the shortest path on the graph which has variable arc lengths.

4.2.1 Lower Bound 1, LB_1:

The number of tools required by the unscheduled jobs which are not in the tool magazine is $\left|\bigcup_{j\in Q} T_j \setminus T_{i_p}\right|$ where i_p is the last scheduled job in a partial schedule and Q is the set of unscheduled jobs as defined before. When the number of the tools which are required by unscheduled jobs and present in the tool magazine is subtracted from $\left|\bigcup_{j\in Q} T_j \setminus T_{i_p}\right|$, the resulting expression provides a lower bound on the number of the tool switches to be made by the jobs in Q. The number of tools which are required by unscheduled jobs and present in the tool magazine is not known exactly, however it can be overestimated by $\min\{C, number\_of\_tools\_on\_the\_tool\_magazine\} - \left|T_{i_p}\right|$. Clearly replacing the number of free slots by its corresponding overestimate (upper bound) does not violate the validity of the proposed lower bound. Hence a valid lower bound on the number of the tool switches is

$$\left|\bigcup_{j\in Q} T_j \setminus T_{i_p}\right| - \left(\min\{C, number\_of\_tools\_on\_the\_tool\_magazine\} - \left|T_{i_p}\right|\right).$$

Any lower bound on the number of tool switches gives a lower bound on the number of tool transporter movements after being divided by the capacity of the tool transporter and rounded up to the smallest integer. Therefore, the subsequent expression is a valid lower bound on the number of tool transporter movements.

$$LB_1 = \left\lceil \frac{\left|\bigcup_{j\in Q} T_j \setminus T_{i_p}\right| - \left(\min\left\{C, number\_of\_tools\_on\_the\_tool\_magazine\right\} - \left|T_{i_p}\right|\right)}{D} \right\rceil$$

$$LB_1 = \left\lceil \frac{\left|T_{i_p} \cup \bigcup_{j\in Q} T_j\right| - \left|T_{i_p}\right| - \min\left\{C, number\_of\_tools\_on\_the\_tool\_magazine\right\} + \left|T_{i_p}\right|}{D} \right\rceil$$

$$LB_1 = \left\lceil \frac{\left|T_{i_p} \cup \bigcup_{j\in Q} T_j\right| - \min\left\{C, number\_of\_tools\_on\_the\_tool\_magazine\right\}}{D} \right\rceil \tag{4.1}$$

The complexity of finding LB_1 is $O(n)$ as the union of the tool sets of $O(n)$ jobs is taken and a union operation requires constant time. Note that when no jobs are scheduled, LB_1 reduces to $\lceil T/D \rceil$, which is a trivial underestimate of the optimal number of tool transporter movements.

4.2.2 Lower Bound 2, LB_2:

The number of tools which are not required by already scheduled jobs or by the jobs scheduled after a job, say job j such that $\left|T_j\right| = C$ minus the number of empty slots of the last tool transporter movement is a lower bound on the number of the tool switches required for the unscheduled jobs. Dividing this value by the tool transporter capacity and rounding up to the smallest integer will provide the subsequent lower bound:

$$LB_2 = \left\lceil \frac{\left|\bigcup_{t\in R} t\right| - x}{D} \right\rceil \tag{4.2}$$

where R is the set of tools that arrived as a first time either at the first node of the Branch and Bound tree or after sequencing a job, say job j such that $|T_j| = C$, and

x is the number of the tools which are only required by the unsequenced jobs and carried in the last tool transporter movement.

Note that given a partial schedule, the number of tools to be allocated can be found by the GKTNS policy. The tools to be carried depend on the sequence; however the number of the tool slots to be allocated to the tools required by the unsequenced jobs is sequence independent.

The complexity of this bound is $O(n)$ as the union of the tool sets of $O(n)$ unsequenced jobs are taken and a union operation requires constant time. Note that when no jobs are scheduled, LB_2 reduces to $\lceil T/D \rceil$, which is a trivial underestimate of the optimal number of tool transporter movements.

Though LB_1 and LB_2 are identical at the root node, they give different values at intermediate nodes and neither of the bounds is dominant, as can be observed from the following example.

Numerical Example 1:

In this example, the number of jobs is 5 ($N = 5$) and the number of available tools is 10 ($T = 10$). The capacity of the tool magazine is taken as 5 ($C = 5$) and the capacity of tool transporter is set to 3 ($D = 3$). The following table presents the job-tool requirement matrix for this problem instance.

Table 4.1 The Job-Tool Requirement Matrix for the Numerical Example 1

Jobs \ Tools	1	2	3	4	5	6	7	8	9	10
1	1	1	1							
2			1	1	1					
3					1	1	1			
4							1	1		
5									1	1

At the root node LB_1 and LB_2 are both $\lceil T/D \rceil = \lceil 10/3 \rceil = 4$.

For each partial sequence generated at the first level, we find the following values for LB_1 and LB_2.

Partial Sequence	LB_1	LB_2
{1}	$\lceil (10-\min\{5,3\})/3 \rceil = 3^{*}$	$\lceil (7-0)/3 \rceil = 3^{**}$
{2}	$\lceil (10-\min\{5,3\})/3 \rceil = 3$	$\lceil (7-0)/3 \rceil = 3$
{3}	$\lceil (10-\min\{5,3\})/3 \rceil = 3$	$\lceil (7-0)/3 \rceil = 3$
{4}	$\lceil (10-\min\{5,2\})/3 \rceil = 3$	$\lceil (8-1)/3 \rceil = 3$
{5}	$\lceil (10-\min\{5,2\})/3 \rceil = 3$	$\lceil (8-1)/3 \rceil = 3$

*For partial sequence $\{1\}$, $T_{i_p} = \{1\}$, $Q = \{2,3,4,5\}$

**For partial sequence $\{1\}$, $R = \{4,5,6,7,8,9,10\}$ and $x = 3-3 = 0$

As can be seen from the above figures, LB_1 and LB_2 do not lead to any differentiation at the first level. However, when we move to the second level from partial sequence $\{1\}$, the following partial sequences and the associated lower bounds are obtained:

Partial Sequence	LB_1	LB_2
{1,2}	$\lceil (8-\min\{5,5\})/3 \rceil = 1^{*}$	$\lceil (5-1)/3 \rceil = 2^{**}$
{1,3}	$\lceil (7-\min\{5,6\})/3 \rceil = 1$	$\lceil (4-0)/3 \rceil = 2$
{1,4}	$\lceil (7-\min\{5,5\})/3 \rceil = 1$	$\lceil (5-1)/3 \rceil = 2$
{1,5}	$\lceil (7-\min\{5,5\})/3 \rceil = 1$	$\lceil (5-1)/3 \rceil = 2$

*For partial sequence $\{1,2\}$, $T_{i_p} = \{2\}$, $Q = \{3,4,5\}$

**For partial sequence $\{1,2\}$, $R = \{6,7,8,9,10\}$ and $x = 3-2 = 1$

As can be seen from the above table, at the second level, LB_2 dominates LB_1 for all partial sequences.

In the same example, if the tool magazine capacity is taken as 3, and the tool transporter capacity is set to 2 and we obtain $LB_1 = LB_2 = \lceil T/D \rceil = \lceil 10/2 \rceil = 5$ at the root node, and for each partial sequence generated at the first level, we find LB_1 and LB_2 values as:

Partial Sequence	LB_1	LB_2
{1}	$\lceil(10-\min\{3,3\})/2\rceil = 4^*$	$\lceil(7-1)/2\rceil = 3^{**}$
{2}	$\lceil(10-\min\{3,3\})/2\rceil = 4$	$\lceil(7-1)/2\rceil = 3$
{3}	$\lceil(10-\min\{3,3\})/2\rceil = 4$	$\lceil(7-1)/2\rceil = 3$
{4}	$\lceil(10-\min\{3,2\})/2\rceil = 4$	$\lceil(8-0)/2\rceil = 4$
{5}	$\lceil(10-\min\{3,2\})/2\rceil = 4$	$\lceil(8-0)/2\rceil = 4$

*For partial sequence $\{1\}$, $T_{i_p} = \{1\}$, $Q = \{2,3,4,5\}$

**For partial sequence $\{1\}$, $R = \{4,5,6,7,8,9,10\}$ and $x = 2-1 = 1$

As can be seen from above values, for all partial sequences LB_1 dominates LB_2.

These examples illustrate that neither LB_1 nor LB_2 is dominant, hence they should be used together in an enumeration approach.

Lower Bounds Based on Hamiltonian Path

When the number of tools required by each job is equal to the tool magazine capacity, i.e. $|T_i| = C \quad \forall i$, and the capacity of the tool transporter is 1, i.e. $D = 1$, our problem reduces to the sequence dependent makespan problem, where the setup time between job i and j is $\left\lceil \frac{|T_j \setminus T_i|}{D} \right\rceil$. The sequence dependent set-up time makespan problem is equivalent to finding a Hamiltonian path with specified arc lengths. (See, Garey and Johnson, 1979)

In our case, the arc lengths are variable as the tool switches are dependent on a sequence of the prior jobs not only to the immediate job. However, if we use lower bounds in place of the actual number of tool switches required by the pair of jobs when processed sequentially, the optimal solution of the resulting Hamiltonian path problem gives a lower bound on the number of tool transporter movements. In doing so, if we let l_{ij} denote a valid lower bound on the number of tool switches required between two consecutive jobs i and j, then $l'_{ij} = \left\lceil \frac{l_{ij}}{D} \right\rceil$ is a valid lower bound on the number of tool transporter movements required between two consecutive jobs i and j.

The number of the tool switches between jobs i and j can be underestimated by $Max\{0, |T_i \cup T_j| - C\}$, therefore $l'_{ij} = \left\lceil \frac{Max\{0, |T_i \cup T_j| - C\}}{D} \right\rceil$ serves as a valid lower bound on the length of arc connecting job i and j. These lower estimates are also symmetric. The following figure shows the network representation of the Hamiltonian path problem.

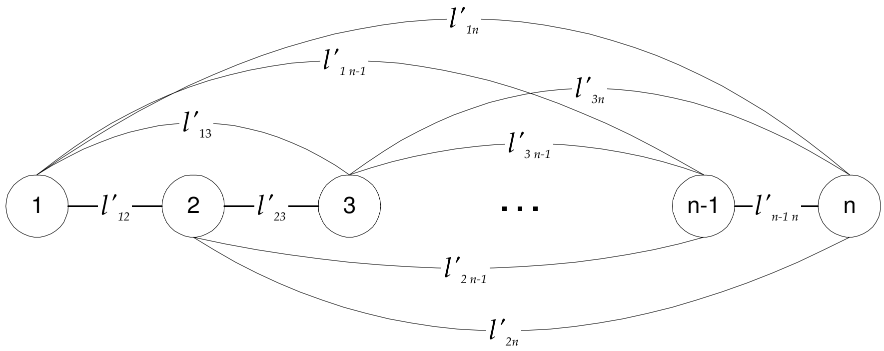

Figure 4.1 Network Representation of l'_{ij} Underestimates

The cost of the optimal Hamiltonian path where the arc lengths are underestimates is a lower bound for our problem. Nevertheless the problem of finding an optimal Hamiltonian path is strongly NP-Hard. (Garey and Johnson, 1979) En route to find a polynomial-time lower bound with underestimates of the arc lengths, we prefer to use lower bounds for the Hamiltonian path in place of optimal solutions. Two such lower bounds used in our study are explained below.

4.2.3 Lower Bound 3, LB_3:

LB_3 recognizes the fact that in an optimal TSP tour, there will be always two arcs connected to each node and for each node calculating the average of the first and second minimum costs of the arcs of that node and summing these averages will be a valid lower bound for TSP. (see Reeves (1995), Chapter 7)

In our case, we use $\left\lceil \frac{Max\{0, |T_i \cup T_j| - C\}}{D} \right\rceil$ as arc lengths.

Note that for a sequencing problem, there is only one arc departing from the first job and one arc arriving to the last job. Consequently, for these two jobs, only the first minimums should be considered. Assume job i_p and i_n are scheduled as the first and last jobs, respectively. For each unscheduled job except i_n, compute the average of the minimum of two arc costs connecting the specified node and all unscheduled jobs and i_p. For the last scheduled job i_n, the minimum cost arc should be found among the arcs of unscheduled jobs but not job i_p. The collection of the averages and the minimum arc length connected to i_p and i_n divided by two provides a valid lower bound.

In computing LB_3 for a partial sequence, the first job, i_p, is known, as it is the last scheduled job of the partial sequence, however the last job is not. Thus, for the last position, we consider each unscheduled job of the partial sequence. The minimum of the lower bounds over all unscheduled jobs considered at the last position provides a valid lower bound on the number of tool transporter movements.

The following procedure gives the stepwise description of LB_3 at the root node.

Procedure 1: *A procedure to find LB_3 at the root node*

Step 0. Calculate the $\lceil l_{ij}/D \rceil$ values for all possible job pairs, (i, j).

Step 1. For each job i, find the first minimum ($M_1(i)$) and second minimum ($M_2(i)$) of $\lceil l_{ij}/D \rceil$ values among the unscheduled jobs, j. Define $M_1^*(j) = \underset{\substack{i \in Q \\ i \neq j}}{Min}\left\{\left\lceil \frac{l_{ij}}{D} \right\rceil\right\}$ as the minimum cost of arc connecting job j to one of the unscheduled jobs.

Step 2. For each (i, j) pair, where job i is the first job and job j is the last job, $LB_3^1(i, j)$ is found as follows:

$$LB_3^1(i, j) = \frac{M_1(i) + M_1^*(j) + \sum_{k \neq i, k \neq j} M_1(k) + M_2(k)}{2} \tag{4.3}$$

Step 3. $LB_3 = \underset{\forall (i,j)}{Min}\left\{LB_3^1(i, j)\right\}$.

The following algorithm briefly discusses the extension of LB_3 to a partial sequence where i_p is the last job in the partial sequence and Q is the set of unscheduled jobs:

Procedure 2: *A procedure to find* LB_3 *for a partial sequence*

Step 0. For each unscheduled job $i \in Q$, find the first minimum ($M_1(i)$) and second minimum ($M_2(i)$) of $\lceil l_{ij}/D \rceil$ values over all $j \in \{Q \cup i_p\}$.

Step 1. Let job $i \in Q$ be the last job. Compute $M_1^*(i)$.

Step 2. Compute ($LB_3^1(i_p, i)$) using equation (4.3) for all $i \in Q$.

Step 3. The lower bound, $LB_3(i_p)$, is $\underset{i \in Q}{Min}\{LB_3^1(i_p, i)\}$.

Given the last job, the lower bound is found in $O(\log n)$ steps, which is the complexity of finding two minimums. Note that the operations can be done in constant time for each job, given the initial order of the arc lengths. There are $(n - |Q|)$ candidates for the last job. These operations are done by considering each $O(n)$ unscheduled jobs at the last position, so the overall complexity of this lower bound is $O(n \log n)$.

Alternatively, $LB_3(i_p)$ can be found by only deducing the maximum arc length among the set of second minimums. This change will decrease the computational burden of the calculations of $LB_3(i_p)$ however at an expense of lower quality. The following procedure states this alternative implementation.

Procedure 3: *A simplified procedure to find* LB_3 *for a partial sequence*

Step 0. For each unscheduled job $i \in Q$, find the first minimum ($M_1(i)$) and second minimum ($M_2(i)$) of $\lceil l_{ij}/D \rceil$ values over all $j \in \{Q \cup i_p\}$.

Step 1. $$LB_3^2(i_p) = \frac{M_1(i_p) - \underset{k \neq i_p}{Max}\{M_2(k)\} + \sum_{k \neq i_p,} M_1(k) + M_2(k)}{2}$$

The computational complexity of procedure 3 is $O(\log n)$ as finding two minimums in step 0 requires order of $O(\log n)$ computations and step 1 requires constant time.

The alternative implementations of $LB_3(i_p)$ is illustrated on the following example.

Numerical Example 2:

We consider 6 jobs with the following $\lceil l_{ij}/D \rceil$ matrix.

	$\lceil l_{ij}/D \rceil$					
	1	**2**	**3**	**4**	**5**	**6**
1	---	2	3	2	2	1
2	2	---	3	1	0	5
3	3	3	---	2	2	5
4	2	1	2	---	2	5
5	2	0	2	2	---	5
6	1	5	5	5	5	---

We compute LB_3 for $Q = \{2,3,4,5,6\}$ and $i_p = 1$.

Step 0. The first and second minimums of jobs are presented below.

Job i	$M_1(i)$	$M_2(i)$
1	1	2
2	0	1
3	2	2
4	1	2
5	0	2
6	1	5

Step1. The $M_1^*(i)$ values for each job $i \in Q$ are found as 0, 2, 1, 0 and 5, respectively.

Step 2. Each unscheduled job is considered in the last position and we calculate the resulting lower bound values.

$$LB_3^1(1,2) = \frac{M_1(2) + M_1^*(1) + \sum_{k\neq 1, k\neq 2} M_1(k) + M_2(k)}{2}$$

$$LB_3^1(1,2) = \frac{1+0+\{(2+2)+(1+2)+(0+2)+(1+5)\}}{2} = 8$$

$$LB_3^1(1,3) = \frac{1+2+\{(0+1)+(1+2)+(0+2)+(1+5)\}}{2} = 7.5$$

$$LB_3^1(1,4) = \frac{1+1+\{(0+1)+(2+2)+(0+2)+(1+5)\}}{2} = 7.5$$

$$LB_3^1(1,5) = \frac{1+0+\{(0+1)+(2+2)+(1+2)+(1+5)\}}{2} = 7.5$$

$$LB_3^1(1,6) = \frac{1+5+\{(0+1)+(2+2)+(1+2)+(0+2)\}}{2} = 8$$

Step 3. Minimum of $LB_3^1(i, j)$ values is $LB_3 = 7.5$.

We now compute LB_3^2 for the same partial sequence as follow:

$$LB_3^2(1) = \frac{M_1(1) - \underset{k\neq 1}{Max}\{M_2(k)\} + \sum_{k\neq 1,} M_1(k) + M_2(k)}{2}$$

$$LB_3^2(2) = \frac{1-5+\left(\{0+1\}+\{2+2\}+\{1+2\}+\{0+2\}+\{1+5\}\right)}{2} = 6$$

This example shows that LB_3^2 is dominated by LB_3^1. Note that the computational complexity of LB_3^2 is lower. In our experiments, we prefer to use LB_3^2 as in majority of the test instances we found that LB_3^2 is equal to LB_3^1 .

4.2.4 Lower Bound 4, LB_4:

LB_4 is based on the Minimum Spanning Tree (MST) idea. A Minimum Spanning Tree is a tree which connects all nodes using the minimum total arc lengths. The degree of a node is the number of arcs connected to that node. In a TSP solution, the degree of each node is exactly 2, whereas in an MST solution, the degrees can be greater than or equal to 1. Therefore an MST solution may not lead to a feasible path between job pairs; however, its solution provides a lower bound on the optimal cost of the TSP solution.

Based on our cost underestimates defined as $\lceil l_{ij}/D \rceil$, a minimum spanning tree for the unscheduled jobs $(i \in Q)$ can be constructed and its cost serves as a lower bound for our problem. The MST should connect to the scheduled job set through the last scheduled job i_p, so an i_p-spanning tree using our cost underestimates is a valid lower bound on the optimal cost of the unscheduled jobs. Below procedure gives the stepwise description of the lower bound.

Procedure 4: *A procedure to find* LB_4 *for a partial sequence*

Step 0. Set $n = 0$.

Step 1. Sort the arcs in Q in nondecreasing order of their $\lceil l_{ij}/D \rceil$ values.

Step 2. Find the first arc in the list that does not form a cycle in the current graph, add the arc to the graph, set $n = n+1$.

Step 3. If $n = |Q| - 1$, go to step 4. Else go to step 2.

Step 4. Find the arc with smallest $\lceil l_{i_p j}/D \rceil$ value, for some $j \in Q$. Connect i_p to the graph with this arc.

Step 5. The cost of the graph is the lower bound.

The complexity of this bound is $O(n^3)$. Kruskal's algorithm is implemented to compute the MST in $O(n^3)$ steps, since at cycle detection phase; the list of all arcs in the graph is kept and compared with the arc at hand.

We illustrate LB_4 through the following numerical example taken from Tang and Denardo (1988a).

Numerical Example 3:

The job-tool requirement matrix is provided in Table 4.2.

Table 4.2 The Job-Tool Requirement Matrix for the Numerical Example 3

Jobs \ Tools	1	2	3	4	5	6	7	8	9
1	1			1				1	1
2	1		1		1				
3		1				1	1	1	
4	1				1		1		1
5			1		1			1	
6	1	1		1					
7							1		
8						1			
9			1						
10					1		1		

The problem can be reduced by the application of ***Theorem 3***. Note that job 7 requires only tool 7 and hence its tool set is a subset of the tool sets of jobs 3, 4, and 10. Therefore we can eliminate job 7 and schedule it just after one of these jobs. Similarly job 8 can be scheduled just after job 3, and job 9 can be inserted just after either job 2 or job 5 without incurring additional tool switches and hence tool transporter movements. Note that the tools required by job 10 is a subset of tools required by job 4, hence it can be scheduled just after job 4. For the remaining jobs the job-tool requirement matrix is provided below in Table 4.3.

Table 4.3 The Revised Job-Tool Requirement Matrix for the Numerical Example 3

Jobs \ Tools	1	2	3	4	5	6	7	8	9
1	1			1				1	1
2	1		1		1				
3		1				1	1	1	
4	1				1		1		1
5			1		1			1	
6	1	1		1					

We set tool magazine capacity to 4 ($C = 4$), and tool transporter capacity to 2 ($D = 2$). Considering the tool requirements of jobs, $\lceil l_{ij}/D \rceil$ values are presented in the below matrix and the network representation is given in Figure 4.2.

$\lceil l_{ij}/D \rceil$

	1	2	3	4	5	6
1	---	1	2	1	1	1
2	1	---	2	1	0	1
3	2	2	---	2	1	1
4	1	1	2	---	1	1
5	1	0	1	1	---	1
6	1	1	1	1	1	---

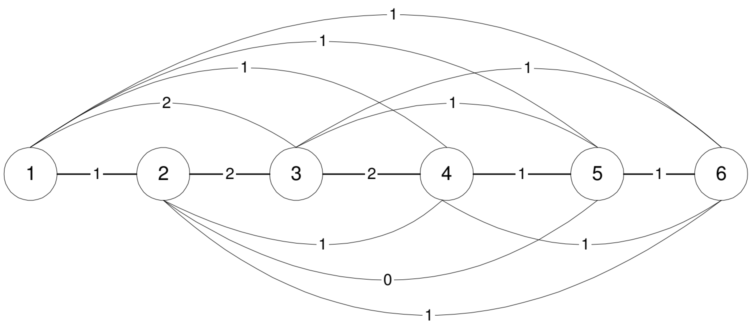

Figure 4.2 Network Representation of l'_{ij} Underestimates for Numerical Example 3

At the root node, the arcs in increasing order of $\lceil l_{ij}/D \rceil$ values are as follows: { (2,5), (1,2), (1,4), (1,5), (1,6), (2,4), (2,6), (3,5), (3,6), (4,5), (4,6), (5,6), (1,3), (2,3), (3,4) }. In MST computations, the arcs are added to the graph in the following order: { (2,5), (1,2), (1,4), (1,6), (3,5) }. Lower bound 4 at the root node is 4.

For partial sequence $\{5, 2\}$, LB_4 is found by applying the following steps:

Note that $Q = \{1, 3, 4, 6\}$ and $i_p = 2$ for this partial sequence.

Step 0. $n = 0$.

Step 1. The sorted list of arcs in increasing order of their $\lceil l_{ij}/D \rceil$ values is { (1,4), (1,6), (3,6), (4,6), (1,3), (3,4) }

Step 2. Arc (1,4) does not form a cycle with the arcs of the graph and therefore it is added to the graph, and removed from the list. $n = 1$.

Step 3. $n = 1 < |Q| - 1$, go to step 2.

Step 2. Arc (1,6) does not form a cycle with the arcs of the graph, therefore it is added to the graph and removed from the list. $n = 2$.

Step 3. $n = 2 < |Q| - 1$, go to step 2.

Step 2. Arc (3,6) does not form a cycle therefore is added to the graph and removed from the list. $n = 3$.

Step 3. $n = 3 = |Q| - 1$, go to step 4.

Step 4. For all $j \in Q$ the list of arcs in their increasing order of $\lceil l_{i_p j} / D \rceil$ values is { (2,1), (2,4), (2,6), (2,3) }. So we connect $i_p = 2$ to the graph via arc (2,4).

Step 5. The resulting MST is shown in Figure 4.3.

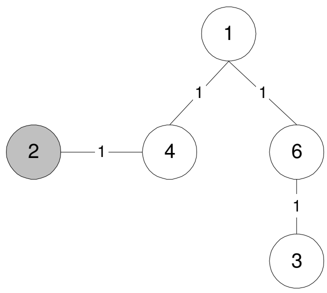

Figure 4.3 Minimum Spanning Tree Representation of Numerical Example 3

The cost of the tree, i.e. LB_4 is 4 for this partial sequence.

When all cost estimates are zero for a particular unscheduled job, the optimal cost of MST is zero, and the i_p-spanning tree gives the minimum arc cost between i_p and one of the unscheduled jobs. In such a case, the use of lower bound 3 becomes more significant.

4.3 UPPER BOUNDING PROCEDURES

Our upper bounding procedure used to find an initial feasible solution in our B&B procedure follows two steps:

Step 1. Finding a job sequence

Step 2. Finding an optimal tool transporter movement sequence to the job sequence found in *Step 1* by applying GKTNS rule.

In *Step 1*, we use two orders:

Order 1. Natural order, i.e. 1, 2, …, n.

Order 2. Order formed by Greedy heuristic that sequences the job requiring the largest number of tools to the first position. In case of ties, the job that requires the most frequent tools is selected. In order to determine the job with most frequent tools, each tool is weighted by the number of jobs it is required by and these weights are summed.

After a job is fixed to position 1, the job having maximum number of common tools with this job is positioned at 2. In case of ties, the job minimizing the number of its tools and the tools of the previous job is preferred. After second position is filled, the other positions are allocated similarly. The heuristic terminates when all jobs are sequenced.

At each node of the search tree, we employ the upper bound in two ways.

First the greedy heuristic is employed to the unscheduled jobs and the GKTNS rule is applied to the sequence of the greedy heuristic to get the number of tool transporter movements. This procedure is named as ***Upper Bound 1***, UB_1.

In order to have a computationally cheaper upper bound, in place of GKTNS rule, we use simple cost estimates, $c_{ij} = \left\lceil \frac{|T_j \setminus T_i|}{D} \right\rceil$ when job i precedes job j on the sequence. Note that the number of the tool switches is limited by the number of the tools required by the current job but not present in the magazine, i.e. $|T_j \setminus T_i|$. Hence the actual number of tool transporter movements is no bigger than $c_{ij} = \left\lceil \frac{|T_j \setminus T_i|}{D} \right\rceil$.

The ***Upper Bound 2***, UB_2, simply adds these c_{ij} estimates over all consecutive job pairs of the greedy heuristic's sequence.

In our preliminary experiments, it is observed that UB_2, although computationally very cheap, did not lead to significant improvements in terms of the number of nodes over the case with no upper bounds.

We illustrate the upper bounding schemes through the numerical example presented for LB_4 taken from Tang and Denardo (1988a).

Numerical Example 3 (Continued) :

In order to find a feasible solution as the incumbent solution at the root node, the upper bounds are computed for two separate sequences; natural order and order formed by the greedy heuristic. For this example, we apply the greedy heuristic at the root node as follows:

Jobs 1, 3, and 4 require 4 tools and jobs 2, 5, and 6 require 3 tools. Since there is a tie, we weight each tool with the number of jobs required by, as; 1 (4), 2 (2), 3 (2), 4 (2), 5 (3), 6 (1), 7 (2), 8 (3), 9 (2). The total frequency weights of jobs 1, 3, 4 are found as 11 (=4+2+3+2), 8 (=2+1+2+3), and 11 (=4+3+2+2) respectively. Still there is a tie between jobs 1 and 4, so we arbitrary select job 1 to the first position. The unscheduled jobs 2, 3, 4, 5, and 6 have respectively 1, 1, 2, 1, and 2 common tools with job 1. Since jobs 4 and 6 create a tie for the second position, we look at the total number of tools required by these jobs and job 1. As the number of tools required by jobs 1 and 4 is 6 and the number of tools required by jobs 1 and 6 is 5, job 6 is scheduled at the second position. Similarly the other positions are filled and the sequence is found as { 1, 6, 2, 5, 3, 4 }.

<u>Upper Bound 1:</u>

The first upper bound for the sequences { 1, 2, 3, 4, 5, 6 } and { 1, 6, 2, 5, 3, 4 } are found by simply applying the GKTNS policy. The resulting costs of these sequences are found as 9 and 8 respectively.

<u>Upper Bound 2:</u>

We compute the $c_{ij} = \left\lceil \frac{|T_j \setminus T_i|}{D} \right\rceil$ values at the root node.

c_{ij}

	1	2	3	4	5	6
1	---	1	2	1	1	1
2	2	---	2	1	1	1
3	2	2	---	2	1	1
4	1	1	2	---	1	1
5	2	1	2	2	---	2
6	1	1	2	2	2	---

For the sequence { 1, 2, 3, 4, 5, 6 } and for the sequence { 1, 6, 2, 5, 3, 4 }, UB_2 values are obtained by the sums $c_{12}+c_{23}+c_{34}+c_{45}+c_{56}=1+2+2+1+2=8$ and $c_{16}+c_{62}+c_{25}+c_{53}+c_{34}=1+1+1+2+2=7$, respectively.

4.4 BRANCH-AND-BOUND PROCEDURE

Recall that our problem reduces to a sequencing problem, as the optimal tool loading and transporter movement sequence can be found by the GKTNS rule for a given job sequence. We develop a Branch-and-Bound algorithm to implicitly enumerate all possible $n!$ alternative sequences.

A node at level l of our B & B tree finds the job of the l^{th} position of the sequence. For each node at level l, there are $(n-l)$ branches emanating; each branch considers an addition of an unsequenced job to the position l.

The cost of the partial schedule is found by applying the GKTNS rule.

As branch selection rule at any level, when there is a tie in terms of the partial solution cost and the estimated lower bounds, the solution which has a smaller cost value is selected to continue the search. For the cases when the costs are also equal, the branching is based on the sequence found by the greedy heuristic.

Our algorithm uses a “depth first search with the left most strategy” due to its relatively low memory requirements and likelihood of achieving high quality upper bounds at earlier branches.

We fathom a node, if its addition as the last job to the current partial sequence violates the decomposition rule stated in ***Theorem 2***. Accordingly, if the last job, i_p, belongs to a decomposed set, say r_v, then only the nodes corresponding to the jobs in r_v are considered for further branching. If there is no unscheduled job in r_v, then a branch is created for all the remaining unscheduled jobs regardless of their decomposition sets.

When the job being scheduled requires exactly C tools, the decomposition sets are updated for the remaining unscheduled jobs.

According to ***Theorem 4***, job i should be assigned to the current partial schedule, if it requires a subset of tools currently present in the magazine. Note that unless there are less than C tools in the magazine or the last sequenced job requires exactly C tools, we cannot know the tools in the tool magazine.

We also fathom a node if the cost of the partial schedule found by the GKTNS rule is no smaller than the incumbent cost. Whenever the next job being scheduled on a branch requires a subset of tools of the previous job, no further branching is made and all other branches created at that level are removed from the search tree. However, we never generate a job whose tool set is a subset of any other job in our experimentation. Hence ***Theorem 3*** cannot find its application.

For each unfathomed node, a lower bound is calculated by our lower bounding procedures sequentially as follows:

We first calculate LB_1, if the total cost of the partial solution plus LB_1 on the unscheduled jobs is no greater than the incumbent cost, we calculate LB_2. If LB_2 added to the partial solution cost is no greater than the incumbent cost, we calculate LB_3. For each node that cannot be fathomed by LB_3, we first calculate an upper bound on the i_p-MST by calculating the total of the costs of arcs ($\lceil l_{ij}/D \rceil$) from the first unscheduled job found by the greedy heuristic to all other unscheduled jobs and i_p. Only if this upper bound turns out to be nonzero, we calculate LB_4, i.e. i_p-MST based lower bound. Note that Kruskal's algorithm adds arcs to the tree while preserving the previously added arcs. So the partial solutions at intermediate iterations give valid lower bounds. We terminate the generation of a complete i_p-MST solution, once we observe that the cost of the partial i_p-MST tree plus the cost of the partial solution is no smaller than the incumbent cost. Note that we calculate lower bounds in the order of their complexities and only when they are required.

We also update the overall lower bound computed at a node as $Max\left\{\left\lceil \frac{T}{D} \right\rceil, LB_{parent\_node} + Cost_{parent\_node}\right\} - Cost_{node}$ value.

Two upper bounds, UB_1 and UB_2 are found at the root node. Among these, the one with lower objective value is selected for branching at the root node. After making extensive experimentation, we decide to use upper bounds partially in such a way that upper bounds are calculated only for the nodes appearing at the first levels of the tree. When there are $N = 10, 15$ jobs, we compute UB_1 at the first $N/5$ levels, and we compute no upper bounds at higher levels. When $N > 15$, we compute the upper bounds only at the first 3 levels.

We fathom a node when the cost of the associated partial sequence and lower bound is no smaller than the incumbent objective value.

The incumbent solution is updated in either of the following cases:

- A complete solution with a better objective function value is found.
- An upper bound on the unscheduled jobs plus the cost of the current partial schedule is smaller than the incumbent solution.

CHAPTER 5

BEAM SEARCH

Branch & Bound algorithms usually suffer from extensive computational time requirements, as a large number of nodes needs to be evaluated. Even for a single machine sequencing problem with n jobs, the number of nodes generated at level k will be $\frac{n!}{(n-k)!}$. In the absence of any lower or upper bounds and reduction mechanisms a total of $\sum_{k=1}^{n} \frac{n!}{(n-k)!}$ nodes and $n!$ leaf nodes are searched. Beam Search Techniques are proposed to reduce the number of nodes evaluated in a Branch & Bound (B&B) tree, thereby leading quicker solutions, however with no guarantee of optimality.

Beam Search is an adaptation of B &B that obtains a solution by partially searching the promising nodes of the B & B tree. It is a type of Breadth First Search where, at each level, the heuristic estimates are used to select fixed number of promising nodes for further branching and the remaining nodes at that level are disregarded permanently. The number of nodes retained at each level, β, is called the "Beam Width" of the search.

Evaluating each node to obtain an estimate for the potential of its offspring may be time consuming. To dispel this difficulty, a filtering mechanism is sometimes incorporated and the resulting algorithm is called "Filtered Beam Search". Filtered Beam Search differs from the Beam Search in the number of evaluations made. The nodes at each level are firstly evaluated by computationally cheaper method called "Filter Function". α is referred as "Filter Width" of the search. The best α nodes are selected for beam evaluation and all other remaining nodes are pruned from the search tree. Figure 5.1 provides the flow chart of the algorithm.

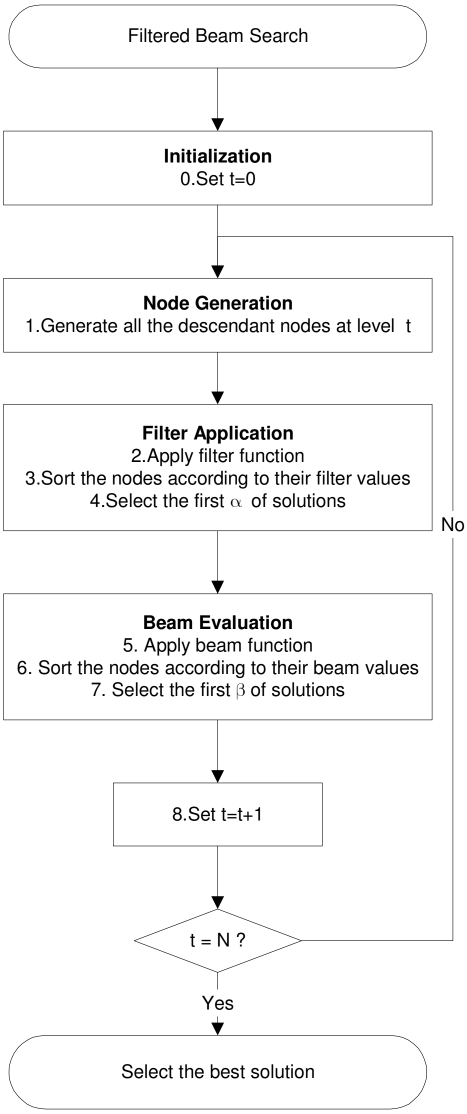

Figure 5.1 Flow Chart of a Filtered Beam Search Algorithm

Beam Width (β) and Filter Width (α) are two important parameters which should be carefully set in the search process to obtain a particular solution space representation. High values of β and α will increase the solution time and low values may easily lead to disregarding good solution alternatives. The trade-off between the quality and speed of the solutions should be carefully decided upon in setting β and α values. Sabuncuoglu & Karabuk (1998) state that these values are usually determined empirically and in most cases, an iterative process is used in which the parameter values are increased until the point beyond which neither β nor α improves the quality of the solution.

Filtered Beam Search may use two different strategies named as *Parallel Beam Search* and *Pooled Beam Search*. Below is the brief discussion of these versions.

5.1 PARALLEL BEAM SEARCH STRATEGY

In the parallel search strategy of the Filtered Beam Search, the most promising nodes are selected among the descendants of a parent node regardless of the nodes emanating from other branches. The filter and beam evaluation functions are applied on each descendant node of a parent node and on a certain number of parallel paths (beams). The number of nodes expanded at each level is smaller than or equal to the beam width. At the start, if the number of first level nodes is smaller than the beam width, then all the nodes are expanded until the number of nodes is greater than the beam width at the next level. As the filter width is determined separately for each beam, it might be smaller than the beam width. Figure 5.2 gives the schematic representation of this strategy.

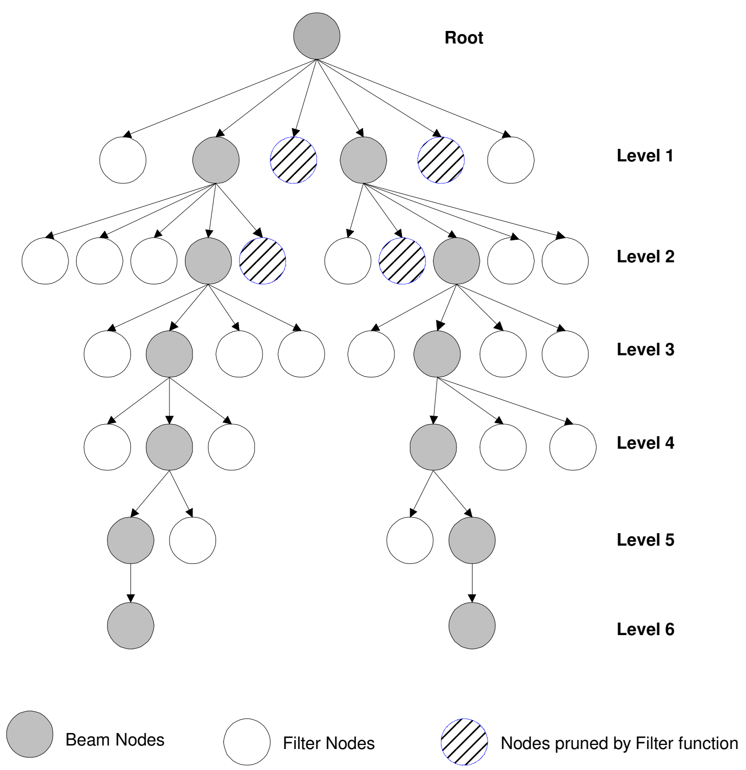

Figure 5.2 Filtered Beam Search Tree Using Parallel Strategy

In the above tree, beam width is set to 2 and filter width to 4. After the generation of the descendants of root node, two of the nodes are pruned by the filter evaluation function. Among the remaining 4 nodes, 2 of them were found to be promising and accepted as beam nodes. The search continued separately on these beam nodes, leading to a solution from each of them. In the second level, only one of the descendants was pruned by the filter evaluation, and on any other levels none of the nodes were pruned by filter evaluations since the number of descendants of a node is smaller than the filter width. At the end of the search, among the two solutions generated, the best one is reported.

5.2 POOLED BEAM SEARCH STRATEGY

In the pooled search strategy, instead of performing independent search on each beam, all nodes generated at a level are grouped and then filter and beam evaluation functions are employed to select the most promising α and then β nodes. The schematic representation of this situation is given in Figure 5.3.

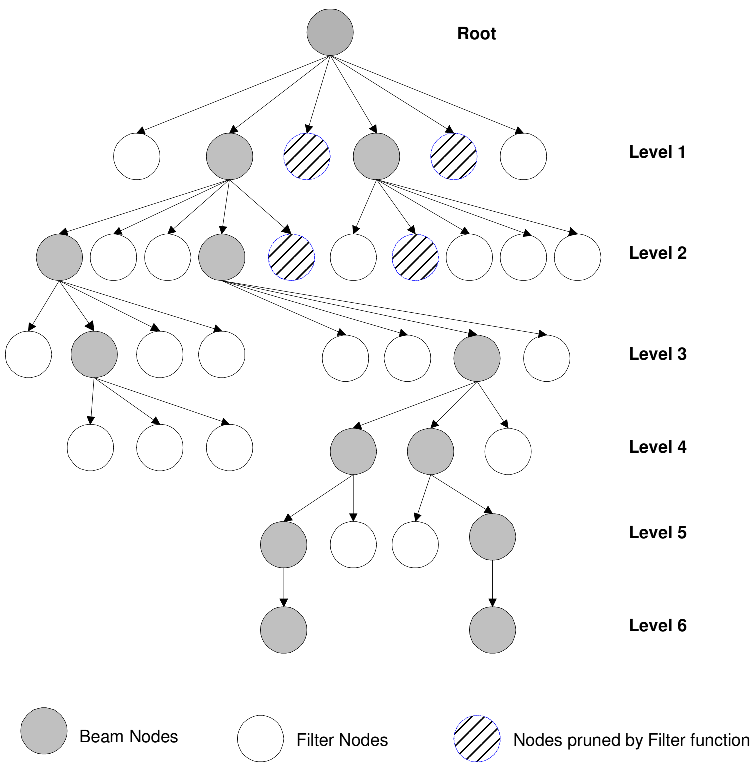

Figure 5.3 Filtered Beam Search Tree Using Pooled Strategy

In the tree of the above figure, beam width is set to 2 and filter width to 8. After generating the descendants of the root node, two of the nodes are pruned by the filter evaluation function. Among the remaining 4 nodes, 2 of them are found to be promising and accepted as beam nodes. In the second level, the descendants of the beam nodes are generated and among these 10 nodes, 2 of them were pruned by the filter evaluation. According to the beam evaluation, two nodes from the same descendant are selected and the search is continued from these nodes. At levels higher than 2, no nodes are pruned by filter evaluations since the number of descendants is smaller than the filter width. At the third level one descendant from each beam node is selected and at the forth level, two descendants of the right beam node are selected. At the end of the search, among two solutions generated, the best one is reported.

The basic disadvantage of the *Pooled Beam Search* is that the nodes that are selected for branching may emanate from the same parent node. Therefore similar partial schedules are likely to be evaluated. However, in the parallel version, we force to get solutions from different parents.

We employ both *Parallel* and *Pooled Beam Search* strategies in our experiments.

Beam and filter evaluation functions determine the promising nodes and the nodes to be permanently disregarded. Therefore these functions have a crucial role in the success of the Filtered Beam Search algorithms. Evaluation functions can be simple such as a priority rating (*One Step Priority Evaluation Function*) or complex such as extending the partial sequence by considering unscheduled jobs (*Total Cost Evaluation Function*). Although One Step Priority Evaluation functions have a local view and therefore can lead to a rejection of promising nodes, they are computationally cheap. On the other hand, Total Cost Evaluation functions offer a global view at an expense of higher computation times.

In this study, we use the following filter and beam evaluation functions.

5.3 EVALUATION FUNCTIONS

We propose the cost of the partial sequence found by the GKTNS policy ($cost$), upper bound 1 estimated for the remaining jobs (UB_1), and upper bound 2 estimated for the remaining jobs (UB_2) as Filter evaluation functions. In addition to these, we employ a simple priority rule favoring the job with the largest number of common tools with the last scheduled job among the forthcoming jobs (F_1). In case of ties with this priority rule, we favor the job minimizing the total number of tools required by the job and the last scheduled job (F_2).

As Beam evaluation functions we use lower bound 1, 2, 3 and 4 individually and in combination, and upper bound 1, and 2 for the unscheduled jobs.

The Beam Search generates a search tree of exactly n levels, since at each level the addition of a single job to the partial sequence is considered. At each level, at most β nodes are retained, resulting in βn beam function evaluations. Therefore a maximum of βn^2, i.e. $O(n^2)$, nodes are evaluated. The overall complexity of the Beam Search algorithm depends on the number of nodes evaluated by filter and beam functions, i.e. β and α, and the complexity of the evaluation functions employed. In cases the evaluation functions run in polynomial time as in our case, the Beam Search algorithm runs in polynomial time.

CHAPTER 6

COMPUTATIONAL RESULTS

In this section, we design an experiment to investigate the performances of Branch and Bound algorithm and Filtered Beam Search heuristic and to test the effects of different parameters on their performance measures. First we describe the generation of the experimental parameters. Then we state the performance measures and finally discuss the results of the computational runs.

6.1 DESIGN OF EXPERIMENTS

We generate the problem parameters as follows:

1. *The Problem Size:* Number of jobs (N) and number of tools required to process all jobs (T) together determine the problem size. The number of jobs is set to $N = 10, 15, 20, 25$, and for each N value, the number of the tools required by all jobs is set to $T = 10, 15, 20, 25$. We additionally test the Beam Search heuristics with problems of larger sizes, in particular, we use $N = 30, 40$ and $T = 40, 60$. Table 6.1 shows N and T combinations used in our experiments.

Table 6.1 N and T Values Used in Our Experiments

N	***T***			
10	10	15	20	25
15	10	15	20	25
20	10	15	20	25
25	10	15	20	25
30	-	-	-	40
40	-	-	-	60

2. *The Job-Tool Requirement Matrix:* The minimum ("*min*") and maximum ("*max*") number of tools required by a job are the main characteristics of the job-tool requirement matrix. The tool requirement of each job is generated as suggested in Crama et al. (1994). The number of tools required by a job, say j, i.e., $\left|T_j\right|$ is drawn from *Uniform[min, max]* distribution. Then, $\left|T_j\right|$ distinct integers are drawn from the uniform distribution over $\left[1,T\right]$ to find the tool set of job j. Finally we perform a check to ensure that the tool required by job j is not a subset of any other job's tools.

3. *The Capacities of Tool Magazine (C) and Tool Transporter (D):* In the majority of the problems, C is set to the maximum number of tools required by a job. To test the effect of C independent from other parameters, we generate some instances where C is greater than the maximum number of tools required by a job. For different values of C, we use different values of tool transporter capacity, D. As noted before when $D=1$, and $D=C$ our problem reduces to the problems of minimizing the number of the tool switches and minimizing the number of the tool switching instants, respectively.

In our main B&B runs, we use the (*min*, *max*), C, D and T values tabulated on the next page.

Table 6.2 T **,** (min, max) **,** C **and** D **Values Used in Branch & Bound Experiments**

T	*(min, max)*	*C*	*D*
10	(2, 5)	5	2, 4
15	(2, 5)	5	2, 4
15	(2, 10)	10	2, 5, 8
15	(5, 10)	10	2, 5, 8
20	(2, 5)	5	2, 4
20	(2, 10)	10	2, 5, 8
20	(5, 10)	10	2, 5, 8
20	(2, 15)	15	2, 5, 8
20	(5, 15)	15	2, 5, 8
20	(2, 5)	10	2, 4
20	(2, 5)	15	2, 4
20	(2, 10)	15	2, 5, 8
25	(2, 5)	5	2, 4
25	(2, 10)	10	2, 5, 8
25	(5, 10)	10	2, 5, 8
25	(2, 15)	15	2, 5, 8
25	(5, 15)	15	2, 5, 8
25	(2, 5)	10	2, 4
25	(2, 5)	15	2, 4
25	(2, 5)	20	2, 4
25	(2, 10)	15	2, 5, 8
25	(2, 10)	20	2, 5, 8
25	(5, 10)	20	2, 5, 8

We use 4 different values of N and for each N value we generate 10 problem instances for each parameter combination.

We set a termination limit of 2 hours for the execution of the Branch-and-Bound algorithm. The best solutions found, the node at which best solution is found and the number of nodes searched till the termination limit are reported and considered in analyzing the results.

All algorithms are coded in Visual C++ 6.0 version. All computational experiments are conducted on an Intel Pentium IV 2500 MHz computer under the Windows NT operating system.

6.2 PERFORMANCE MEASURES

In this section we first discuss the performance measures used to evaluate the B&B algorithm and then the performance measures for the Filtered Beam Search algorithm.

We use the following five performance measures in the evaluation of B&B algorithm:

1. The CPU time in seconds (average, maximum)
2. Total number of nodes generated (average, maximum)
3. Total number of nodes generated until reaching best/optimal solution(average, maximum)
4. Number of unsolved instances, out of 10, with in the termination limit of 2 hours

Note that first two performance measures apply to the Filtered Beam Search as well. For the Filtered Beam Search, we additionally report the number of instances optimal/best solution is found out of 10 instances.

6.3 PRELIMINARY EXPERIMENTS

We perform an initial experimentation to decide on the upper bounds and the lower bounds together with their order of implementation in our main experiments. We now give the parameter settings of our initial experiments, and discuss the effects of bounding mechanisms.

Recall that we have developed 2 different upper bounds and 4 different lower bounds. In order to test the performance of these bounds, we examine 14 versions of the Branch-and-Bound algorithm differing in the bounding mechanisms employed. The versions together with their abbreviations are listed in Table 6.3.

Table 6.3 Bounding Mechanisms Used in Branch & Bound Algorithms

Name	*Upper Bounds Used*	*Lower Bounds Used*
BB_1	UB_1	LB_1, LB_2, LB_3, LB_4
BB_2	UB_2	LB_1, LB_2, LB_3, LB_4
BB_3	---	LB_1, LB_2, LB_3, LB_4
BB_4	UB_1	LB_1, LB_2, LB_3
BB_5	UB_1	LB_1, LB_2, LB_4
BB_6	UB_1	LB_1, LB_2
BB_7	UB_1	LB_1, LB_3
BB_8	UB_1	LB_1, LB_4
BB_9	UB_1	LB_3, LB_4
BB_{10}	UB_1	LB_1
BB_{11}	UB_1	LB_2
BB_{12}	UB_1	LB_3
BB_{13}	UB_1	LB_4
BB_{14}	---	---

In our preliminary experimentation, we use the same data sets provided by Hertz et al. (1998). We additionally generate random data sets with $N = 10$, $T = 15$, $C = 10$, $(min, max) = (2,10)$ and $D = 2,5$. Table 6.4 lists the (min, max), C and D values used in our preliminary experiments for different N and T values.

Table 6.4 Parameters Used in Preliminary Experiments

N	*T*	*(min, max)*	*C*	*D*
10	10	(2,4)	4	1, 2, 3
10	10	(2,4)	5	1, 2, 3
10	10	(2,4)	6	1, 2, 3
10	10	(2,4)	7	1, 2, 3
15	20	(2,6)	6	1, 2, 3
15	20	(2,6)	8	1, 2, 3
15	20	(2,6)	10	1, 2, 3, 6
15	20	(2,6)	12	1, 2, 3, 6
10	15	(2,10)	10	2, 5
10	15	(2,10)	10	2, 5

For a specific combination with $N = 15$, $T = 20$, $C = 6$, $(min, max) = (2, 6)$ and $D = 1, 2, 3$, the performance of the algorithms are reported in the Tables 6.5, 6.6, 6.7, 6.8. The results obtained from the initial runs for other parameter combinations reported in Table 6.4 are similar and presented in **Appendix A**.

Table 6.5 Total Number of Nodes for $N = 15$, $T = 20$, $C = 6$, $(min, max) = (2, 6)$

	D=1		D=2		D=3	
	Avg.	Max.	Avg.	Max.	Avg.	Max.
BB_1	11364919.50	35150142.00	11497674.70	33577644.00	22429039.60	53215947.00
BB_2	11367291.40	35156842.00	11499069.30	33577766.00	22437911.20	53290121.00
BB_3	11366165.70	35156837.00	11499068.00	33577737.00	22437912.90	53290121.00
BB_4	11994980.60	36759166.00	12466131.00	36557275.00	25932233.20	58111504.00
BB_5	11800807.00	35887001.00	12417647.50	36391850.00	25723245.20	59225047.00
BB_6	14543241.30	40419129.00	22430717.90	80390950.00	73392112.80	131128562.00
BB_7	11994980.60	36759166.00	12467154.50	36557364.00	25942243.00	58197660.00
BB_8	11800807.00	35887001.00	12418493.80	36391929.00	25731617.70	59299213.00
BB_9	147197557.10	192877585.00	130993972.70	185545050.00	119297452.10	189566464.00
BB_{10}	14543241.30	40419129.00	22432522.70	80390950.00	74014197.70	133026307.00
BB_{11}	171617697.00	207040184.00	164953341.90	204796814.00	168070614.00	214715845.00
BB_{12}	170130453.00	213162574.00	154950694.00	204858658.00	145195740.90	209322854.00
BB_{13}	141078400.40	194588684.00	120094604.10	187313182.00	114637762.40	198425845.00
BB_{14}	246724818.80	282356871.00	234905815.10	272263204.00	236017880.90	275958825.00

Table 6.6 The CPU Time (sec.) for $N = 15$, $T = 20$, $C = 6$, $(min, max) = (2, 6)$

	D=1		D=2		D=3	
	Avg.	Max.	Avg.	Max.	Avg.	Max.
BB_1	784.30	2276.28	835.24	2338.37	1550.57	3590.13
BB_2	658.63	1897.92	701.38	1965.29	1301.42	3009.51
BB_3	658.74	1897.78	703.18	1980.07	1302.84	3009.22
BB_4	716.53	2128.82	771.09	2201.84	1549.22	3375.66
BB_5	809.08	2286.04	911.93	2705.29	1815.00	3967.58
BB_6	844.48	2263.61	1320.21	4922.53	4093.56	7200.00
BB_7	702.56	2087.12	755.36	2153.54	1508.61	3291.67
BB_8	796.97	2253.53	897.21	2663.06	1777.57	3891.49
BB_9	7200.00	7200.00	7191.17	7200.00	6167.60	7200.00
BB_{10}	830.38	2225.56	1300.09	4859.75	4059.85	7200.00
BB_{11}	7200.00	7200.00	7200.00	7200.00	7200.00	7200.00
BB_{12}	7200.00	7200.00	7200.00	7200.00	6497.02	7200.00
BB_{13}	7200.00	7200.00	7200.00	7200.00	6258.70	7200.00
BB_{14}	7200.00	7200.00	7200.00	7200.00	7200.00	7200.00

Table 6.7 # of Nodes to Optimality for $N = 15$, $T = 20$, $C = 6$, $(min, max) = (2,6)$

	D=1				D=2				D=3			
	Opt. Node		% of Opt. Node		Opt. Node		% of Opt. Node		Opt. Node		% of Opt. Node	
	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.
BB_1	7185643.00	33871695	63.23	96.36	4662529.10	31647804	40.55	94.25	1836153.00	16212528	8.19	30.47
BB_2	7189039.70	33881199	63.24	96.37	4663115.30	31647927	40.55	94.25	1836824.80	16212564	8.19	30.42
BB_3	7187180.30	33881194	63.23	96.37	4663114.00	31647898	40.55	94.25	1836826.50	16212564	8.19	30.42
BB_4	7619620.50	35465883	63.52	96.48	5054620.40	34512168	40.55	94.41	2676644.30	24282263	10.32	41.79
BB_5	7521735.30	34608554	63.74	96.44	5131576.40	34405047	41.32	94.54	1884004.80	16217364	7.32	27.38
BB_6	9428519.40	39122054	64.83	96.79	8755645.90	56051026	39.03	69.72	1565250.70	9857783	2.13	7.52
BB_7	7619620.50	35465883	63.52	96.48	5054629.30	34512257	40.54	94.41	2676655.10	24282263	10.32	41.72
BB_8	7521735.30	34608554	63.74	96.44	5131584.30	34405126	41.32	94.54	1884005.60	16217364	7.32	27.35
BB_9	437588.90	3291155	0.30	1.71	3946860.00	37575991	3.01	20.25	24350527.70	122287184	20.41	64.51
BB_{10}	9428519.40	39122054	64.83	96.79	8755657.70	56051144	39.03	69.72	1565270.10	9857783	2.11	7.41
BB_{11}	4309983.60	15932637	2.51	7.70	3566572.60	11684173	2.16	5.71	21535574.80	213276267	12.81	99.33
BB_{12}	743407.60	4615957	0.44	2.17	6864483.80	66505919	4.43	32.46	35152787.80	143639254	24.21	68.62
BB_{13}	673237.90	5272748	0.48	2.71	4696774.80	44755006	3.91	23.89	18769280.20	149884630	16.37	75.54
BB_{14}	36735067.80	125645376	14.89	44.50	14063162.90	80776001	5.99	29.67	255370.20	1933968	0.11	0.70

Table 6.8 # of Unsolved Instances out of 10 for $N = 15$, $T = 20$, $C = 6$, $(min, max) = (2,6)$

	D=1	D=2	D=3
BB_1	0	0	0
BB_2	0	0	0
BB_3	0	0	0
BB_4	0	0	0
BB_5	0	0	0
BB_6	0	0	3
BB_7	0	0	0
BB_8	0	0	0
BB_9	10	9	8
BB_{10}	0	0	3
BB_{11}	10	10	10
BB_{12}	10	10	8
BB_{13}	10	10	8
BB_{14}	10	10	10

6.3.1 Performance of the Lower Bounds

As can be seen from Table 6.8, the number of problem instances solved out of 10 drastically decreases as we use fewer lower bounds, i.e. we move down in the table. Moreover, it is a notable fact that without any lower and upper bounds, no problem instance can be solved within our time limit of 2 hours. The most powerful lower bound is LB_1, as without LB_1, the majority of the problem instances cannot be solved even though some other lower bounds and the best upper bound is utilized (***BB9***). However, when LB_1 is used together with UB_1, i.e. ***BB10*** case, for $D = 1$ and $D = 2$, all and for $D = 3$ most of the problem instances can be solved within two hours.

To analyze the performance of lower bounding schemes, we can compare ***BB**$_3$*, which incorporates all lower bounds, but no upper bound and ***BB**$_{14}$* which does not use any lower or upper bounds. Clearly utilization of all lower bounds drastically improves the number of instances solved out of ten within our time limit of 2 hours from 0 to 10, in these three sets of problems. We also observe that the node evaluations for no lower bound case, ***BB**$_{14}$*, is approximately 230-240 million, however, for ***BB**$_3$*, approximately 10-11 million nodes are evaluated which is the indication of a significant improvement in the performance of the algorithm.

We next try to see the incremental effects of the lower bounds. First we test whether LB_3 and LB_4, that seem to be unpromising individually, become effective when employed together. But still, this combination (***BB**$_9$* case) performs very poor in terms of both total # of nodes and CPU times. We further examine the combined effects of other lower bounds with LB_1. As can be seen from the tables, these combinations perform better in the CPU times, the number of nodes and in solving the majority of the problems. We finally observe that using all lower bounds in a sequential manner together with UB_1 (***BB**$_1$* case) provides the lowest number of node evaluations, therefore smallest CPU times among all lower bound combinations tested.

We do not report the percent deviation of the lower bounds at the root nodes from the optimal solutions. LB_1 and LB_2 values at the root node are same. Moreover, in our experiments we observe that the performances of LB_3 and LB_4 are relatively poor at the root node. In many problem instances where the job-tool requirement matrix is sparse and tool magazine capacity is large, the $\left\lceil \frac{l_{ij}}{D} \right\rceil$ values are quite small thereby deteriorating the performance of LB_3 and LB_4 utilizing these estimates. In those instances, LB_1 and LB_2 however achieve very good results.

6.3.2 Performance of the Upper Bounds

To test the performance of the upper bounds we try three different combinations, ***BB**$_1$*, ***BB**$_2$* and ***BB**$_3$* where all lower bounds are used sequentially. Among these combinations, ***BB**$_1$* (that uses UB_1) seems the best alternative in terms of the total number nodes generated. The performances of ***BB*2** (that uses UB_2) and ***BB*3** (no upper bound case) are quite similar. In terms of the CPU times, ***BB*2** turns out to be slightly better. The CPU times for ***BB*2** and ***BB*3** are quite similar, thus we can conclude that UB_2 is not significantly powerful.

Additional runs are carried to see the effects of using UB_1 at the first few levels of the B&B tree. The results of these partial tests are reported in Table 6.9 and Table 6.10.

Table 6.9 Total # of Nodes for Different Versions of UB_1

Setting					UB_1		Partial UB_1		No UB	
N	T	(min, max)	C	D	Avg.	Max.	Avg.	Max.	Avg.	Max.
10	15	(2,10)	10	2	29650.6	94352	29655.3	94354	29796.8	94354
10	15	(2,10)	10	5	220625.8	684173	220625.8	684173	220630.2	684173
10	10	(2,4)	4	1	7446.6	17392	7449.3	17392	7609	18365
15	20	(2,6)	6	1	11364919.5	35150142	11365749.7	35156828	11366165.7	35156837
15	20	(2,6)	8	1	941430.1	2637078	833391	2637078	941584	2637093
15	20	(2,6)	10	1	339675.1	1585432	339702	1585432	339779.7	1585432
15	20	(2,6)	12	1	35.6	171	41.4	229	119.8	264
10	10	(2,4)	4	2	7811.1	16390	7817.7	16390	7828.2	16395
15	20	(2,6)	6	2	11497674.7	33577644	11498562.4	33577728	11499068	33577737
15	20	(2,6)	8	2	850474.3	4503995	854488.6	4504224	854680.2	4504330
15	20	(2,6)	10	2	177911.6	699792	183538.3	699792	183729.4	699857
15	20	(2,6)	12	2	17825.3	178000	20048.6	200209	20246.6	200209
10	10	(2,4)	4	3	33823	95955	33830.1	95955	39827.9	95958
15	20	(2,6)	6	3	22429039.6	53215947	22437383.3	53290121	22437912.9	53290121
15	20	(2,6)	8	3	1965018.7	6145828	1979300.9	6145828	1979770.3	6145957
15	20	(2,6)	10	3	594925.2	1711715	632490.3	1873923	632616.2	1874066
15	20	(2,6)	12	3	511.7	3152	534	3187	586.7	3207
15	20	(2,6)	10	6	12750201.20	78704122.00	12761480.70	78704145.00	12763613.20	78704145.00
15	20	(2,6)	12	6	13719.80	136800.00	13725.20	136800.00	13884.80	136800.00

As can be seen from the table, using UB_1 at the first 2 levels for $N = 10$, at the first 3 levels for $N \geq 15$ provides the best results in terms of the CPU time and the number of nodes. The number of nodes when UB_1 is used in the first levels is close to the case that uses UB_1 throughout all search. The solution times are the smallest in partial utilization case compared to all and no upper bound cases. The results of the additional tests to see the effects of using UB_1 in further levels show that the savings in the number of nodes are small despite the considerable increase in CPU times.

Table 6.10 The CPU Time (Seconds) for Different Versions of UB_1

Setting					UB_1		Partial UB_1		No UB	
N	T	(min, max)	C	D	Avg.	Max.	Avg.	Max.	Avg.	Max.
10	15	(2,10)	10	2	0.99	3.12	0.83	2.58	0.82	2.53
10	15	(2,10)	10	5	6.80	19.35	5.68	16.32	5.66	16.24
10	10	(2,4)	4	1	0.22	0.53	0.19	0.46	0.19	0.48
15	20	(2,6)	6	1	784.30	2276.28	648.76	1867.91	658.74	1897.78
15	20	(2,6)	8	1	54.20	133.16	38.56	108.17	44.51	110.00
15	20	(2,6)	10	1	17.84	78.74	14.60	64.75	14.54	64.61
15	20	(2,6)	12	1	0.00	0.01	0.00	0.01	0.01	0.02
10	10	(2,4)	4	2	0.24	0.50	0.21	0.43	0.21	0.43
15	20	(2,6)	6	2	835.24	2338.37	689.82	1930.67	703.18	1980.07
15	20	(2,6)	8	2	53.08	266.09	42.54	212.19	43.31	216.61
15	20	(2,6)	10	2	10.71	39.71	8.80	31.78	8.96	32.38
15	20	(2,6)	12	2	1.06	10.58	0.95	9.42	0.96	9.45
10	10	(2,4)	4	3	0.93	2.45	0.79	2.11	0.91	2.11
15	20	(2,6)	6	3	1550.57	3590.13	1276.17	2946.61	1302.84	3009.22
15	20	(2,6)	8	3	130.47	403.50	104.39	320.31	106.91	329.35
15	20	(2,6)	10	3	34.05	96.62	28.46	85.97	29.15	88.24
15	20	(2,6)	12	3	0.03	0.16	0.03	0.14	0.03	0.13
15	20	(2,6)	10	6	553.52	3067.36	446.91	2510.17	462.86	2606.41
15	20	(2,6)	12	6	0.83	8.25	0.66	6.51	0.67	6.61

Based on these preliminary runs, we employ all lower bounds in a sequential manner together with UB_1 used only in first few levels in our main runs.

6.3.3 Performance of the Decomposition Theorem

The effect of ***Theorem 2*** becomes more pronounced when the total number of tools is large and the minimum and maximum tool requirements of the jobs are small. In such cases, decomposing jobs into distinct subgroups is more likely.

Table 6.11 reports the performance of the B&B algorithm with and without decomposition theorem (***Theorem 2***).

Table 6.11 Performance of B&B with and without *Theorem 2*

					With Decomposition				Without Decomposition			
Setting					Total # of Nodes		CPU (Sec.)		Total # of Nodes		CPU (Sec.)	
N	T	(min, max)	C	D	Avg	Max	Avg	Max	Avg	Max	Avg	Max
10	15	(2,10)	10	2	29655.3	94354	0.83	2.58	29655.3	94354	0.80	2.50
10	15	(2,10)	10	5	220625.8	684173	5.68	16.32	220639.4	684309	5.60	15.95
10	10	(2,4)	4	1	7449.3	17392	0.19	0.46	7452.2	17392	0.18	0.44
15	20	(2,6)	6	1	11365749.7	35156828	648.76	1867.91	12939392.1	35229008	681.14	1740.57
15	20	(2,6)	8	1	833391.0	2637078	38.56	108.17	1515051.3	8772021	65.90	350.72
15	20	(2,6)	10	1	339702.0	1585432	14.60	64.75	339702.0	1585432	14.82	65.58
15	20	(2,6)	12	1	41.4	229	0.00	0.01	41.4	229	0.00	0.02
10	10	(2,4)	4	2	7817.7	16390	0.21	0.43	7859.2	16409	0.20	0.42
15	20	(2,6)	6	2	11498562.4	33577728	689.82	1930.67	12061482.3	33627963	683.91	1811.52
15	20	(2,6)	8	2	854488.6	4504224	42.54	212.19	903080.8	4504224	45.41	215.31
15	20	(2,6)	10	2	183538.3	699792	8.80	31.78	183538.3	699792	8.93	32.24
15	20	(2,6)	12	2	20048.6	200209	0.95	9.42	20048.6	200209	0.97	9.65
10	10	(2,4)	4	3	33830.1	95955	0.79	2.11	34859.6	96691	0.79	2.06
15	20	(2,6)	6	3	22437383.3	53290121	1276.17	2946.61	24932386.0	53429976	1348.44	2861.82
15	20	(2,6)	8	3	1979300.9	6145828	104.39	320.31	2059735.7	6145828	109.71	325.44
15	20	(2,6)	10	3	632490.3	1873923	28.46	85.97	632512.8	1873923	28.94	87.50
15	20	(2,6)	12	3	534.0	3187	0.03	0.14	534.0	3187	0.02	0.14
15	20	(2,6)	10	6	12761480.7	78704145	446.91	2510.17	12761480.7	78704145	455.52	2555.93
15	20	(2,6)	12	6	13725.2	136800	0.66	6.51	13725.2	136800	0.66	6.62
10	20	(5,15)	15	5	263461.4	442343	8.08	13.33	263461.4	442343	8.09	13.20
10	25	(5,15)	15	8	184531.5	752325	7.23	28.89	184531.5	752325	7.25	29.10
15	15	(2,5)	5	2	9294143.8	40487705	482.06	1958.20	9332480.9	40522618	466.87	1903.64
15	20	(2,10)	15	2	7970040.9	74842806	297.32	2754.83	7970040.9	74842806	302.00	2798.94
15	25	(2,10)	15	5	9506059.8	58509068	504.54	2994.22	9506059.8	58509068	509.75	3016.39

As can be observed from Table 6.11, in general ***Theorem 2***, reduces the total number of nodes, thereby the CPU times. However, in some cases although total numbers of nodes are same when ***Theorem 2*** is and is not used, the CPU times without decomposition theorem seems better. These cases occur when the effort spent to test the decomposition theorem in partial sequences outweighs the reductions obtained through node eliminations. Note that such cases occur only when $|T_i| = C$ for some job i.

6.4 DISCUSSION OF THE RESULTS

In our main runs we use all lower bounds in their order of complexity. We employ upper bounds to partial sequences only at the first levels of the tree. We employ ***Theorem 2*** and update the decomposition sets when possible during the search. The Branch-and-Bound results for all of the sets are provided in **Appendix B**.

In this section, we investigate the effects of problem parameters on the performance measures.

6.4.1 *Effect of Number of Jobs, N*

The effect of the number of jobs, N, is quite obvious as keeping all other things same and increasing N leads to a drastic increase in the CPU times, thereby reducing the problem instances solved to optimality. N is the most predominant parameter and this effect is expected as it directly influences the number of branches and the depth of the search tree.

6.4.2 *Effect of Number of Tools, T*

To investigate the effect of T on the performance measures of the B&B algorithm, we use different values of T while keeping all other parameters fixed. Table 6.12 reports the algorithm performance for different values of T, for specific problem sets where the other parameters are fixed.

Table 6.12 B&B Performances with Different T Values

Setting					Total # of nodes		CPU Time (sec.)	
T	*N*	*(min, max)*	*C*	*D*	*Avg.*	*Max.*	*Avg.*	*Max.*
10	10	(2,5)	5	2	7472.70	20726	0.20	0.44
15					2781.00	5508	0.10	0.20
20					3248.90	9231	0.11	0.30
25					4350.80	25680	0.16	0.84
10	10	(2,5)	5	4	58470.30	280752	1.32	5.89
15					43831.70	145663	1.17	3.59
20					20304.40	78363	0.60	2.19
25					8604.60	23237	0.29	0.80
10	15	(2,5)	5	2	25125688.80	76859863	972.61	3022.78
15					9294143.80	40487705	482.06	1958.20
20					4537710.90	9297856	251.03	504.47
25					2035475.50	4898631	131.14	325.97

As can be observed from Table 6.12., for harder instances, as T increases the total number of nodes and the CPU time decrease. Note that as T increases, the computational complexity of the GKTNS policy also increases, which directly affects the solution times. However, the figures in Table 6.12, show that an increase in T, leads to a reduction in the total number of nodes and the CPU times. This contradiction is due to the additional reductions made by ***Theorem 2***. As T increases while keeping the minimum and maximum tool requirements for each job fixed, the probability of having distinct job sets increases, thereby leading to more reductions in the search tree.

6.4.3 Effect of Tool Transporter Capacity, D

In our experiments, we investigate the effect of the tool transporter capacity, D, on the performance of the algorithm. Table 6.13 summarizes the results of the problem sets, where all other parameters are fixed and the tool transporter capacity is set to $D = 2, 5, 8$.

Table 6.13 B&B Performances with Different D Values

Setting					Total # of nodes		CPU Time (sec.)	
D	*T*	*(min, max)*	*C*	*N*	*Avg.*	*Max.*	*Avg.*	*Max.*
2	20	(2,10)	15	10	805.8	7959	0.03	0.27
5					2755.0	17021	0.09	0.52
8					9339.3	48678	0.28	1.44
2	25	(2,10)	15	10	29.0	59	0.00	0.00
5					7410.1	36143	0.26	1.11
8					25233.7	192657	0.81	6.16
2	20	(2,10)	15	15	7970040.9	74842806	297.32	2754.83
5					22798786.1 (1)*	198446526	847.50	7200.00
8					63807193.9 (2)	202719696	2422.61	7200.00
2	25	(2,10)	15	15	606403.0	2252908	35.98	123.38
5					9506059.8	58509068	504.54	2994.22
8					46885357.1 (3)	150557058	2312.33	7200.00
2	20	(2,10)	15	20	91864779.9 (6)	145028232	5876.85	7200.00
5					113449966.3 (8)	143923283	6617.49	7200.00

*The numbers in the parentheses give the number of unsolved instances.

As can be observed from Table 6.13, the problem complexity increases significantly with an increase in tool transporter capacity, D. The total number of nodes and the CPU times increase notably for the larger D values while all other parameters are fixed. Moreover, as D increases, the number of problems solved out of 10 instances also decreases.

6.4.4 Effect of (min, max), C=max

The effect of the minimum and maximum tool requirement values of the jobs on the performance measures are analyzed together for the cases where the maximum tool requirement is set to C, tool magazine capacity. Table 6.14 presents the values of the performance measures for (min, max) combinations of (2,5), (2,10) and (5,10) and $C = max$.

Table 6.14 B&B Performances with Different (min, max) Values

Setting					Total # of nodes		CPU Time (sec.)	
(min, max)	*C*	*D*	*N*	*T*	*Avg.*	*Max.*	*Avg.*	*Max.*
(2,5)	5	2	10	25	4350.8	25680	0.16	0.84
(2,10)	10				13566.3	46690	0.51	1.58
(5,10)	10				19559.6	40404	0.87	1.81
(2,5)	5	2	15	25	2035475.5	4898631	131.14	325.97
(2,10)	10				40414956.4 (1)*	119390632	2322.46	7200.00
(5,10)	10				35955355.3 (1)	104211448	2615.13	7200.00
(2,10)	10	5	10	25	22009.6	64936	0.83	2.45
(5,10)	10				13612.4	70225	0.57	2.63
(2,10)	10	8	10	25	128988.5	302851	4.36	10.69
(5,10)	10				51069.6	213979	1.94	7.78
(2,10)	10	5	15	25	59170431.6 (3)	131343821	3712.33	7200.00
(5,10)	10				85173991.0 (5)	121440501	5489.45	7200.00

*The numbers in the parentheses give the number of unsolved instances.

Increasing the maximum value while keeping the minimum value fixed leads to an increase in the number of nodes and CPU times. Note that for harder instances where N is large, the decrease in the performance of the algorithm is much more notable.

Increasing the minimum value while keeping the maximum value fixed, in general, improves the performance of the algorithm in the number of nodes and the CPU time. The better performance can be attributed to the increasing power of LB_3 and LB_4. As the difference between minimum and maximum tool requirement values decreases, the job-tool requirement matrix becomes denser, thereby leading to better $\left\lceil \frac{l_{ij}}{D} \right\rceil$ estimates in LB_3 and LB_4 computations. An exception is due to the last entry in the table where an increase in the minimum value, decreases the number of instances solved sharply. The conclusions from the last two entries may be misleading as about half of the instances could not be solved to optimality in 2 hours time limit. These instances are harder than the previous ones in particular due to high N and D values.

6.4.5 Effect of Tool Magazine Capacity, C

We further investigate the effect of C, i.e. tool magazine capacity, on the performance measures when the minimum and maximum tool requirements are kept constant. Table 6.15 presents some of the values obtained from the settings (2,5) and (2,10), with $C = 5,10,15,20$ combinations.

Table 6.15 B&B Performances with Different C Values

Setting					Total # of nodes		CPU Time (sec.)	
C	*(min, max)*	*D*	*N*	*T*	*Avg.*	*Max.*	*Avg.*	*Max.*
5	(2,5)	2	10	25	4350.8	25680	0.16	0.84
10					11.0	11	0.00	0.01
15					11.0	11	0.00	0.00
20					11.0	11	0.00	0.00
10	(2,10)	8	10	25	128988.5	302851	4.36	10.69
15					25233.7	192657	0.81	6.16
20					11.0	11	0.00	0.01
5	(2,5)	2	15	25	2035475.5	4898631	131.14	325.97
10					153.8	1394	0.01	0.07
15					16.0	16	0.00	0.00
20					16.0	16	0.00	0.01
10	(2,10)	2	15	25	40414956.4 (1)*	119390632	2322.46	7200.00
15					606403.0	2252908	35.98	123.38
20					16.0	16	0.00	0.01
10	(2,10)	5	15	25	59170431.6 (3)	131343821	3712.33	7200.00
15					9506059.8	58509068	504.54	2994.22
20					601277.8	5968570	32.50	322.85

*The numbers in the parentheses give the number of unsolved instances.

As can be observed from the above table, as C increases, the performance of the algorithm improves. This is mainly due to the fact that as the capacity of the tool magazine increases, more tools can be inserted simultaneously and less number of additional tool switches will be required. Thus the performance of LB_1 will improve significantly.

6.5 PERFORMANCE OF BEAM SEARCH ALGORITHMS

In this section, we analyze the performance of the Filtered Beam Search algorithms with respect to different beam/filter evaluation functions and search strategies. Finally, the effects of filter and beam search parameters, α and β, on the performance measures are investigated.

6.5.1 Effects of the Evaluation Functions

In this section, we discuss the effects of beam and filter evaluation functions on the performance measures. We make our experiments with the problem settings presented in Table 6.16.

Table 6.16 Parameters Used in Beam Search Experiments

	N	*T*	*(min, max)*	*C*	*D*
PS_1	10	10	(2,4)	4	1
PS_2	15	20	(2,6)	6	1
PS_3	15	20	(2,6)	8	1
PS_4	10	10	(2,4)	4	3
PS_5	15	20	(2,6)	6	3
PS_6	15	20	(2,6)	8	3
PS_7	15	20	(2,6)	10	6
PS_8	15	20	(2,6)	12	6
PS_9	15	10	(2,5)	5	2
PS_{10}	15	20	(5,10)	10	2
PS_{11}	15	25	(2,10)	15	5
PS_{12}	20	20	(2,10)	15	2

Effect of the Beam Evaluation Function

In all experiments presented in this section, we present the values obtained for a beam width of $5N$. The results for other beam width values are presented in **Appendix C**.

We first compare Beam Search results employing UB_1 and UB_2 respectively. As can be seen from Tables 6.17 and 6.18, in all problem sets, the percent deviation of the Beam Search with UB_1 is no worse than that of UB_2. Moreover, the number of instances where the optimum values are found is much higher when UB_1 is used as a beam evaluation function. This behavior is effected neither by the search strategy (parallel vs. pooled) nor by the beam width.

Table 6.17 Performance of Parallel Beam Search using UB_1 and UB_2

	Parallel Beam Search with UB_1					Parallel Beam Search with UB_2				
	% dev. Avg.	% dev. Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal	% dev. Avg.	% dev. Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal
PS_1	1.60	8.33	1811.40	0.052	8	3.14	8.33	1765.40	0.035	6
PS_2	3.03	8.33	6848.80	0.497	4	6.48	15.63	6727.70	0.336	1
PS_3	3.17	9.09	6805.70	0.467	4	4.11	9.09	6700.20	0.319	3
PS_4	0.00	0.00	1788.90	0.057	10	0.00	0.00	1808.80	0.035	10
PS_5	3.85	12.50	6738.30	0.517	6	7.46	15.38	6644.40	0.344	3
PS_6	1.25	12.50	6575.60	0.479	9	8.08	12.50	6572.50	0.328	3
PS_7	0.00	0.00	6109.70	0.448	10	9.50	25.00	6543.30	0.324	6
PS_8	0.00	0.00	5849.90	0.414	10	5.83	33.33	6271.00	0.310	8
PS_9	5.53	14.29	6769.30	0.351	5	7.55	14.29	6687.50	0.254	3
PS_{10}	5.43	12.50	6835.80	0.619	3	9.71	15.00	6734.10	0.411	0
PS_{11}	4.86	20.00	6483.70	0.585	7	6.52	20.00	6381.30	0.390	6
PS_{12}	0.83	8.33	15810.70	1.787	9	6.97	9.09	16003.30	1.368	2

Table 6.18 Performance of Pooled Beam Search using UB_1 and UB_2

	Pooled Beam Search with UB_1					Pooled Beam Search with UB_2				
	% dev. Avg.	% dev. Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal	% dev. Avg.	% dev. Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal
PS_1	0.00	0.00	1844.00	0.112	10	0.77	7.69	1841.50	0.095	9
PS_2	2.24	4.17	6909.10	2.556	4	3.95	9.38	6861.00	2.427	3
PS_3	1.39	5.00	6919.40	2.521	7	4.13	10.00	6885.30	2.386	4
PS_4	0.00	0.00	1831.30	0.095	10	5.00	16.67	1832.90	0.063	7
PS_5	3.85	12.50	6866.30	2.232	6	3.44	10.00	6858.30	2.016	6
PS_6	3.75	12.50	6883.90	1.951	7	9.58	25.00	6884.10	1.841	3
PS_7	0.00	0.00	6763.60	1.469	10	13.50	25.00	6767.20	0.723	4
PS_8	0.00	0.00	6783.90	1.385	10	13.33	33.33	6830.60	0.681	5
PS_9	1.94	11.11	6975.00	1.980	8	8.87	28.57	6972.60	1.691	3
PS_{10}	1.07	6.25	6975.00	2.602	7	7.08	12.50	6975.00	2.443	2
PS_{11}	4.86	20.00	6929.10	1.743	7	12.62	16.67	6937.70	1.699	2
PS_{12}	3.48	9.09	17400.00	17.093	6	11.36	18.18	17313.40	19.420	0

The number of nodes and the CPU times for UB_1 and UB_2 cases are quite similar. When parallel and pooled strategies are compared, we see that although, the total number of nodes are similar, the CPU times differ significantly. This observation is true for other beam evaluation functions and is due to the extra sorting time needed for the pooled search strategy. From Tables 6.17 and 6.18, we can further conclude that no search strategy is dominant as the pooled search strategy performs better for some instances and for some others the parallel strategy performs better.

We investigate the performance of using the lower bounds sequentially in their complexity orders as a beam evaluation function. Some of the results are presented in Table 6.19.

Table 6.19 Performance of Parallel vs. Pooled Beam Search using all Lower Bounds

	Parallel Beam Search with all LBs					Pooled Beam Search with all LBs				
	% dev. Avg.	Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal	% dev. Avg.	Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal
PS_1	0.77	7.69	1820.40	0.060	9	0.00	0.00	1849.50	0.122	10
PS_2	7.85	14.81	6870.80	1.584	1	3.12	8.33	6962.90	3.671	4
PS_3	4.60	13.64	6852.10	0.483	3	1.34	4.55	6911.70	2.385	7
PS_4	0.00	0.00	1782.00	0.069	10	4.58	16.67	1827.40	0.111	7
PS_5	7.43	12.50	6793.10	1.579	2	4.69	12.50	6961.80	3.094	5
PS_6	9.51	14.29	6764.40	0.781	2	9.61	25.00	6954.20	1.966	3
PS_7	7.00	25.00	6451.10	0.378	7	15.50	40.00	6793.20	0.944	4
PS_8	0.00	0.00	5872.90	0.315	10	13.33	33.33	6871.60	1.114	5
PS_9	9.66	14.29	6842.90	1.528	1	7.43	18.18	6973.00	2.754	4
PS_{10}	7.30	12.50	6934.90	2.001	0	1.68	6.25	6975.00	3.750	7
PS_{11}	4.86	20.00	6779.40	0.497	7	8.19	20.00	6967.00	1.161	5
PS_{12}	1.74	9.09	16935.10	1.544	7	3.48	9.09	17400.00	12.044	6

The results once again show that no strategy is dominant when lower bounds are used. Moreover, when we compare the performance of Beam Search with lower bounds and Beam Search with UB_1, we can conclude that in general, Beam Search with UB_1 performs better in terms of percent deviation from the optimal and number of instances solved to optimality. However, usually, the computation time of Beam Search using lower bounds is better.

We further test the individual performances of the lower bounds and report some results in Tables 6.20, 6.21, 6.22 and 6.23.

Table 6.20 Performance of Parallel vs. Pooled Beam Search using LB_1

	Parallel Beam Search with LB_1					Pooled Beam Search with LB_1				
	% dev. Avg.	Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal	% dev. Avg.	Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal
PS_1	1.54	7.69	1825.90	0.039	8	0.00	0.00	1849.40	0.100	10
PS_2	7.80	14.81	6874.20	0.366	1	4.56	11.11	6964.40	2.644	3
PS_3	5.10	13.64	6869.40	0.339	2	1.80	9.09	6967.50	2.660	7
PS_4	0.00	0.00	1809.90	0.045	10	3.33	16.67	1839.40	0.070	8
PS_5	7.43	12.50	6824.30	0.371	2	5.85	12.50	6965.50	1.952	4
PS_6	9.51	14.29	6772.40	0.347	2	7.11	25.00	6958.20	1.600	5
PS_7	7.00	25.00	6451.10	0.334	7	15.50	40.00	6793.20	0.918	4
PS_8	0.00	0.00	5883.20	0.308	10	15.83	33.33	6874.60	1.191	4
PS_9	9.66	14.29	6899.50	0.290	1	6.64	14.29	6973.00	1.433	4
PS_{10}	9.18	15.00	6957.00	0.436	1	5.26	10.53	6975.00	1.830	2
PS_{11}	4.86	20.00	6777.40	0.427	7	8.19	20.00	6967.00	1.102	5
PS_{12}	1.74	9.09	16935.10	1.478	7	3.48	9.09	17400.00	12.230	6

The performance of the beam search that uses LB_1 is quite similar to the one that uses all lower bounds. The performance of LB_2 in terms of the percent deviation from the optimal and the number of instances optimal solution is found are the worst among all lower bounds although its computation time is lower.

Table 6.21 Performance of Parallel vs. Pooled Beam Search using LB_2

	Parallel Beam Search with LB_2					Pooled Beam Search with LB_2				
	% dev. Avg.	Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal	% dev. Avg.	Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal
PS_1	10.70	18.18	1776.50	0.039	2	11.41	18.18	1820.20	0.093	2
PS_2	15.28	20.83	6756.70	0.386	0	15.70	25.00	6912.60	2.205	0
PS_3	14.88	33.33	6542.90	0.356	0	15.38	33.33	6762.90	1.979	0
PS_4	0.00	0.00	1812.20	0.041	10	3.33	16.67	1844.70	0.096	8
PS_5	13.30	23.08	6727.40	0.392	0	14.30	23.08	6826.30	1.656	0
PS_6	15.62	25.00	6549.40	0.363	0	18.01	25.00	6835.60	1.417	0
PS_7	11.50	25.00	6160.20	0.349	5	15.50	40.00	6638.40	0.838	4
PS_8	18.33	33.33	6238.50	0.353	3	18.33	33.33	6760.30	0.758	3
PS_9	17.42	28.57	6892.60	0.288	1	20.87	42.86	6967.60	1.375	0
PS_{10}	15.36	21.05	6920.40	0.466	0	16.10	25.00	6975.00	1.763	0
PS_{11}	9.38	28.57	6568.60	0.444	5	15.81	28.57	6888.90	1.112	2
PS_{12}	16.44	25.00	16839.30	1.537	0	18.26	27.27	17235.30	9.120	0

The performances of LB_3 and LB_4 are similar in terms of the percent deviation and the number of instances optimal solution is found.

Table 6.22 Performance of Parallel vs. Pooled Beam Search using LB_3

	Parallel Beam Search with LB_3					Pooled Beam Search with LB_3				
	% dev. Avg.	% dev. Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal	% dev. Avg.	% dev. Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal
PS_1	0.00	0.00	1815.00	0.068	10	2.39	9.09	1821.30	0.132	7
PS_2	6.29	8.33	6858.50	1.940	1	5.35	16.67	6875.30	3.988	1
PS_3	5.95	10.00	6853.30	1.058	2	6.82	19.05	6885.10	2.684	3
PS_4	0.00	0.00	1793.10	0.061	10	2.92	16.67	1822.30	0.104	8
PS_5	5.59	12.50	6766.60	1.765	4	7.18	10.00	6870.50	3.263	2
PS_6	9.51	14.29	6767.70	1.462	2	10.76	25.00	6883.60	2.284	2
PS_7	9.50	25.00	6543.50	0.509	6	13.50	25.00	6769.00	0.913	4
PS_8	5.83	33.33	6271.00	0.316	8	13.33	33.33	6830.60	0.679	5
PS_9	9.66	14.29	6873.80	1.745	1	12.91	28.57	6962.80	2.965	0
PS_{10}	7.58	12.50	6924.60	1.919	2	4.01	12.50	6974.80	3.785	4
PS_{11}	7.95	20.00	6845.20	0.548	5	8.19	20.00	6975.00	1.163	5
PS_{12}	6.97	9.09	17149.40	1.479	2	9.62	18.18	17400.00	12.982	0

Table 6.23 Performance of Parallel vs. Pooled Beam Search using LB_4

	Parallel Beam Search with LB_4					Pooled Beam Search with LB_4				
	% dev. Avg.	% dev. Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal	% dev. Avg.	% dev. Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal
PS_1	0.77	7.69	1810.40	0.041	9	2.39	9.09	1836.80	0.103	7
PS_2	4.98	9.52	6866.40	0.373	1	4.08	7.41	6855.30	2.351	2
PS_3	5.95	13.64	6843.40	0.341	2	5.92	15.00	6885.10	1.993	3
PS_4	0.00	0.00	1776.00	0.039	10	6.01	16.67	1841.80	0.086	6
PS_5	7.35	12.50	6674.30	0.366	2	8.35	12.50	6832.50	1.974	1
PS_6	9.65	25.00	6770.80	0.342	3	9.90	25.00	6883.60	1.157	4
PS_7	9.50	25.00	6543.30	0.333	6	13.50	25.00	6767.20	0.732	4
PS_8	5.83	33.33	6271.00	0.321	8	13.33	33.33	6830.60	0.686	5
PS_9	8.66	14.29	6759.90	0.275	2	14.02	28.57	6973.30	1.911	0
PS_{10}	8.88	12.50	6918.20	0.449	0	4.49	12.50	6975.00	2.344	3
PS_{11}	7.95	20.00	6845.20	0.432	5	8.19	20.00	6975.00	1.047	5
PS_{12}	6.97	9.09	17149.40	1.501	2	9.62	18.18	17400.00	13.018	0

Finally, we use the simple priority rule, $F_1 \& F_2$ discussed in **Chapter 5** as the beam evaluation function. In $F_1 \& F_2$, F_1 is used as a selection rule and F_2 is used as a tie breaker. The results are tabulated in Table 6.24.

Table 6.24 Performance of Parallel vs. Pooled Beam Search using $F_1 \& F_2$

	Parallel Beam Search with F_1&F_2					Pooled Beam Search with F_1&F_2				
	% dev. Avg.	Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal	% dev. Avg.	Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal
PS_1	7.85	15.38	1635.00	0.030	3	11.41	18.18	1747.50	0.108	2
PS_2	15.28	20.83	6190.90	0.323	0	15.70	25.00	6540.30	2.778	0
PS_3	14.88	33.33	6353.60	0.315	0	15.38	33.33	6566.80	2.807	0
PS_4	1.67	16.67	1706.30	0.036	9	10.60	16.67	1780.10	0.122	3
PS_5	13.30	23.08	5971.00	0.322	0	14.30	23.08	6393.70	2.694	0
PS_6	17.87	25.00	6196.70	0.321	0	19.12	25.00	6699.50	2.807	0
PS_7	15.50	40.00	5770.30	0.308	4	15.50	40.00	6375.90	2.532	4
PS_8	15.83	33.33	5856.30	0.310	4	18.33	33.33	6548.10	2.692	3
PS_9	15.22	28.57	6350.50	0.243	1	21.27	33.33	6822.60	3.146	0
PS_{10}	18.23	25.00	6149.50	0.391	0	18.23	25.00	6613.60	2.805	0
PS_{11}	16.05	28.57	6376.20	0.393	1	17.48	28.57	6780.40	2.639	1
PS_{12}	18.33	27.27	16273.50	1.392	0	20.00	27.27	17080.80	23.601	0

The CPU time of the Beam Search with the priority rule is a bit lower, but the performance of the algorithm in terms of the percent deviation and the number of instances optimal solution is found is very poor.

Using all these observations, we continue by employing all lower bounds or UB_1 as the beam evaluation function in Filtered Beam Search experiments.

Effect of the Filter Evaluation Function

In this section, we investigate the effect of the filter evaluation functions discussed in **Chapter 5**. In all of the experiments presented in this section, we use $\beta = 3N$ as the beam width and $\alpha = 5N$ as the filter width. The results associated with other beam and filter width values are presented in **Appendix D**.

We first investigate the performance of the Filtered Beam Search using priority rule $F_1 \& F_2$ as the filter function and all lower bounds sequentially as the beam evaluation function. The results are presented in Table 6.25. As can be observed from this table, the performance of this algorithm in the percent deviation and the number of instances solved optimally is far beyond the classical Beam Search using UB_1 or lower bounds. The performance of the algorithm is only comparable with the Beam Search algorithm using $F_1 \& F_2$ as beam evaluation function. However the CPU times are lower than the beam search values.

Table 6.25 Performance of Parallel vs. Pooled Filtered Beam Search using $F_1 \& F_2$ as Filter Function and LB_s as Beam Function

	Parallel					Pooled				
	% dev. Avg.	Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal	% dev. Avg.	Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal
PS_1	9.67	18.18	1133.80	0.030	2	9.73	18.18	1141.70	0.044	2
PS_2	15.70	25.00	4209.40	0.458	0	14.75	25.00	4251.00	0.774	1
PS_3	14.38	33.33	4209.50	0.193	0	14.27	33.33	4263.00	0.635	1
PS_4	12.26	16.67	1131.90	0.033	2	12.26	16.67	1124.20	0.045	2
PS_5	13.30	23.08	4209.50	0.571	0	13.30	23.08	4232.10	0.814	1
PS_6	17.87	25.00	4209.50	0.220	0	17.87	25.00	4261.60	0.656	1
PS_7	15.50	40.00	4209.50	0.191	4	15.50	40.00	4257.00	0.635	4
PS_8	18.33	33.33	4209.50	0.192	3	20.83	33.33	4257.00	0.638	2
PS_9	20.16	33.33	4260.00	0.520	0	22.27	33.33	4271.40	0.839	0
PS_{10}	17.18	25.00	4260.00	0.628	0	18.23	25.00	4275.00	0.837	0
PS_{11}	12.71	28.57	4260.00	0.229	3	14.38	28.57	4275.00	0.624	2
PS_{12}	16.44	25.00	10580.00	0.842	0	20.00	27.27	10600.00	5.416	0

Similarly the performance of Filtered Beam Search using $F_1 \& F_2$ as the filter evaluation function and UB_1 as beam evaluation function are significantly worse than the Beam Search using UB_1. These values are reported in Table 6.26.

Table 6.26 Performance of Parallel vs. Pooled Filtered Beam Search using $F_1 \& F_2$ as Filter Function and UB_1 as Beam Function

	Parallel					Pooled				
	% dev. Avg.	Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal	% dev. Avg.	Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal
PS_1	7.85	15.38	1125.70	0.027	3	8.82	18.18	1139.00	0.040	3
PS_2	15.70	25.00	4174.10	0.231	0	15.70	25.00	4221.00	0.640	0
PS_3	13.88	33.33	4209.50	0.217	0	15.38	33.33	4255.90	0.645	0
PS_4	7.68	16.67	1121.60	0.028	5	13.10	16.67	1129.70	0.038	1
PS_5	12.63	23.08	4168.20	0.238	0	14.30	23.08	4185.70	0.637	0
PS_6	16.62	25.00	4209.50	0.230	0	17.87	25.00	4248.10	0.638	0
PS_7	15.50	40.00	4209.50	0.233	4	15.50	40.00	4252.20	0.850	4
PS_8	15.83	33.33	4209.50	0.229	4	20.83	33.33	4232.50	0.637	2
PS_9	17.53	42.86	4255.30	0.176	0	20.44	33.33	4275.00	0.666	0
PS_{10}	17.11	25.00	4260.00	0.285	0	18.23	25.00	4275.00	0.662	0
PS_{11}	12.71	28.57	4260.00	0.280	3	15.81	28.57	4275.00	0.633	2
PS_{12}	14.77	18.18	10580.00	0.912	0	19.92	33.33	10600.00	5.401	0

The performances of Filtered Beam Search using the cost of the partial schedule as the filter evaluation function and using all lower bounds or UB_1 as the beam evaluation functions are tabulated in Table 6.27 and Table 6.28 respectively.

Table 6.27 Performance of Parallel vs. Pooled Filtered Beam Search using $cost$ as Filter Function and LB_s as Beam Function

	Parallel					Pooled				
	% dev. Avg.	% dev. Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal	% dev. Avg.	% dev. Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal
PS_1	11.41	18.18	1114.10	0.035	2	3.34	9.09	1147.50	0.046	6
PS_2	15.70	25.00	4168.20	0.479	0	6.98	14.81	4268.00	0.727	1
PS_3	14.88	33.33	4213.10	0.247	0	1.39	5.00	4269.60	0.618	7
PS_4	14.76	16.67	1115.80	0.040	0	1.67	16.67	1133.50	0.053	9
PS_5	14.30	23.08	4186.60	0.558	0	9.26	20.00	4257.20	0.951	2
PS_6	19.12	25.00	4216.70	0.288	0	12.83	25.00	4263.60	0.849	2
PS_7	15.50	40.00	4216.70	0.353	4	13.50	40.00	4265.60	1.201	5
PS_8	18.33	33.33	4216.70	0.249	3	15.83	33.33	4265.60	0.993	4
PS_9	23.70	42.86	4248.30	0.473	0	18.49	28.57	4272.60	0.906	0
PS_{10}	18.23	25.00	4258.70	0.655	0	15.03	20.00	4274.60	0.956	0
PS_{11}	14.14	28.57	4260.00	0.305	3	14.14	28.57	4275.00	0.972	3
PS_{12}	14.77	25.00	10580.00	1.032	0	9.55	18.18	10600.00	6.542	1

Table 6.28 Performance of Parallel vs. Pooled Filtered Beam Search using $cost$ as Filter Function and UB_1 as Beam Function

	Parallel					Pooled				
	% dev. Avg.	% dev. Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal	% dev. Avg.	% dev. Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal
PS_1	7.85	15.38	1127.10	0.032	3	2.37	8.33	1144.50	0.043	7
PS_2	14.88	20.83	4178.50	0.284	0	5.98	12.50	4246.60	0.606	2
PS_3	11.87	23.81	4213.10	0.270	1	2.71	8.70	4221.40	0.602	5
PS_4	11.85	16.67	1110.10	0.033	2	1.67	16.67	1136.00	0.058	9
PS_5	12.70	20.00	4182.30	0.294	0	4.85	12.50	4196.80	0.808	5
PS_6	14.51	25.00	4216.70	0.288	0	5.86	12.50	4220.50	0.819	5
PS_7	15.50	40.00	4216.70	0.308	4	9.50	25.00	4219.60	0.997	6
PS_8	10.83	33.33	4216.70	0.286	6	7.50	25.00	4219.90	0.993	7
PS_9	19.44	33.33	4252.90	0.215	0	5.33	14.29	4273.10	0.794	5
PS_{10}	15.89	25.00	4260.00	0.355	0	4.44	10.53	4275.00	0.784	3
PS_{11}	7.95	20.00	4260.00	0.357	5	3.43	20.00	4275.00	0.996	8
PS_{12}	6.97	9.09	10580.00	1.098	2	3.48	9.09	10600.00	6.667	6

The performances of the algorithms using cost as filter evaluation function with parallel strategy are very poor compared to the performance of the beam search using the same beam evaluation function. However, the performance of the pooled strategy in this case turns out to be promising. This is due to the grouping of the nodes in a level and choosing the best in pooled strategy. The CPU times reported in these tables lead to similar conclusions made before.

The performance of the Filtered Beam Search using UB_1 as the filter evaluation function and using all lower bounds as the beam evaluation function is tabulated in Table 6.29. Note that the performance of the algorithm is worse than that of Beam Search using UB_1 or all lower bounds. The pooled strategy again performs better.

Table 6.29 Performance of Parallel vs. Pooled Filtered Beam Search using UB_1 as Filter Function and LB_s as Beam Function

	Parallel					Pooled				
	% dev.		Avg. # of	Avg. CPU	#	% dev.		Avg. # of	Avg. CPU	#
	Avg.	Max.	Nodes	Time (sec.)	optimal	Avg.	Max.	Nodes	Time (sec.)	optimal
PS_1	10.50	18.18	1107.90	0.039	2	4.82	15.38	1148.50	0.052	5
PS_2	15.28	20.83	4176.70	0.573	0	10.35	14.81	4266.20	1.099	0
PS_3	14.91	28.57	4214.30	0.295	0	7.82	14.29	4265.80	0.701	2
PS_4	11.85	16.67	1109.70	0.045	2	4.35	16.67	1146.30	0.058	7
PS_5	14.30	23.08	4181.00	0.623	0	10.03	15.38	4264.40	0.943	0
PS_6	17.87	25.00	4217.90	0.347	0	13.26	25.00	4270.20	0.866	0
PS_7	15.50	40.00	4221.50	0.314	4	11.50	25.00	4270.50	0.961	5
PS_8	18.33	33.33	4220.30	0.302	3	10.00	25.00	4270.40	1.002	6
PS_9	23.70	42.86	4250.00	0.537	0	12.91	28.57	4274.60	0.859	0
PS_{10}	18.23	25.00	4259.70	0.742	0	13.16	21.05	4275.00	0.963	0
PS_{11}	16.05	28.57	4260.00	0.379	1	9.62	20.00	4275.00	1.005	4
PS_{12}	14.77	25.00	10580.00	1.206	0	7.88	18.18	10600.00	7.241	3

6.5.2 Effects of the Search Parameters

In this section, we discuss the effects of beam and filter widths on the performance measures.

Effect of the Beam Width, β

In order to see the effect of beam width, we tested our beam search algorithms utilizing different beam evaluation functions and search strategies with 12 problem sets.

The results obtained from these tests are presented in **Appendix C**. Tables 6.30 and 6.31 present the results obtained from the two problem sets.

Table 6.30 Effect of β on the Parallel Beam Search Algorithm

	β	Parallel Beam Search with UB_1						
		% dev.		Total # of Nodes		CPU Time (sec.)		# optimal
		Avg.	Max.	Avg.	Max.	Avg.	Max.	
PS_9	1N	6.53	14.29	1524.80	1560.00	0.083	0.090	4
	2N	5.53	14.29	2845.00	2910.00	0.152	0.160	5
	3N	5.53	14.29	4160.00	4241.00	0.223	0.250	5
	4N	5.53	14.29	5472.80	5586.00	0.283	0.320	5
	5N	5.53	14.29	6769.30	6933.00	0.351	0.380	5
PS_{12}	1N	5.15	9.09	3581.60	3706.00	0.417	0.440	4
	2N	2.50	8.33	6744.00	6928.00	0.780	0.811	7
	3N	2.50	8.33	9885.30	10194.00	1.123	1.171	7
	4N	0.83	8.33	12855.70	13441.00	1.449	1.532	9
	5N	0.83	8.33	15810.70	16691.00	1.787	1.892	9

Table 6.31 Effect of β on the Pooled Beam Search Algorithm

	β	Pooled Beam Search with UB_1						
		% dev.		Total # of Nodes		CPU Time (sec.)		# optimal
		Avg.	Max.	Avg.	Max.	Avg.	Max.	
PS_9	1N	8.41	14.29	1575.00	1575.00	0.095	0.100	2
	2N	7.50	14.29	2925.00	2925.00	0.214	0.230	3
	3N	3.37	14.29	4275.00	4275.00	0.481	0.540	7
	4N	1.94	11.11	5625.00	5625.00	1.017	1.151	8
	5N	1.94	11.11	6975.00	6975.00	1.980	2.213	8
PS_{12}	1N	6.06	9.09	3800.00	3800.00	0.510	0.540	3
	2N	4.39	9.09	7200.00	7200.00	1.517	1.592	5
	3N	3.48	9.09	10600.00	10600.00	4.000	4.316	6
	4N	3.48	9.09	14000.00	14000.00	9.042	9.764	6
	5N	3.48	9.09	17400.00	17400.00	17.093	18.326	6

As can be observed from Tables 6.30 and 6.31, as the beam width increases, the percent deviation from the optimal solution decreases for both parallel and pooled versions. Moreover the number of instances where the optimal solution is found increases with increasing values of β at the expense of the total number of nodes and hence the

CPU time. Note in the table that, in some cases, the increase in β does not lead to better performance in percent deviations.

The performances of different versions of the Beam Searches with different beam widths presented in **Appendix C** lead to similar conclusions. Due to the insignificant computational times, and better performance measures attained, we use $\beta = 5N$ in our further experiments.

Effect of the Filter Width, α

In order to see the effect of filter width, we tested our Filtered Beam Search algorithms utilizing different beam evaluation functions and search strategies with 12 problem sets. Tables 6.32 and 6.33 present the results of Filtered Beam Search using UB_1 as the beam evaluation function and *cost* as the filter evaluation function on two problem sets with different beam and filter widths.

Table 6.32 Effects of β and α on the Parallel Beam Search Algorithm

			Parallel Filtered Beam Search						
			% dev.		Total # of Nodes		CPU Time (sec.)		#
	β	α	Avg.	Max.	Avg.	Max.	Avg.	Max.	optimal
PS_9	3N	---	5.53	14.29	4160.00	4241.00	0.223	0.250	5
	5N	---	5.53	14.29	6769.30	6933.00	0.351	0.380	5
	3N	5N	19.44	33.33	4252.90	4260.00	0.215	0.230	0
	5N	10N	23.70	42.86	6951.60	6960.00	0.376	0.400	0
PS_{12}	3N	---	2.50	8.33	9885.30	10194.00	1.123	1.171	7
	5N	---	0.83	8.33	15810.70	16691.00	1.787	1.892	9
	3N	5N	6.97	9.09	10580.00	10580.00	1.098	1.141	2
	5N	10N	16.44	27.27	17380.00	17380.00	1.930	1.992	1

Table 6.33 Effects of β and α on the Pooled Beam Search Algorithm

			Pooled Filtered Beam Search						
			% dev.		Total # of Nodes		CPU Time (sec.)		#
	β	α	Avg.	Max.	Avg.	Max.	Avg.	Max.	optimal
PS_9	3N	---	3.37	14.29	4275.00	4275.00	0.481	0.540	7
	5N	---	1.94	11.11	6975.00	6975.00	1.980	2.213	8
	3N	5N	5.33	14.29	4273.10	4275.00	0.794	0.941	5
	5N	10N	7.69	25.00	6973.40	6975.00	3.939	4.486	4
PS_{12}	3N	---	3.48	9.09	10600.00	10600.00	4.000	4.316	6
	5N	---	3.48	9.09	17400.00	17400.00	17.093	18.326	6
	3N	5N	3.48	9.09	10600.00	10600.00	6.667	7.220	6
	5N	10N	1.74	9.09	17400.00	17400.00	32.338	35.140	8

As the filter width increases, the percent deviation from the optimal solution and the number of instances where the optimal solution is found increases for parallel strategy but decreases for pooled strategy at the expense of the total number of nodes and hence the CPU time.

The performances of the Filtered Beam Searches using different filter and beam widths are similar and presented in **Appendix D**. Due to the very small computational times, and better average deviations, we use Beam Search of $\beta = 5N$ in our further experiments.

6.5.3 Performance Comparison of Beam Search and Other Heuristics

In this section, we compare the performance of Beam Search with truncated B&B and other heuristics in the literature. In Table 6.34, we report the percent average and maximum deviation, the number of instances the best/optimal solution is found, average CPU time and number of node evaluations in Beam Search for both parallel and pooled versions using UB_1 and all LB_s as the beam evaluation functions for $N = 20, 25$. The best results obtained from four different Beam Searches are summarized in the last two columns. The results of the Beam Search for $N = 10, 15$ are even better and presented in **Appendix E**. The performance of the Branch-and-Bound algorithm for the same settings is provided in **Appendix B**.

Note that the B&B values are optimal for some problem settings, but for some others they are the best results reported within a time limit of 2 hours. Therefore in some cases the solutions reported from Beam Search is even better than the best known solutions, leading to negative percent deviations.

First of all, the performances of Beam Searches are very robust in terms of the percent deviations and the number of instances where optimal/best solution is found. Generally, the deviations are around 0-3% and at most 10-13% and in at least half of the instances best solutions are found. Even though, for a given setting, the number of node evaluations for all versions are quite similar, the CPU times for pooled version is much higher due to the time spent for sorting all the nodes at a level. The CPU time for Beam Search using UB_1 turns out to be slightly longer than that of LB_s. The difference becomes more apparent when pooled versions are compared. As N increases, the number of nodes and hence the CPU time increases. The CPU times are very promising being no more than 70-80 seconds and neither of the Beam Search strategies is dominant in terms of percent deviations. Therefore one can conclude that Beam Search is a powerful alternative for near optimal solutions for all problem combinations.

Table 6.34 Performance Comparison of Beam Search vs. Truncated B&B

					Parallel								Pooled									
Setting					LB_s				UB_1				LB_s				UB_1				Best of all	
N	*T*	*(min, max)*	*C*	*D*	% dev.	# best	CPU (sec.)	Avg. Nodes	% dev.	# best	CPU (sec.)	Avg. Nodes	% dev.	# best	CPU (sec.)	Avg. Nodes	% dev.	# best	CPU (sec.)	Avg. Nodes	% dev.	# best
20	20	(2,5)	15	2	0.00	10	1.299	16735.3	0.00	10	1.559	16776.5	0.00	10	13.141	17230.0	0.00	10	11.128	17175.0	0.00	10
20	20	(2,5)	15	4	0.00	10	1.178	14659.1	0.00	10	1.451	15185.4	0.00	10	6.699	16174.6	0.00	10	9.650	16625.1	0.00	10
20	20	(2,10)	15	2	1.74	7	1.551	16935.1	0.83	9	1.787	15810.7	3.48	6	12.070	17400.0	3.48	6	17.072	17400.0	0.00	6
20	20	(2,10)	15	5	7.00	6	1.703	16777.2	3.67	8	1.827	15571.6	12.00	3	7.040	17372.8	3.67	8	12.823	17030.0	3.67	8
20	25	(2,5)	15	2	0.00	10	1.474	17102.8	0.00	10	1.790	17090.0	0.00	10	15.590	17310.0	0.00	10	12.228	17147.4	0.00	10
20	25	(2,5)	15	4	0.00	10	1.448	16430.7	0.00	10	1.758	16170.0	0.00	10	10.973	17038.8	0.00	10	10.284	16986.4	0.00	10
20	25	(2,10)	15	2	6.21	1	1.716	17202.4	6.31	3	2.146	16662.2	4.25	5	16.287	17400.0	4.92	4	19.526	17400.0	3.48	4
20	25	(2,10)	15	5	7.62	5	2.246	17130.1	4.76	7	2.191	16160.4	11.73	1	8.418	17395.8	6.19	6	14.198	17310.0	3.51	6
20	25	(2,10)	15	8	6.67	6	2.855	17014.4	1.67	9	2.213	15948.3	12.33	3	9.132	17380.4	1.67	9	13.003	17380.0	0.00	9
20	15	(2,5)	10	2	1.25	9	1.195	16582.9	0.00	10	1.400	15800.4	3.75	7	12.427	17359.0	1.25	9	15.566	17306.0	0.00	9
20	15	(2,5)	10	4	0.00	10	1.196	16211.2	0.00	10	1.375	15138.8	4.50	8	9.082	17384.0	2.00	9	11.218	17260.0	0.00	9
20	20	(2,5)	10	2	3.73	6	1.326	16633.2	1.91	8	1.663	16481.5	2.91	7	13.353	17400.0	2.82	7	17.216	17400.0	1.91	7
20	20	(2,5)	10	4	5.33	7	1.322	16264.3	0.00	10	1.619	15580.2	10.67	4	9.184	17352.4	0.00	10	13.744	17400.0	0.00	10
20	25	(2,5)	10	2	3.34	6	1.431	17023.4	3.34	6	1.809	16885.8	4.11	5	17.448	17391.0	2.51	7	17.385	17400.0	0.91	7
20	25	(2,5)	10	4	9.29	4	1.422	16533.2	9.29	4	1.808	16394.2	11.43	4	9.225	17258.0	9.29	4	12.964	17214.5	6.43	4
20	25	(2,5)	20	2	0.00	10	1.490	17028.2	0.00	10	1.725	17011.4	0.00	10	13.373	17220.0	0.00	10	9.809	17206.8	0.00	10
20	25	(2,5)	20	4	0.00	10	1.421	16006.6	0.00	10	1.672	16000.1	0.00	10	8.237	16644.0	0.00	10	7.039	16623.0	0.00	10
20	25	(5,10)	20	2	1.54	8	1.763	16623.8	0.77	9	2.153	16197.5	1.54	8	14.908	17400.0	0.00	10	17.648	17400.0	0.00	10
20	25	(5,10)	20	5	0.00	10	1.733	15372.9	0.00	10	2.013	14488.5	6.67	6	6.281	16930.4	1.67	9	11.703	16816.0	0.00	9
20	25	(2,10)	20	2	0.00	10	1.609	16950.5	0.00	10	1.992	16785.8	0.00	10	17.260	17310.0	0.00	10	14.954	17268.0	0.00	10
20	25	(2,10)	20	5	0.00	10	1.554	15293.5	0.00	10	1.860	14909.2	4.00	8	7.444	16949.7	0.00	10	11.234	16675.4	0.00	10
20	25	(2,10)	20	8	5.00	8	1.610	15740.5	0.00	10	1.871	14801.1	12.50	5	8.514	17057.0	0.00	10	9.559	16879.6	0.00	10
25	20	(2,5)	10	4	5.33	7	3.926	33311.6	3.10	8	4.661	32343.0	9.86	4	41.698	34908.9	1.67	9	68.912	34896.7	1.67	9
25	25	(2,5)	20	2	0.00	10	4.458	34649.9	0.00	10	4.950	34469.0	0.00	10	80.902	34775.0	0.00	10	46.990	34602.4	0.00	10
25	25	(2,5)	20	4	0.00	10	4.385	33647.1	0.00	10	4.891	33200.5	0.00	10	49.730	33964.4	0.00	10	30.969	33912.0	0.00	10
25	25	(5,10)	20	2	0.83	8	5.194	33798.5	1.55	8	6.006	32434.4	2.98	6	64.149	35000.0	2.26	6	90.181	35000.0	0.12	6
25	25	(5,10)	20	5	-4.29	10	5.217	32128.7	0.24	9	5.905	30699.5	4.76	7	33.719	34856.5	0.00	10	60.273	34825.0	-4.29	10
25	25	(5,10)	20	8	0.00	9	5.264	32219.4	-3.67	10	5.925	30580.0	5.67	6	32.520	34471.9	-3.67	10	54.676	34326.8	-3.67	10
25	25	(2,10)	20	2	0.77	9	4.788	34171.3	0.77	9	5.552	33052.1	2.25	7	76.351	34993.6	2.31	7	82.883	34800.4	0.77	7
25	25	(2,10)	20	5	-2.86	10	4.810	32543.4	-4.29	10	5.317	30354.1	2.00	9	35.506	34875.0	-5.71	10	60.536	34562.5	-5.71	10
25	25	(2,10)	20	8	2.50	9	4.904	33006.9	-4.00	10	5.538	31543.7	8.50	6	31.665	34570.3	-2.00	10	49.449	33777.9	-4.00	10

We further compare the performance of Beam Search with other algorithms in the literature presented by Hertz et al. (1998). We provide the results of the GENIUS*, Nearest Neighbor* and 2-opt* heuristic of Hertz et al. (1998) in Table 6.35. We use a beam width of $5N$ in these experiments. The deviations of Beam Search presented in this table are the deviations from the best reported values of Hertz et al. (1998). We give the performances of Beam Search using UB_1 and/or LB_s with parallel or pooled strategy in **Appendix F**. The deviation of the best results and the total CPU time of all these different versions are reported in Table 6.35.

Table 6.35 Performance Comparison of Beam Search vs Heuristics from Hertz et al. (1998)

Setting					GENIUS*		NN*		2-opt*		Beam Search	
N	T	(min,max)	C	D	% dev.	CPU Time (sec.)	% dev.	CPU Time (sec.)	% dev.	CPU Time (sec.)	% dev.	CPU Time (sec.)
10	10	(2, 4)	4	1	0.80	34.00	0.80	1.20	8.80	0.50	0.00	0.34
10	10	(2, 4)	5	1	0.00	34.10	1.80	1.50	7.40	0.50	0.00	0.28
10	10	(2, 4)	6	1	0.00	33.10	2.00	1.50	4.00	0.40	0.00	0.26
10	10	(2, 4)	7	1	0.00	33.70	0.00	1.50	0.00	0.40	0.00	0.22
15	20	(2, 6)	6	1	1.10	118.90	6.70	7.60	11.50	3.40	0.43	8.26
15	20	(2, 6)	8	1	0.40	121.10	4.50	7.80	8.60	2.60	-0.48	5.80
15	20	(2, 6)	10	1	0.00	113.70	2.50	8.20	7.10	2.50	0.02	5.17
15	20	(2, 6)	12	1	0.00	110.60	0.00	8.30	0.50	2.40	0.00	4.88
30	40	(5, 15)	15	1	1.30	1004.40	10.00	158.90	9.70	91.20	2.34	3553.53
30	40	(5, 15)	17	1	1.90	1084.50	10.60	158.90	10.40	83.60	2.56	3608.26
30	40	(5, 15)	20	1	2.40	1120.20	9.10	163.60	12.70	69.60	3.88	2005.13
30	40	(5, 15)	25	1	1.30	950.00	6.30	169.80	10.40	49.10	1.28	1140.67
40	60	(7, 20)	20	1	1.20	2947.90	9.10	679.10	8.30	461.80	2.17	45568.39
40	60	(7, 20)	22	1	1.70	2939.10	7.90	690.70	8.30	457.00	3.13	51075.30
40	60	(7, 20)	25	1	1.40	2902.50	8.10	717.20	8.10	459.60	3.11	40335.83
40	60	(7, 20)	30	1	0.80	2495.40	7.40	763.00	8.70	124.40	3.11	16514.51

The percent deviations and the number instances of best solutions found are better than the reported values in Hertz et al. (1998) for the combinations where $N = 10, 15$. Note that the CPU times of the Beam Search for these instances are also quite small. When we consider the larger problem sizes, we observe that Beam Search algorithms suffer from high node evaluations and therefore computation times. As can be seen from Table 6.35, the performance of the Beam Search in terms of percent deviation is better than NN* and 2-opt*, but it is worse than GENIUS*. We did not discuss the relative performances in CPU times, as our algorithms and theirs are conducted in different media.

In **Appendix F**, we also provide the performance of Beam Search using F_1 & F_2 as the beam evaluation function with both parallel and pooled strategies. The performance of these versions are very poor, and is outperformed by both Beam Search using UB_1 and LB_s. Thus one can conclude that our Beam Search algorithms outperform the one given in Zhou et al. (2004).

We do not discuss the individual effects of the parameters on the Beam Search algorithm. As expected, when N increases, the total number of nodes and the CPU time also increase. However the percent deviation from the optimal solution and the number of instances solved to optimality do not change considerably.

The detailed results of the experiments are given in **Appendices A**, **B**, **C**, **D**, **E** and **F**. We report the preliminary experiments results of the Branch-and-Bound algorithms in **Appendix A**, the full experimentation results of B&B in **Appendix B**, the preliminary experiments results of Beam Search algorithms in **Appendix C** and the Filtered Beam Search results in **Appendix D**. Finally we tabulate of the experimental results for the comparison of truncated B&B and Beam Search in **Appendix E** and the performances of Beam Searches on the instances taken from Hertz et al. (1998) in **Appendix F**.

CHAPTER 7

CONCLUSIONS AND DIRECTIONS FOR FUTURE RESEARCH

In this study, we addressed an FMS scheduling problem in operational level which simultaneously considers job scheduling and tool switching decisions by coordinating automatic tool transfers between machine and tool storage areas to account for tool availability and tool switches on the machine.

We show various links of the problem with other well-studied problems in the literature and show its NP-Hardness. We propose a Branch-and-Bound algorithm for optimization and various Beam and Filtered Beam Search algorithms for approximation purposes. The efficiency of the algorithms is enhanced with reduction and decomposition mechanisms and several lower and upper bounding schemes.

The results of our computational experiments reveal that our Branch-and-Bound algorithm can satisfactorily solve moderate-sized problems with up to 25 jobs and 25 tools.

We observe from our experiments that the dominant factor effecting the difficulty of the problem is the number of jobs, N. Moreover, the capacity of the tool magazine, C, the capacity of the tool transporter, D, and the sparsity of the job-tool requirement matrix are other dominant factors affecting the problem complexity. The most difficult problem combinations are the ones having large N and D values together with small tool magazine capacities, C, and dense job-tool matrices. We observe that for most of the problems, the optimal/best solutions are found in the first half of the search supporting the appropriateness of our depth-first strategy in B&B and the power of the lower bounding mechanisms. Hence one can suggest the use of truncated B&B as an attractive alternative to the optimal solutions.

The Beam Search algorithms with lower bounds and first upper bound produce results that are very close to the optimal ones in very short CPU times. As the beam width increases, higher quality solutions are attained at an expense of higher node evaluations and CPU times. The performance of Filtered Beam Search is relatively poor. They are quick and the complexities are exponential functions of the problem parameters.

To the best of our knowledge, our study is the first optimization attempt to solve the tool transportation problem in the FMS environments. There are a number of research areas to which our study can be extended, the most noteworthy of which are discussed below.

Recall that the objective of our model is to minimize the number of tool transporter movements. This objective is adequate when the majority of the idle time of an FMS is due to the tool transporter, i.e. tool interchanging time and processing time of a job is insignificant compared to the transportation time. A model considering all of these times and aiming to minimize a function of those, like total flow time or total weighted flow time, might well fit to the cases where these times are significant.

Our model assumes that all tools require exactly one tool slot in the magazine and tool transporter. In practice, there can be additional restrictions on the placement of tools on the magazine. Tools may not be of the same size and weight, so the tool placement and weight balancing of the magazine and the tool transporter may become an important concern.

In some cases, the number of tools required by some jobs may be more than the tool magazine capacity of the machine, thereby necessitating the tool switches during the processing of a job. These cases are also worth addressing.

Matzliach & Tzur (1998) state a dynamic tool switching problem in which the parts arrive randomly and the tool sizes are non-uniform. Obviously, such an environment is more realistic and extension of our model to this dynamic environment can be another interesting future research area.

The tool loading and part sequencing among multiple machines is usually more practical than single machine environments. In these cases, the problem would have additional complexities brought by tool and tool transporter sharing among the machines.

Another issue whose importance becomes magnified with growing batch sizes is the tool life. The tool changes due to the tool wear or breakage can trigger more frequent switches, thus should also be incorporated into the model.

We study an operational problem where the capacity of the tool transporter is a fixed parameter. A design problem may point out the case where the capacity is a decision variable for a given upper limit on the maximum number of tool transporter movements required between each job pair.

REFERENCES

Al-Fawzan, M.A., and Al-Sultan, K.S., (2002), "A tabu search-based algorithm for minimizing the number of tool switches on a flexible machine", *Computers & Industrial Engineering*, Vol. 44, pp. 35-47.

Bard, J.F., (1988), "A heuristic for minimizing the number of tool switches on a flexible machine", *IIE Transactions*, Vol. 20, pp. 382-391.

Crama, Y. (1997), "Combinatorial optimization models for production scheduling in automated manufacturing systems", *European Journal of Operational Research*, Vol. 99, pp. 136–153.

Crama, Y., Kolen, A.W.J., Oerlemans, A.G., and Spieksma, F.C.R., (1994), "Minimizing the number of tool switches on a flexible machine", *International Journal of Flexible Manufacturing Systems*, Vol. 6, pp. 33-54.

Denizel, M., (2003), "Minimization of the number of tool magazine setups on automated machines: A lagrangean decomposition approach", *Operations Research*, Vol. 51, pp. 309-320.

Djellab, H., Djellab, K., and Gourgand, M., (2000), "A new heuristic based on a hypergraph representation for the tool switching problem", *International Journal of Production Economics*, Vol. 64, pp. 165-176.

Fathi, Y., and Barnette, K.W., (2002), "Heuristic procedures for the parallel machine problem with tool switches", *International Journal of Production Research*, Vol. 40, pp. 151-164.

Garey, M.R., and Johnson, D.S., (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco.

Grieco, A., Semeraro, Q., Tolio, T., and Toma, S., (1995), "Simulation of tool and part flow in FMSs", *International Journal of Production Research*, Vol. 33, pp. 643-658.

Hertz, A., Laporte, G., Mittaz, M., and Stecke, K.E., (1998), "Heuristics for minimizing tool switches when scheduling part types on a flexible machine", *IIE Transactions*, Vol. 30, pp. 689-694.

Hertz, A., and Widmer, M., (1996), "An improved tabu search approach for solving the job shop scheduling problem with tooling constraints", *Discrete Applied Mathematics*, Vol. 65, pp. 319–345.

Keung, K.W., Ip, W.H., and Lee, T.C., (2001), "The solution of a multi-objective tool selection model using the GA approach", *International Journal of Advanced Manufacturing Technology*, Vol. 18, pp. 771–777.

Keung, K.W., Ip, W.H., and Lee, T.C., (2001), "A genetic algorithm approach to the multiple machine tool selection problem", *Journal of Intelligent Manufacturing*, Vol. 12, pp. 331–342.

Khan, B.K., Gupta, B.D., Sen Gupta, D.K., and Kumar, K.D., (2000), "A generalized procedure for minimizing tool changeovers of two parallel and identical CNC machining centers", *Production Planning & Control*, Vol. 11, pp.62-73.

Laporte, G., Salazar-González, J.J., and Semet, F., (2004), "Exact algorithms for the job sequencing and tool switching problem", *IIE Transactions*, Vol. 36, pp. 37-45.

Matzliach, B., and Tzur, M., (1998), "The online tool switching problem with non-uniform tool size", *International Journal of Production Research*, Vol. 36, pp. 3407-3420.

Privault, C., and Finke, G., (1995), "Modeling a tool switching problem on a single NC-machine", *Journal of Intelligent Manufacturing*, Vol. 6, pp. 87- 94.

Reeves, C.R., (1995), *Modern heuristic Techniques for Combinatorial Problems*, McGraw-Hill, Berkshire.

Roh, H.K., and Kim, Y.D., (1997), "Due-date based loading and scheduling methods for a flexible manufacturing system with an automatic tool transporter", *International Journal of Production Research*, Vol. 35, pp. 2989-3003.

Rupe, J., and Kuo, W., (1997), "Solutions to a modified tool loading problem for a single FMM", *International Journal of Production Research*, Vol. 35, pp. 2253-2268.

Sabuncuoğlu, İ., and Karabük, S., (1998), "A beam search-based algorithm and evaluation of scheduling approaches for flexible manufacturing systems", *IIE Transactions*, Vol. 30, pp. 179-191.

Shirazi, R., and Frizelle, G.D.M., (2001), "Minimizing the number of tool switches on a flexible machine: an empirical study", *International Journal of Production Research*, Vol. 39, pp. 3547-3560.

Song, C.Y., and Hwang, H., (2002), “Optimal tooling policy for a tool switching problem of a flexible machine with automatic tool transporter”, *International Journal of Production Research*, Vol. 40, pp. 873-883.

Tang, C., and Denardo, E., (1988a), “Models arising from a flexible manufacturing machines, Part I: Minimization of the number of tool switches", *Operations Research*, Vol. 36, pp. 767-777.

Tang, C., and Denardo, E., (1988b), “Models arising from a flexible manufacturing machines, Part II: Minimization of the number of switching instants”, *Operations Research*, Vol. 36, pp. 778-784.

Tzur, M., and Altman, A., (2004), “Minimization of tool switches for a flexible manufacturing machine with slot assignment of different tool sizes”, *IIE Transactions*, Vol. 36, pp. 95-110.

Widmer, M., (1991), “Job shop scheduling with tooling constraints: A tabu search approach”, *Journal of the Operational Research Society*, Vol. 42, pp. 75-82.

Veeramani, D., Upton, D.M., and Barash, M.M., (1992), “Cutting-tool management in computer-integrated manufacturing”, *International Journal of Flexible Manufacturing Systems*, Vol. 3, pp. 237-265.

Zhou, B.H., Xi, L.F., and Cao, Y.S., (2004), “A beam-search-based algorithm for the tool switching problem on a flexible machine”, *Advanced Manufacturing Technology* (in press).

APPENDIX A

COMPUTATIONAL RESULTS FOR PRELIMINARY B&B EXPERIMENTS

In this appendix, we provide the detailed results of our preliminary Branch-and-Bound experiments. Parameters used in these experiments are given in Table 6.4.

Table A.1 Preliminary Results for B&B for N=10, T=10, (*min,max*)=(2,4), C=4, D=1

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		# unsolved
	Avg.	Max.	Avg.	Max.	Avg.	Max.	
BB_1	7446.60	17392.00	0.22	0.53	69.10	567.00	0
BB_2	7608.30	18365.00	0.19	0.48	316.10	1525.00	0
BB_3	7609.00	18365.00	0.19	0.48	316.90	1525.00	0
BB_4	7905.70	19704.00	0.20	0.48	85.10	727.00	0
BB_5	7570.70	17392.00	0.22	0.53	70.40	580.00	0
BB_6	8786.30	20786.00	0.22	0.52	191.70	1793.00	0
BB_7	7905.70	19704.00	0.20	0.49	85.10	727.00	0
BB_8	7570.70	17392.00	0.21	0.52	70.40	580.00	0
BB_9	170389.50	409413.00	3.59	7.61	154.50	1421.00	0
BB_{10}	8786.30	20786.00	0.21	0.50	191.70	1793.00	0
BB_{11}	208539.70	413966.00	3.98	7.34	834.40	8220.00	0
BB_{12}	223893.10	573630.00	3.98	9.17	204.70	1923.00	0
BB_{13}	176085.10	409621.00	3.69	7.50	158.30	1459.00	0
BB_{14}	1044705.70	1708981.00	13.77	22.76	20726.60	102351.00	0

Table A.2 Preliminary Results for B&B for N=10, T=10, (*min,max*)=(2,4), C=4, D=2

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	7811.10	16390.00	0.24	0.50	258.70	1277.00	0
BB_2	7826.90	16395.00	0.21	0.43	288.00	1291.00	0
BB_3	7828.20	16395.00	0.21	0.43	289.90	1291.00	0
BB_4	8316.80	16940.00	0.23	0.48	263.60	1307.00	0
BB_5	8524.90	16450.00	0.27	0.53	876.50	7134.00	0
BB_6	20865.80	63665.00	0.52	1.54	715.30	4490.00	0
BB_7	8320.40	16940.00	0.22	0.46	263.60	1307.00	0
BB_8	8528.50	16450.00	0.26	0.50	876.50	7134.00	0
BB_9	88365.60	340652.00	1.88	6.57	649.40	4634.00	0
BB_{10}	20869.40	63665.00	0.50	1.50	715.30	4490.00	0
BB_{11}	313418.00	521376.00	6.11	9.41	3071.00	17202.00	0
BB_{12}	111568.20	414528.00	2.06	7.10	920.70	6622.00	0
BB_{13}	95809.40	348982.00	2.06	6.70	1820.70	13016.00	0
BB_{14}	1103986.70	1509950.00	15.01	20.91	32423.40	165723.00	0

Table A.3 Preliminary Results for B&B for N=10, T=10, (*min,max*)=(2,4), C=4, D=3

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	33823.00	95955.00	0.93	2.45	4.70	16.00	0
BB_2	39826.50	95958.00	0.91	2.11	6361.20	35756.00	0
BB_3	39827.90	95958.00	0.91	2.11	6362.70	35757.00	0
BB_4	39522.10	115583.00	0.98	2.69	4.70	16.00	0
BB_5	34810.60	96175.00	0.97	2.47	4.70	16.00	0
BB_6	178238.10	468421.00	3.85	9.57	4.70	16.00	0
BB_7	39529.20	115583.00	0.92	2.50	4.70	16.00	0
BB_8	34817.70	96175.00	0.91	2.32	4.70	16.00	0
BB_9	115352.90	248833.00	2.44	5.19	4.70	16.00	0
BB_{10}	178296.10	468421.00	3.70	9.17	4.70	16.00	0
BB_{11}	871303.50	1562096.00	16.38	29.31	4.70	16.00	0
BB_{12}	156922.30	309564.00	2.93	5.65	4.70	16.00	0
BB_{13}	123702.50	272447.00	2.62	5.43	4.70	16.00	0
BB_{14}	1998458.50	3439728.00	26.98	46.96	145200.40	1026555.00	0

Table A.4 Preliminary Results for B&B for N=10, T=10, (*min,max*)=(2,4), C=5, D=1

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	2316.30	10030.00	0.06	0.24	10.60	45.00	0
BB_2	2485.40	10465.00	0.05	0.21	211.70	651.00	0
BB_3	2485.40	10465.00	0.05	0.20	211.70	651.00	0
BB_4	2316.30	10030.00	0.06	0.24	10.60	45.00	0
BB_5	2316.30	10030.00	0.06	0.23	10.60	45.00	0
BB_6	2316.30	10030.00	0.06	0.23	10.60	45.00	0
BB_7	2316.30	10030.00	0.06	0.28	10.60	45.00	0
BB_8	2316.30	10030.00	0.05	0.22	10.60	45.00	0
BB_9	527397.60	1329631.00	8.10	19.52	10.60	45.00	0
BB_{10}	2316.30	10030.00	0.06	0.23	10.60	45.00	0
BB_{11}	427533.40	1270461.00	7.05	20.39	10.60	45.00	0
BB_{12}	543923.40	1329631.00	8.16	19.50	10.60	45.00	0
BB_{13}	652772.40	1879073.00	9.49	25.94	10.60	45.00	0
BB_{14}	750724.40	1886231.00	9.09	22.65	17189.40	111870.00	0

Table A.5 Preliminary Results for B&B for N=10, T=10, (*min,max*)=(2,4), C=5, D=2

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	1969.30	5993.00	0.05	0.14	958.10	3398.00	0
BB_2	2165.80	7242.00	0.05	0.14	1023.70	3474.00	0
BB_3	2165.80	7242.00	0.05	0.15	1023.70	3474.00	0
BB_4	1969.30	5993.00	0.05	0.13	958.10	3398.00	0
BB_5	1969.30	5993.00	0.05	0.13	958.10	3398.00	0
BB_6	1969.30	5993.00	0.05	0.13	958.10	3398.00	0
BB_7	2121.40	7242.00	0.05	0.15	969.50	3452.00	0
BB_8	2121.40	7242.00	0.05	0.14	969.50	3452.00	0
BB_9	373387.90	973542.00	6.35	14.37	111792.50	698619.00	0
BB_{10}	2121.40	7242.00	0.05	0.14	969.50	3452.00	0
BB_{11}	225810.70	398972.00	4.04	6.90	76265.80	317002.00	0
BB_{12}	412895.70	973542.00	6.54	14.35	127149.80	787380.00	0
BB_{13}	405145.00	1151453.00	6.67	16.27	114546.60	707170.00	0
BB_{14}	562666.40	1151453.00	7.09	13.78	162517.60	856488.00	0

Table A.6 Preliminary Results for B&B for N=10, T=10, (*min,max*)=(2,4), C=5, D=3

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	7665.20	18620.00	0.19	0.43	3942.40	16540.00	0
BB_2	7756.00	18634.00	0.16	0.39	4048.80	16583.00	0
BB_3	7756.00	18634.00	0.16	0.37	4048.80	16583.00	0
BB_4	7777.60	19081.00	0.18	0.40	4054.80	17001.00	0
BB_5	7665.20	18620.00	0.19	0.42	3942.40	16540.00	0
BB_6	8170.00	21016.00	0.18	0.43	4447.20	18936.00	0
BB_7	7777.60	19081.00	0.17	0.41	4054.80	17001.00	0
BB_8	7665.20	18620.00	0.18	0.41	3942.40	16540.00	0
BB_9	287696.50	826889.00	5.22	13.95	45462.50	181977.00	0
BB_{10}	8170.00	21016.00	0.17	0.40	4447.20	18936.00	0
BB_{11}	403557.70	1148261.00	6.97	18.85	83229.50	252068.00	0
BB_{12}	341484.50	1017342.00	5.63	15.64	56637.10	213577.00	0
BB_{13}	306684.30	861197.00	5.43	14.13	47621.70	183660.00	0
BB_{14}	553723.70	1485782.00	7.12	18.25	117111.40	274884.00	0

Table A.7 Preliminary Results for B&B for N=10, T=10, (*min,max*)=(2,4), C=6, D=1

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	1360.30	13460.00	0.03	0.29	9.10	45.00	0
BB_2	1404.90	13474.00	0.03	0.25	61.90	140.00	0
BB_3	1404.90	13474.00	0.03	0.25	61.90	140.00	0
BB_4	1360.30	13460.00	0.03	0.30	9.10	45.00	0
BB_5	1360.30	13460.00	0.03	0.29	9.10	45.00	0
BB_6	1360.30	13460.00	0.03	0.30	9.10	45.00	0
BB_7	1360.30	13460.00	0.03	0.28	9.10	45.00	0
BB_8	1360.30	13460.00	0.03	0.28	9.10	45.00	0
BB_9	233261.20	2332469.00	3.17	31.69	9.10	45.00	0
BB_{10}	1360.30	13460.00	0.03	0.28	9.10	45.00	0
BB_{11}	155656.20	1556419.00	2.42	24.24	9.10	45.00	0
BB_{12}	233261.20	2332469.00	3.17	31.70	9.10	45.00	0
BB_{13}	233738.00	2337237.00	3.11	31.10	9.10	45.00	0
BB_{14}	237032.90	2337237.00	2.78	27.36	3313.60	19706.00	0

Table A.8 Preliminary Results for B&B for N=10, T=10, (*min,max*)=(2,4), C=6, D=2

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	1815.30	7631.00	0.05	0.20	10.20	43.00	0
BB_2	1846.60	7631.00	0.04	0.16	48.00	140.00	0
BB_3	1846.60	7631.00	0.04	0.16	48.00	140.00	0
BB_4	1815.30	7631.00	0.05	0.20	10.20	43.00	0
BB_5	1815.30	7631.00	0.05	0.19	10.20	43.00	0
BB_6	1815.30	7631.00	0.05	0.19	10.20	43.00	0
BB_7	1947.10	8311.00	0.05	0.21	10.20	43.00	0
BB_8	1947.10	8311.00	0.06	0.27	10.20	43.00	0
BB_9	515991.70	1691127.00	7.49	23.97	92.60	866.00	0
BB_{10}	1947.10	8311.00	0.05	0.20	10.20	43.00	0
BB_{11}	276138.00	921341.00	4.56	14.98	70.00	640.00	0
BB_{12}	515991.70	1691127.00	7.49	24.01	92.60	866.00	0
BB_{13}	523437.10	1695861.00	7.43	23.55	92.60	866.00	0
BB_{14}	524911.50	1695861.00	6.38	20.31	1587.20	14440.00	0

Table A.9 Preliminary Results for B&B for N=10, T=10, (*min,max*)=(2,4), C=6, D=3

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	99.10	650.00	0.00	0.01	90.70	645.00	0
BB_2	121.10	670.00	0.00	0.01	116.70	670.00	0
BB_3	121.10	670.00	0.00	0.01	116.70	670.00	0
BB_4	99.10	650.00	0.01	0.08	90.70	645.00	0
BB_5	99.10	650.00	0.00	0.01	90.70	645.00	0
BB_6	99.10	650.00	0.01	0.02	90.70	645.00	0
BB_7	99.10	650.00	0.00	0.01	90.70	645.00	0
BB_8	99.10	650.00	0.00	0.02	90.70	645.00	0
BB_9	2813.50	26928.00	0.05	0.39	2805.10	26923.00	0
BB_{10}	99.10	650.00	0.00	0.02	90.70	645.00	0
BB_{11}	2463.90	23457.00	0.05	0.47	2455.50	23452.00	0
BB_{12}	2813.50	26928.00	0.04	0.39	2805.10	26923.00	0
BB_{13}	2856.60	27359.00	0.04	0.39	2848.20	27354.00	0
BB_{14}	2883.40	27379.00	0.04	0.34	2879.00	27379.00	0

Table A.10 Preliminary Results for B&B for N=10, T=10, (*min,max*)=(2,4), C=7, D=1

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	11.00	11.00	0.00	0.01	0.30	3.00	0
BB_2	15.40	55.00	0.00	0.00	5.50	55.00	0
BB_3	15.40	55.00	0.00	0.00	5.50	55.00	0
BB_4	11.00	11.00	0.00	0.00	0.30	3.00	0
BB_5	11.00	11.00	0.00	0.00	0.30	3.00	0
BB_6	11.00	11.00	0.00	0.00	0.30	3.00	0
BB_7	11.00	11.00	0.00	0.00	0.30	3.00	0
BB_8	11.00	11.00	0.00	0.00	0.30	3.00	0
BB_9	11.00	11.00	0.00	0.01	0.30	3.00	0
BB_{10}	11.00	11.00	0.00	0.01	0.30	3.00	0
BB_{11}	11.00	11.00	0.00	0.01	0.30	3.00	0
BB_{12}	11.00	11.00	0.00	0.00	0.30	3.00	0
BB_{13}	11.00	11.00	0.00	0.00	0.30	3.00	0
BB_{14}	15.40	55.00	0.00	0.00	5.50	55.00	0

Table A.11 Preliminary Results for B&B for N=10, T=10, (*min,max*)=(2,4), C=7, D=2

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	12.70	28.00	0.00	0.00	2.90	25.00	0
BB_2	25.50	68.00	0.00	0.00	17.80	68.00	0
BB_3	25.50	68.00	0.00	0.00	17.80	68.00	0
BB_4	12.70	28.00	0.00	0.00	2.90	25.00	0
BB_5	12.70	28.00	0.00	0.01	2.90	25.00	0
BB_6	12.70	28.00	0.00	0.00	2.90	25.00	0
BB_7	12.70	28.00	0.00	0.00	2.90	25.00	0
BB_8	12.70	28.00	0.00	0.01	2.90	25.00	0
BB_9	12.70	28.00	0.00	0.00	2.90	25.00	0
BB_{10}	12.70	28.00	0.00	0.00	2.90	25.00	0
BB_{11}	12.70	28.00	0.00	0.00	2.90	25.00	0
BB_{12}	12.70	28.00	0.00	0.00	2.90	25.00	0
BB_{13}	12.70	28.00	0.00	0.01	2.90	25.00	0
BB_{14}	511.10	4924.00	0.01	0.05	503.40	4924.00	0

Table A.12 Preliminary Results for B&B for N=10, T=10, (*min,max*)=(2,4), C=7, D=3

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	11.00	11.00	0.00	0.00	0.30	3.00	0
BB_2	15.70	58.00	0.00	0.01	5.80	58.00	0
BB_3	15.70	58.00	0.00	0.00	5.80	58.00	0
BB_4	11.00	11.00	0.00	0.00	0.30	3.00	0
BB_5	11.00	11.00	0.00	0.01	0.30	3.00	0
BB_6	11.00	11.00	0.00	0.01	0.30	3.00	0
BB_7	11.00	11.00	0.00	0.01	0.30	3.00	0
BB_8	11.00	11.00	0.00	0.00	0.30	3.00	0
BB_9	11.00	11.00	0.00	0.00	0.30	3.00	0
BB_{10}	11.00	11.00	0.00	0.01	0.30	3.00	0
BB_{11}	11.00	11.00	0.00	0.01	0.30	3.00	0
BB_{12}	11.00	11.00	0.00	0.00	0.30	3.00	0
BB_{13}	11.00	11.00	0.00	0.00	0.30	3.00	0
BB_{14}	16.30	64.00	0.00	0.00	6.40	64.00	0

Table A.13 Preliminary Results for B&B for N=15, T=20, (*min,max*)=(2,6), C=6, D=1

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	11364919.50	35150142.00	784.30	2276.28	7185643.00	33871695.00	0
BB_2	11367291.40	35156842.00	658.63	1897.92	7189039.70	33881199.00	0
BB_3	11366165.70	35156837.00	658.74	1897.78	7187180.30	33881194.00	0
BB_4	11994980.60	36759166.00	716.53	2128.82	7619620.50	35465883.00	0
BB_5	11800807.00	35887001.00	809.08	2286.04	7521735.30	34608554.00	0
BB_6	14543241.30	40419129.00	844.48	2263.61	9428519.40	39122054.00	0
BB_7	11994980.60	36759166.00	702.56	2087.12	7619620.50	35465883.00	0
BB_8	11800807.00	35887001.00	796.97	2253.53	7521735.30	34608554.00	0
BB_9	147197557.10	192877585.00	7200.00	7200.00	437588.90	3291155.00	10
BB_{10}	14543241.30	40419129.00	830.38	2225.56	9428519.40	39122054.00	0
BB_{11}	171617697.00	207040184.00	7200.00	7200.00	4309983.60	15932637.00	10
BB_{12}	170130453.00	213162574.00	7200.00	7200.00	743407.60	4615957.00	10
BB_{13}	141078400.40	194588684.00	7200.00	7200.00	673237.90	5272748.00	10
BB_{14}	246724818.80	282356871.00	7200.00	7200.00	36735067.80	125645376.00	10

Table A.14 Preliminary Results for B&B for N=15, T=20, (*min,max*)=(2,6), C=6, D=2

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	11497674.70	33577644.00	835.24	2338.37	4662529.10	31647804.00	0
BB_2	11499069.30	33577766.00	701.38	1965.29	4663115.30	31647927.00	0
BB_3	11499068.00	33577737.00	703.18	1980.07	4663114.00	31647898.00	0
BB_4	12466131.00	36557275.00	771.09	2201.84	5054620.40	34512168.00	0
BB_5	12417647.50	36391850.00	911.93	2705.29	5131576.40	34405047.00	0
BB_6	22430717.90	80390950.00	1320.21	4922.53	8755645.90	56051026.00	0
BB_7	12467154.50	36557364.00	755.36	2153.54	5054629.30	34512257.00	0
BB_8	12418493.80	36391929.00	897.21	2663.06	5131584.30	34405126.00	0
BB_9	130993972.70	185545050.00	7191.17	7200.00	3946860.00	37575991.00	9
BB_{10}	22432522.70	80390950.00	1300.09	4859.75	8755657.70	56051144.00	0
BB_{11}	164953341.90	204796814.00	7200.00	7200.00	3566572.60	11684173.00	10
BB_{12}	154950694.00	204858658.00	7200.00	7200.00	6864483.80	66505919.00	10
BB_{13}	120094604.10	187313182.00	7200.00	7200.00	4696774.80	44755006.00	10
BB_{14}	234905815.10	272263204.00	7200.00	7200.00	14063162.90	80776001.00	10

Table A.15 Preliminary Results for B&B for N=15, T=20, (*min,max*)=(2,6), C=6, D=3

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	22429039.60	53215947.00	1550.57	3590.13	1836153.00	16212528.00	0
BB_2	22437911.20	53290121.00	1301.42	3009.51	1836824.80	16212564.00	0
BB_3	22437912.90	53290121.00	1302.84	3009.22	1836826.50	16212564.00	0
BB_4	25932233.20	58111504.00	1549.22	3375.66	2676644.30	24282263.00	0
BB_5	25723245.20	59225047.00	1815.00	3967.58	1884004.80	16217364.00	0
BB_6	73392112.80	131128562.00	4093.56	7200.00	1565250.70	9857783.00	3
BB_7	25942243.00	58197660.00	1508.61	3291.67	2676655.10	24282263.00	0
BB_8	25731617.70	59299213.00	1777.57	3891.49	1884005.60	16217364.00	0
BB_9	119297452.10	189566464.00	6167.60	7200.00	24350527.70	122287184.00	8
BB_{10}	74014197.70	133026307.00	4059.85	7200.00	1565270.10	9857783.00	3
BB_{11}	168070614.00	214715845.00	7200.00	7200.00	21535574.80	213276267.00	10
BB_{12}	145195740.90	209322854.00	6497.02	7200.00	35152787.80	143639254.00	8
BB_{13}	114637762.40	198425845.00	6258.70	7200.00	18769280.20	149884630.00	8
BB_{14}	236017880.90	275958825.00	7200.00	7200.00	255370.20	1933968.00	10

Table A.16 Preliminary Results for B&B for N=15, T=20, (*min,max*)=(2,6), C=8, D=1

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	941430.10	2637078.00	54.20	133.16	213900.60	1154092.00	0
BB_2	941585.40	2637093.00	44.52	110.06	214067.70	1154128.00	0
BB_3	941584.00	2637093.00	44.51	110.00	214066.30	1154128.00	0
BB_4	941430.10	2637078.00	52.42	132.97	213900.60	1154092.00	0
BB_5	941430.10	2637078.00	53.74	131.71	213900.60	1154092.00	0
BB_6	941430.10	2637078.00	51.90	131.67	213900.60	1154092.00	0
BB_7	941430.10	2637078.00	51.33	129.47	213900.60	1154092.00	0
BB_8	941430.10	2637078.00	52.77	128.62	213900.60	1154092.00	0
BB_9	187352151.20	254010202.00	6480.00	7200.00	16434558.30	122835714.00	9
BB_{10}	941430.10	2637078.00	50.97	128.80	213900.60	1154092.00	0
BB_{11}	175347951.80	227326164.00	6480.00	7200.00	3826880.60	27111966.00	9
BB_{12}	195645545.00	253982755.00	6480.00	7200.00	19029634.20	141050959.00	9
BB_{13}	193508646.50	266235327.00	6480.00	7200.00	19808308.20	153521067.00	9
BB_{14}	239887092.40	303330616.00	6480.01	7200.00	8927127.90	73266492.00	9

Table A.17 Preliminary Results for B&B for N=15, T=20, (*min,max*)=(2,6), C=8, D=2

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	850474.30	4503995.00	53.08	266.09	426321.90	4237301.00	0
BB_2	854680.10	4504330.00	43.28	216.67	426580.00	4237648.00	0
BB_3	854680.20	4504330.00	43.31	216.61	426580.10	4237648.00	0
BB_4	850474.30	4503995.00	49.76	248.95	426321.90	4237301.00	0
BB_5	850474.30	4503995.00	52.70	263.91	426321.90	4237301.00	0
BB_6	850474.30	4503995.00	49.31	246.21	426321.90	4237301.00	0
BB_7	860243.90	4503995.00	49.37	243.62	426321.90	4237301.00	0
BB_8	860243.90	4503995.00	52.40	259.32	426321.90	4237301.00	0
BB_9	165432420.00	227359243.00	6481.37	7200.00	4468292.70	32660839.00	9
BB_{10}	860243.90	4503995.00	48.99	241.12	426321.90	4237301.00	0
BB_{11}	160626327.20	208683032.00	6480.57	7200.00	7182950.30	43753474.00	9
BB_{12}	176440885.00	227361085.00	6481.37	7200.00	5396117.20	40385288.00	9
BB_{13}	169035024.00	241414159.00	6481.40	7200.00	5291828.40	36599835.00	9
BB_{14}	224064546.70	279775545.00	6481.23	7200.00	14314368.30	90931559.00	9

Table A.18 Preliminary Results for B&B for N=15, T=20, (*min,max*)=(2,6), C=8, D=3

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	1965018.70	6145828.00	130.47	403.50	41002.20	356373.00	0
BB_2	1979770.30	6145957.00	106.66	327.22	41178.90	356452.00	0
BB_3	1979770.30	6145957.00	106.91	329.35	41178.90	356452.00	0
BB_4	1977319.00	6145828.00	119.10	364.03	41002.20	356373.00	0
BB_5	1966591.90	6145828.00	129.77	401.95	41002.20	356373.00	0
BB_6	1984815.10	6145828.00	118.52	365.08	41002.20	356373.00	0
BB_7	1997749.50	6145828.00	118.14	357.90	41002.20	356373.00	0
BB_8	1987022.40	6145828.00	128.91	395.22	41002.20	356373.00	0
BB_9	185360185.60	213667471.00	7200.00	7200.00	449764.60	4109409.00	10
BB_{10}	2005245.60	6145828.00	117.68	358.87	41002.20	356373.00	0
BB_{11}	176698936.60	202838865.00	7200.00	7200.00	13642481.20	131329006.00	10
BB_{12}	197510948.70	223667907.00	7200.00	7200.00	527543.40	4809746.00	10
BB_{13}	189164154.80	216587429.00	7200.00	7200.00	479006.80	4394155.00	10
BB_{14}	247053615.50	274183071.00	7200.00	7200.00	630603.00	5743113.00	10

Table A.19 Preliminary Results for B&B for N=15, T=20,(*min,max*)=(2,6),C=10, D=1

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	339675.10	1585432.00	17.84	78.74	4321.50	27289.00	0
BB_2	339779.70	1585432.00	14.58	64.85	4430.40	27345.00	0
BB_3	339779.70	1585432.00	14.54	64.61	4430.40	27345.00	0
BB_4	339675.10	1585432.00	17.83	78.66	4321.50	27289.00	0
BB_5	339675.10	1585432.00	17.76	78.31	4321.50	27289.00	0
BB_6	339675.10	1585432.00	17.76	78.37	4321.50	27289.00	0
BB_7	339675.10	1585432.00	17.46	76.83	4321.50	27289.00	0
BB_8	339675.10	1585432.00	17.40	76.51	4321.50	27289.00	0
BB_9	147466755.60	279643456.00	4334.43	7200.00	1038887.70	3897159.00	6
BB_{10}	339675.10	1585432.00	17.40	76.50	4321.50	27289.00	0
BB_{11}	113828964.70	216233452.00	3914.55	7200.00	32008085.40	160502143.00	4
BB_{12}	147486443.40	279646257.00	4334.43	7200.00	1038887.70	3897159.00	6
BB_{13}	149932555.50	284285935.00	4334.44	7200.00	1053996.50	3956519.00	6
BB_{14}	170302416.70	317526539.00	4346.09	7200.00	1684367.30	9395738.00	6

Table A.20 Preliminary Results for B&B for N=15, T=20,(*min,max*)=(2,6),C=10, D=2

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	177911.60	699792.00	10.71	39.71	35933.60	190222.00	0
BB_2	183729.40	699857.00	8.95	32.35	36434.00	190250.00	0
BB_3	183729.40	699857.00	8.96	32.38	36434.00	190250.00	0
BB_4	177911.60	699792.00	10.71	39.70	35933.60	190222.00	0
BB_5	177911.60	699792.00	10.67	39.58	35933.60	190222.00	0
BB_6	177911.60	699792.00	10.69	39.63	35933.60	190222.00	0
BB_7	208611.60	699792.00	12.30	38.93	40554.50	190222.00	0
BB_8	208611.60	699792.00	12.26	38.83	40554.50	190222.00	0
BB_9	141288918.50	259721458.00	4360.60	7200.00	1276920.40	11734346.00	6
BB_{10}	208611.60	699792.00	12.27	38.87	40554.50	190222.00	0
BB_{11}	114286939.30	219663570.00	4166.17	7200.00	16666695.70	161262290.00	5
BB_{12}	141305302.90	259762306.00	4360.59	7200.00	1276920.40	11734346.00	6
BB_{13}	143469036.00	263190497.00	4360.70	7200.00	1301214.50	11929206.00	6
BB_{14}	162322469.70	294307011.00	4355.96	7200.00	1640463.60	11935096.00	6

Table A.21 Preliminary Results for B&B for N=15, T=20,(*min,max*)=(2,6),C=10, D=3

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	594925.20	1711715.00	34.05	96.62	173983.80	1711712.00	0
BB_2	632616.20	1874066.00	29.16	88.27	174078.40	1711925.00	0
BB_3	632616.20	1874066.00	29.15	88.24	174078.40	1711925.00	0
BB_4	594925.20	1711715.00	34.01	96.38	173983.80	1711712.00	0
BB_5	594925.20	1711715.00	33.88	96.07	173983.80	1711712.00	0
BB_6	594925.20	1711715.00	33.87	96.06	173983.80	1711712.00	0
BB_7	657209.80	1992780.00	36.90	115.18	174003.00	1711904.00	0
BB_8	657209.80	1992780.00	36.77	114.86	174003.00	1711904.00	0
BB_9	155536675.70	244459703.00	5050.07	7200.00	750679.80	3857579.00	7
BB_{10}	657209.80	1992780.00	36.78	114.83	174003.00	1711904.00	0
BB_{11}	132302881.00	205979420.00	5047.68	7200.00	503805.30	2602518.00	7
BB_{12}	155522982.90	244417373.00	5050.08	7200.00	750679.80	3857579.00	7
BB_{13}	157729404.00	247520594.00	5049.98	7200.00	769851.80	4029039.00	7
BB_{14}	181584197.70	285537922.00	5049.20	7200.00	771433.10	4029076.00	7

Table A.22 Preliminary Results for B&B for N=15, T=20,(*min,max*)=(2,6),C=12, D=1

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	35.60	171.00	0.00	0.01	23.50	168.00	0
BB_2	119.80	264.00	0.01	0.04	115.00	264.00	0
BB_3	119.80	264.00	0.01	0.02	115.00	264.00	0
BB_4	35.60	171.00	0.00	0.01	23.50	168.00	0
BB_5	35.60	171.00	0.00	0.02	23.50	168.00	0
BB_6	35.60	171.00	0.00	0.01	23.50	168.00	0
BB_7	35.60	171.00	0.01	0.09	23.50	168.00	0
BB_8	35.60	171.00	0.00	0.01	23.50	168.00	0
BB_9	146103.40	1460849.00	4.17	41.68	146091.30	1460846.00	0
BB_{10}	35.60	171.00	0.00	0.01	23.50	168.00	0
BB_{11}	89846.00	898275.00	2.92	29.15	89833.90	898272.00	0
BB_{12}	146103.40	1460849.00	4.17	41.64	146091.30	1460846.00	0
BB_{13}	146103.40	1460849.00	4.10	41.05	146091.30	1460846.00	0
BB_{14}	185471.90	1516232.00	4.48	37.11	185467.10	1516232.00	0

Table A.23 Preliminary Results for B&B for N=15, T=20,(*min,max*)=(2,6),C=12, D=2

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	17825.30	178000.00	1.06	10.58	17813.80	177997.00	0
BB_2	20246.60	200209.00	0.96	9.45	20241.80	200209.00	0
BB_3	20246.60	200209.00	0.96	9.45	20241.80	200209.00	0
BB_4	17825.30	178000.00	1.06	10.58	17813.80	177997.00	0
BB_5	17825.30	178000.00	1.06	10.54	17813.80	177997.00	0
BB_6	17825.30	178000.00	1.06	10.57	17813.80	177997.00	0
BB_7	24186.00	241607.00	1.39	13.86	24174.50	241604.00	0
BB_8	24186.00	241607.00	1.38	13.81	24174.50	241604.00	0
BB_9	22857894.90	228570084.00	720.03	7200.00	875.30	8733.00	1
BB_{10}	24186.00	241607.00	1.38	13.82	24174.50	241604.00	0
BB_{11}	19620470.40	196198490.00	720.02	7200.00	610.20	6082.00	1
BB_{12}	22859408.30	228585218.00	720.03	7200.00	875.30	8733.00	1
BB_{13}	23183039.80	231821533.00	720.03	7200.00	875.30	8733.00	1
BB_{14}	26474684.40	261255684.00	728.53	7200.00	349111.20	3406730.00	1

Table A.24 Preliminary Results for B&B for N=15, T=20,(*min,max*)=(2,6),C=12, D=3

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	511.70	3152.00	0.03	0.16	499.20	3146.00	0
BB_2	586.70	3207.00	0.03	0.14	577.10	3207.00	0
BB_3	586.70	3207.00	0.03	0.13	577.10	3207.00	0
BB_4	511.70	3152.00	0.03	0.17	499.20	3146.00	0
BB_5	511.70	3152.00	0.03	0.16	499.20	3146.00	0
BB_6	511.70	3152.00	0.03	0.17	499.20	3146.00	0
BB_7	1046.60	8329.00	0.05	0.40	1034.10	8323.00	0
BB_8	1046.60	8329.00	0.05	0.41	1034.10	8323.00	0
BB_9	6954507.50	56340189.00	186.47	1466.92	6954495.00	56340183.00	0
BB_{10}	1046.60	8329.00	0.05	0.40	1034.10	8323.00	0
BB_{11}	1063938.30	5413005.00	34.67	183.45	1063925.80	5412999.00	0
BB_{12}	6954507.50	56340189.00	186.48	1467.16	6954495.00	56340183.00	0
BB_{13}	6954507.50	56340189.00	184.06	1448.03	6954495.00	56340183.00	0
BB_{14}	6968927.40	56340224.00	170.78	1345.61	6968917.80	56340224.00	0

Table A.25 Preliminary Results for B&B for N=15, T=20,(*min,max*)=(2,6),C=10, D=6

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	12750201.20	78704122.00	553.52	3067.36	9826911.30	78703347.00	0
BB_2	12763613.20	78704145.00	462.23	2601.57	9827801.90	78703373.00	0
BB_3	12763613.20	78704145.00	462.86	2606.41	9827801.90	78703373.00	0
BB_4	12750201.20	78704122.00	551.74	3066.44	9826911.30	78703347.00	0
BB_5	12750201.20	78704122.00	549.83	3044.11	9826911.30	78703347.00	0
BB_6	12750201.20	78704122.00	548.14	3044.22	9826911.30	78703347.00	0
BB_7	13820876.00	88795090.00	571.47	3302.38	10835810.60	88792340.00	0
BB_8	13820876.00	88795090.00	569.45	3277.64	10835810.60	88792340.00	0
BB_9	169176908.80	254112498.00	5760.00	7200.00	2.60	8.00	8
BB_{10}	13820876.00	88795090.00	567.57	3276.20	10835810.60	88792340.00	0
BB_{11}	134384635.60	232161387.00	5076.93	7200.00	2.60	8.00	7
BB_{12}	170299241.00	254072246.00	5760.00	7200.00	2.60	8.00	8
BB_{13}	171072244.60	257246156.00	5760.00	7200.00	2.60	8.00	8
BB_{14}	200931901.00	282047798.00	5760.19	7200.00	24821.70	177824.00	8

Table A.26 Preliminary Results for B&B for N=15, T=20,(*min,max*)=(2,6),C=12, D=6

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	13719.80	136800.00	0.83	8.25	31.00	180.00	0
BB_2	13884.80	136800.00	0.67	6.60	203.20	681.00	0
BB_3	13884.80	136800.00	0.67	6.61	203.20	681.00	0
BB_4	13719.80	136800.00	0.83	8.24	31.00	180.00	0
BB_5	13719.80	136800.00	0.83	8.22	31.00	180.00	0
BB_6	13719.80	136800.00	0.83	8.23	31.00	180.00	0
BB_7	2057257.80	20572180.00	104.78	1047.74	31.00	180.00	0
BB_8	2057257.80	20572180.00	104.44	1044.40	31.00	180.00	0
BB_9	23510560.70	234310622.00	722.50	7200.00	79489.70	794767.00	1
BB_{10}	2057257.80	20572180.00	104.35	1043.51	31.00	180.00	0
BB_{11}	7610251.20	75755786.00	273.07	2718.10	34663.80	346508.00	0
BB_{12}	23491335.20	234118367.00	722.50	7200.00	79489.70	794767.00	1
BB_{13}	23712053.80	236325553.00	722.48	7200.00	79489.70	794767.00	1
BB_{14}	58188857.90	298454504.00	1442.30	7200.00	81412.50	794840.00	2

Table A.27 Preliminary Results for B&B for N=10,T=15,(*min,max*)=(2,10),C=10,D=2

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	29650.60	94352.00	0.99	3.12	3412.80	19025.00	0
BB_2	29796.80	94354.00	0.82	2.57	3595.30	19068.00	0
BB_3	29796.80	94354.00	0.82	2.53	3595.50	19068.00	0
BB_4	31643.40	101687.00	0.95	3.02	3663.90	19025.00	0
BB_5	29750.50	94940.00	0.99	3.07	3415.10	19028.00	0
BB_6	34519.00	107634.00	1.00	3.08	3926.40	19455.00	0
BB_7	31646.20	101687.00	0.91	2.89	3666.70	19053.00	0
BB_8	29753.30	94940.00	0.94	2.95	3417.90	19056.00	0
BB_9	322690.50	778502.00	7.96	17.21	20125.70	153540.00	0
BB_{10}	34527.80	107634.00	0.96	2.97	3929.20	19483.00	0
BB_{11}	585993.90	1101518.00	13.96	26.74	37851.40	289473.00	0
BB_{12}	417555.30	995277.00	9.52	22.70	27568.10	203419.00	0
BB_{13}	348509.60	922413.00	8.35	18.61	20200.80	153714.00	0
BB_{14}	1060738.60	1996807.00	18.75	36.22	83791.90	654186.00	0

Table A.28 Preliminary Results for B&B for N=10,T=15,(*min,max*)=(2,10),C=10,D=5

	Total # of Nodes		CPU Time (Sec.)		Opt. Node		#
	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
BB_1	220625.80	684173.00	6.80	19.35	5.40	22.00	0
BB_2	220630.20	684173.00	5.67	16.29	22.80	63.00	0
BB_3	220630.20	684173.00	5.66	16.24	22.80	63.00	0
BB_4	257389.40	807133.00	7.31	21.16	5.40	22.00	0
BB_5	224412.60	684173.00	6.83	19.02	5.40	22.00	0
BB_6	468626.70	1145443.00	12.29	29.74	5.40	22.00	0
BB_7	260529.30	807133.00	6.98	19.88	5.40	22.00	0
BB_8	227552.50	684173.00	6.51	17.76	5.40	22.00	0
BB_9	554327.70	1191493.00	13.31	27.81	5.40	22.00	0
BB_{10}	471766.60	1145443.00	11.80	28.25	5.40	22.00	0
BB_{11}	1676004.90	3030630.00	39.95	75.34	5.40	22.00	0
BB_{12}	707493.10	1640037.00	16.32	38.38	5.40	22.00	0
BB_{13}	565522.30	1191493.00	13.43	27.38	5.40	22.00	0
BB_{14}	2072225.70	3641873.00	37.97	70.73	30.00	135.00	0

APPENDIX B

COMPUTATIONAL RESULTS FOR BRANCH-AND-BOUND

In this appendix, we provide the computational results for the full Branch-and-Bound experiments. The number of tools, T, tool magazine capacity, C, tool transporter capacity, D, and the parameters of job tool matrix , (min, max), used in these experiments are given in Table 6.2. We use 4 different number of job, N values for each of the combination presented in Table 6.2 and we generate 10 problem instances for each setting.

Table B.1 Branch and Bound Results for N=10

	Setting				Total # of nodes		CPU Time (sec.)		# Opt./ Best Node		#
N	*T*	*(min, max)*	*C*	*D*	*Avg.*	*Max.*	*Avg.*	*Max.*	*Avg.*	*Max.*	*unsolved*
10	10	(2,5)	5	2	7472.7	20726	0.20	0.44	113.9	850	0
10	10	(2,5)	5	4	58470.3	280752	1.32	5.89	28.0	251	0
10	15	(2,5)	5	2	2781.0	5508	0.10	0.20	183.1	942	0
10	15	(2,5)	5	4	43831.7	145663	1.17	3.59	15287.3	143168	0
10	15	(2,10)	10	2	18424.0	50230	0.54	1.47	37.6	103	0
10	15	(2,10)	10	5	178031.8	407713	4.67	10.64	8.9	55	0
10	15	(2,10)	10	8	75692.5	215482	2.18	5.72	82.5	662	0
10	15	(5,10)	10	2	17644.6	68446	0.55	1.84	2178.9	13590	0
10	15	(5,10)	10	5	200098.3	511673	5.24	13.02	0.8	3	0
10	15	(5,10)	10	8	131957.9	241455	3.61	6.59	3.5	14	0
10	15	(2,5)	10	2	11.0	11	0.00	0.01	0.0	0	0
10	15	(2,5)	10	4	11.0	11	0.00	0.00	0.2	2	0
10	20	(2,5)	5	2	3248.9	9231	0.11	0.30	898.9	4936	0
10	20	(2,5)	5	4	20304.4	78363	0.60	2.19	3.1	7	0
10	20	(2,10)	10	2	15566.5	69796	0.53	2.06	778.8	4248	0
10	20	(2,10)	10	5	38343.8	76028	1.31	2.53	884.2	8604	0
10	20	(2,10)	10	8	142993.6	323501	4.48	10.53	30284.9	298616	0
10	20	(5,10)	10	2	14627.5	28109	0.59	1.22	3552.6	16582	0
10	20	(5,10)	10	5	73290.2	321630	2.44	10.47	647.8	6401	0
10	20	(5,10)	10	8	84876.3	396684	2.77	12.84	10315.7	63361	0
10	20	(2,15)	15	2	24183.6	71537	0.80	2.24	11283.7	67397	0
10	20	(2,15)	15	5	130161.0	557591	3.95	16.39	1114.2	9074	0
10	20	(2,15)	15	8	108856.3	440041	3.32	12.17	2990.9	24781	0
10	20	(5,15)	15	2	23864.6	44634	0.84	1.55	6691.3	24429	0
10	20	(5,15)	15	5	263461.4	442343	8.08	13.33	257.9	2551	0
10	20	(5,15)	15	8	110294.3	241391	3.59	7.80	33207.3	123992	0
10	20	(2,5)	10	2	11.0	11	0.00	0.00	0.2	2	0
10	20	(2,5)	10	4	11.0	11	0.00	0.00	0.2	2	0
10	20	(2,5)	15	2	11.0	11	0.00	0.02	0.0	0	0
10	20	(2,5)	15	4	11.0	11	0.00	0.00	0.0	0	0
10	20	(2,10)	15	2	805.8	7959	0.03	0.27	1.0	3	0
10	20	(2,10)	15	5	2755.0	17021	0.09	0.52	7.5	66	0
10	20	(2,10)	15	8	9339.3	48678	0.28	1.44	3389.4	33794	0
10	25	(2,5)	5	2	4350.8	25680	0.16	0.84	2589.9	24812	0
10	25	(2,5)	5	4	8604.6	23237	0.29	0.80	2050.8	20469	0
10	25	(2,10)	10	2	13566.3	46690	0.51	1.58	2467.5	18035	0
10	25	(2,10)	10	5	22009.6	64936	0.83	2.45	2331.1	22317	0
10	25	(2,10)	10	8	128988.5	302851	4.36	10.69	12942.7	129406	0
10	25	(5,10)	10	2	19559.6	40404	0.87	1.81	8600.2	27247	0
10	25	(5,10)	10	5	13612.4	70225	0.57	2.63	16.2	81	0
10	25	(5,10)	10	8	51069.6	213979	1.94	7.78	28238.7	168876	0
10	25	(2,15)	15	2	31706.2	121387	1.28	5.27	12563.2	78756	0
10	25	(2,15)	15	5	35866.9	72091	1.44	2.80	445.9	3730	0
10	25	(2,15)	15	8	90037.9	410821	3.25	14.09	5241.0	37370	0
10	25	(5,15)	15	2	34989.1	76080	1.49	3.06	1737.4	8656	0
10	25	(5,15)	15	5	77677.8	153644	3.19	6.14	5.8	26	0
10	25	(5,15)	15	8	184531.5	752325	7.23	28.89	1719.6	17098	0
10	25	(2,5)	10	2	11.0	11	0.00	0.01	0.0	0	0
10	25	(2,5)	10	4	11.0	11	0.00	0.01	0.2	2	0
10	25	(2,5)	15	2	11.0	11	0.00	0.00	0.0	0	0
10	25	(2,5)	15	4	11.0	11	0.00	0.00	0.0	0	0
10	25	(2,5)	20	2	11.0	11	0.00	0.00	0.0	0	0
10	25	(2,5)	20	4	11.0	11	0.00	0.01	0.0	0	0
10	25	(2,10)	15	2	29.0	59	0.00	0.00	22.6	59	0
10	25	(2,10)	15	5	7410.1	36143	0.26	1.11	0.6	4	0
10	25	(2,10)	15	8	25233.7	192657	0.81	6.16	2.8	12	0
10	25	(2,10)	20	2	11.0	11	0.00	0.00	0.0	0	0
10	25	(2,10)	20	5	11.0	11	0.00	0.00	0.0	0	0
10	25	(2,10)	20	8	11.0	11	0.00	0.01	1.5	10	0
10	25	(5,10)	20	2	11.0	11	0.00	0.01	0.2	2	0
10	25	(5,10)	20	5	12.7	28	0.00	0.01	3.4	24	0
10	25	(5,10)	20	8	131.0	1031	0.01	0.05	19.9	184	0

Table B.2 Branch and Bound Results for N=15

	Setting				Total # of nodes		CPU Time (sec.)		# Opt. Node		#
N	***T***	***(min, max)***	***C***	***D***	***Avg.***	***Max.***	***Avg.***	***Max.***	***Avg.***	***Max.***	***unsolved***
15	10	(2,5)	5	2	25125688.8	76859863	972.61	3022.78	6534651.1	37758625	0
15	10	(2,5)	5	4	222603588.2	284292328	7020.08	7200.00	26440.8	264077	9
15	15	(2,5)	5	2	9294143.8	40487705	482.06	1958.20	5476251.5	39453314	0
15	15	(2,10)	10	2	48471285.4	143656491	2504.42	7200.00	1419090.9	7323467	1
15	15	(5,10)	10	2	59770709.9	144539540	3153.64	7200.00	126544.7	537286	1
15	15	(2,5)	10	2	16.0	16	0.00	0.01	2.0	7	0
15	15	(2,5)	10	4	2145999.6	21411931	71.71	715.47	4803.6	19701	0
15	20	(2,5)	5	2	4537710.9	9297856	251.03	504.47	425377.2	1848928	0
15	20	(2,5)	5	4	120284686.8	201198043	5363.41	7200.00	12546978.2	98045867	6
15	20	(2,10)	10	2	99410196.5	161159256	5537.90	7200.00	17182332.1	131234493	5
15	20	(5,10)	10	2	64370569.9	119177660	4056.29	7200.00	21742489.8	91675380	2
15	20	(2,15)	15	2	63090033.4	137358077	3304.34	7200.00	10632465.8	81532611	2
15	20	(5,15)	15	2	84281765.9	154223118	4565.53	7200.00	15436580.6	47595655	2
15	20	(2,5)	10	2	92231.4	920608	4.20	41.90	160.4	1469	0
15	20	(2,5)	10	4	337553.2	1455738	13.39	51.02	319987.4	1455738	0
15	20	(2,5)	15	2	16.0	16	0.00	0.00	0.0	0	0
15	20	(2,5)	15	4	16.0	16	0.00	0.01	0.0	0	0
15	20	(2,10)	15	2	7970040.9	74842806	297.32	2754.83	55895.3	351682	0
15	20	(2,10)	15	5	22798786.1	198446526	847.50	7200.00	103616.5	1036111	1
15	20	(2,10)	15	8	63807193.9	202719696	2422.61	7200.00	50107.0	275660	2
15	25	(2,5)	5	2	2035475.5	4898631	131.14	325.97	662292.7	4447385	0
15	25	(2,5)	5	4	110869055.6	161095098	5413.09	7200.00	19722117.9	142218959	5
15	25	(2,10)	10	2	40414956.4	119390632	2322.46	7200.00	5466060.1	44383626	1
15	25	(2,10)	10	5	59170431.6	131343821	3712.33	7200.00	736768.3	3836856	3
15	25	(5,10)	10	2	35955355.3	104211448	2615.13	7200.00	3050988.0	23055529	1
15	25	(5,10)	10	5	85173991.0	121440501	5489.45	7200.00	342121.6	3419911	5
15	25	(2,15)	15	2	65295781.8	119548363	4250.54	7200.00	2933347.9	12640060	4
15	25	(2,15)	15	5	104925159.6	133225515	6503.54	7200.00	1527093.1	11001757	8
15	25	(2,5)	10	2	153.8	1394	0.01	0.07	140.4	1394	0
15	25	(2,5)	10	4	331.4	1948	0.02	0.09	322.3	1948	0
15	25	(2,5)	15	2	16.0	16	0.00	0.00	0.0	0	0
15	25	(2,5)	15	4	16.0	16	0.00	0.01	0.0	0	0
15	25	(2,5)	20	2	16.0	16	0.00	0.01	0.0	0	0
15	25	(2,5)	20	4	16.0	16	0.00	0.01	0.0	0	0
15	25	(2,10)	15	2	606403.0	2252908	35.98	123.38	7981.6	58411	0
15	25	(2,10)	15	5	9506059.8	58509068	504.54	2994.22	115556.3	1097328	0
15	25	(2,10)	15	8	46885357.1	150557058	2312.33	7200.00	1037932.5	8790217	3
15	25	(2,10)	20	2	16.0	16	0.00	0.01	0.6	4	0
15	25	(2,10)	20	5	601277.8	5968570	32.50	322.85	4414.2	39259	0
15	25	(2,10)	20	8	7425.8	24337	0.55	1.72	1496.0	14960	0
15	25	(5,10)	20	2	85.9	715	0.01	0.04	72.8	715	0
15	25	(5,10)	20	5	1235023.1	3388344	74.48	191.15	1784.0	16857	0
15	25	(5,10)	20	8	322.3	2975	0.02	0.15	310.9	2975	0

Table B.3 Branch and Bound Results for N=20

	Setting				Total # of nodes		CPU Time (sec.)		# Opt. Node		#
N	T	(min, max)	C	D	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
20	15	(2,5)	10	2	32745614.0	190185166	1497.41	7200.00	1326190.7	12872036	2
20	15	(2,5)	10	4	127514577.0	188006375	5820.75	7200.00	4826.6	47995	8
20	20	(2,5)	10	2	30721967.3	122209972	1918.96	7200.00	2461.6	12501	2
20	20	(2,5)	10	4	16531868.1	143649321	877.22	7200.00	221382.0	1941780	1
20	20	(2,5)	15	2	21.0	21	0.00	0.02	0.8	4	0
20	20	(2,5)	15	4	21.0	21	0.00	0.02	2.2	12	0
20	20	(2,10)	15	2	91864779.9	145028232	5876.85	7200.00	18852.2	99065	6
20	20	(2,10)	15	5	113449966.3	143923283	6617.49	7200.00	15289.2	125936	8
20	25	(2,5)	10	2	22683174.4	131107740	1460.10	7200.00	443791.0	4212304	1
20	25	(2,5)	10	4	9036454.5	49927106	731.90	4178.95	2278853.2	9979631	0
20	25	(2,5)	15	2	21.0	21	0.00	0.02	0.2	2	0
20	25	(2,5)	15	4	24.7	58	0.00	0.02	4.7	43	0
20	25	(2,5)	20	2	21.0	21	0.00	0.01	0.0	0	0
20	25	(2,5)	20	4	21.0	21	0.00	0.01	0.2	2	0
20	25	(2,10)	15	2	75412957.8	107368312	5969.93	7200.00	6891433.7	44401325	8
20	25	(2,10)	15	5	80978698.0	122314428	5836.13	7200.00	2777412.5	26342052	8
20	25	(2,10)	15	8	123679785.0	144119297	7200.00	7200.00	28662.2	285036	10
20	25	(2,10)	20	2	64.8	251	0.01	0.03	51.0	251	0
20	25	(2,10)	20	5	65180065.8	108180125	4714.05	7200.00	6160750.7	61607473	6
20	25	(2,10)	20	8	145097.6	1449853	7.60	75.89	145086.1	1449853	0
20	25	(5,10)	20	2	27414951.7	102647602	2161.17	7200.00	13269.0	110332	3
20	25	(5,10)	20	5	51853232.0	101004014	4354.55	7200.00	75.8	516	4

Table B.4 Branch and Bound Results for N=25

	Setting				Total # of nodes		CPU Time (sec.)		# Opt. Node		#
N	T	(min, max)	C	D	Avg.	Max.	Avg.	Max.	Avg.	Max.	unsolved
25	20	(2,5)	10	4	66677799.9	154370761	4669.00	7200.00	7370920.2	43910492	6
25	25	(2,5)	20	2	26.0	26	0.01	0.02	0.0	0	0
25	25	(2,5)	20	4	26.0	26	0.00	0.02	0.0	0	0
25	25	(5,10)	20	2	64711568.8	88641385	6480.06	7200.00	2887689.8	18649592	9
25	25	(5,10)	20	5	83267695.6	101680268	7200.00	7200.00	796666.5	7924895	10
25	25	(5,10)	20	8	81671426.1	107930697	7200.00	7200.00	300.1	2598	10
25	25	(2,10)	20	2	24187857.6	84465092	2160.10	7200.00	1414.2	8601	3
25	25	(2,10)	20	5	79324570.2	107587977	6480.02	7200.00	14.7	60	9
25	25	(2,10)	20	8	74052324.8	111603775	5760.02	7200.00	34.1	325	8

APPENDIX C

COMPUTATIONAL RESULTS FOR BEAM SEARCH EXPERIMENTS

In this appendix, we give the results of our preliminary experiments for Beam Search algorithms. The parameter values used in the preliminary Beam Search experiments are given in Table 6.16. The beam width values tested in these experiments are $1N$, $2N$, $3N$, $4N$, and $5N$; the results for beam width of $5N$ are also summarized in main text. These experiments are also carried out to see the effects of various beam evaluation functions, upper bounds; UB_1, UB_2, lower bounds; LB_1, LB_2, LB_3, LB_4, all lower bounds in combination, namely LB_s, and a simple priority rule as F_1 & F_2.

Table C.1 Effect of β on the Beam Search Algorithm using UB_1 as Beam Evaluation Function

		Parallel Beam Search with UB_1							Pooled Beam Search with UB_1						
		% dev.		Total # of Nodes		CPU Time (sec.)		#	% dev.		Total # of Nodes		CPU Time (sec.)		#
	β	Avg.	Max.	Avg.	Max.	Avg.	Max.	opt.	Avg.	Max.	Avg.	Max.	Avg.	Max.	opt.
PS_1	1N	2.37	8.33	436.8	440.0	0.012	0.020	7	2.37	8.33	449.2	450.0	0.015	0.020	7
	2N	2.37	8.33	785.8	790.0	0.023	0.030	7	2.37	8.33	797.6	800.0	0.028	0.030	7
	3N	1.60	8.33	1130.5	1140.0	0.030	0.030	8	1.54	7.69	1147.8	1150.0	0.043	0.050	8
	4N	1.60	8.33	1466.3	1486.0	0.045	0.050	8	0.00	0.00	1496.9	1500.0	0.074	0.080	10
	5N	1.60	8.33	1811.4	1836.0	0.052	0.070	8	0.00	0.00	1844.0	1850.0	0.112	0.120	10
PS_2	1N	4.46	8.33	1537.8	1560.0	0.116	0.130	2	3.27	7.14	1562.0	1575.0	0.127	0.140	3
	2N	3.03	8.33	2868.2	2910.0	0.218	0.240	4	2.60	4.17	2901.5	2925.0	0.288	0.310	3
	3N	3.03	8.33	4198.3	4260.0	0.311	0.350	4	2.24	4.17	4239.1	4275.0	0.643	0.711	4
	4N	3.03	8.33	5527.5	5610.0	0.403	0.470	4	2.24	4.17	5573.5	5625.0	1.334	1.432	4
	5N	3.03	8.33	6848.8	6950.0	0.497	0.550	4	2.24	4.17	6909.1	6975.0	2.556	2.753	4
PS_3	1N	3.17	9.09	1531.0	1560.0	0.113	0.120	4	3.57	9.09	1558.1	1575.0	0.122	0.130	4
	2N	3.57	9.09	2858.2	2910.0	0.200	0.220	4	2.67	8.00	2889.5	2925.0	0.278	0.300	5
	3N	3.57	9.09	4178.7	4254.0	0.289	0.320	4	2.19	8.00	4221.4	4275.0	0.624	0.660	6
	4N	3.57	9.09	5489.9	5604.0	0.377	0.420	4	1.79	5.00	5554.1	5625.0	1.317	1.412	6
	5N	3.17	9.09	6805.7	6936.0	0.467	0.510	4	1.39	5.00	6919.4	6975.0	2.521	2.703	7
PS_4	1N	0.00	0.00	428.9	440.0	0.015	0.020	10	0.00	0.00	443.6	450.0	0.015	0.020	10
	2N	0.00	0.00	767.5	790.0	0.025	0.030	10	0.00	0.00	781.5	800.0	0.028	0.030	10
	3N	0.00	0.00	1113.4	1140.0	0.035	0.040	10	0.00	0.00	1139.1	1150.0	0.043	0.050	10
	4N	0.00	0.00	1446.4	1490.0	0.042	0.050	10	0.00	0.00	1477.1	1500.0	0.066	0.070	10
	5N	0.00	0.00	1788.9	1840.0	0.057	0.100	10	0.00	0.00	1831.3	1850.0	0.095	0.110	10
PS_5	1N	4.76	12.50	1529.1	1554.0	0.123	0.140	5	3.85	12.50	1548.9	1575.0	0.129	0.150	6
	2N	3.85	12.50	2835.5	2896.0	0.220	0.250	6	3.85	12.50	2877.2	2925.0	0.287	0.320	6
	3N	3.85	12.50	4151.1	4245.0	0.317	0.350	6	3.85	12.50	4195.2	4275.0	0.587	0.670	6
	4N	3.85	12.50	5447.7	5565.0	0.418	0.470	6	3.85	12.50	5539.2	5625.0	1.189	1.351	6
	5N	3.85	12.50	6738.3	6913.0	0.517	0.590	6	3.85	12.50	6866.3	6975.0	2.232	2.543	6
PS_6	1N	2.50	12.50	1504.1	1549.0	0.116	0.120	8	4.75	12.50	1555.6	1575.0	0.126	0.140	6
	2N	2.50	12.50	2788.5	2899.0	0.210	0.240	8	3.75	12.50	2887.4	2925.0	0.267	0.290	7
	3N	1.25	12.50	4038.8	4241.0	0.296	0.330	9	3.75	12.50	4219.9	4275.0	0.553	0.660	7
	4N	1.25	12.50	5295.2	5552.0	0.389	0.430	9	3.75	12.50	5551.9	5625.0	1.067	1.271	7
	5N	1.25	12.50	6575.6	6886.0	0.479	0.550	9	3.75	12.50	6883.9	6975.0	1.951	2.313	7
PS_7	1N	2.00	20.00	1414.9	1535.0	0.109	0.130	9	2.00	20.00	1537.0	1575.0	0.121	0.140	9
	2N	0.00	0.00	2668.2	2797.0	0.201	0.220	10	2.00	20.00	2844.6	2925.0	0.241	0.280	9
	3N	0.00	0.00	3906.7	4083.0	0.287	0.310	10	2.00	20.00	4140.6	4275.0	0.448	0.530	9
	4N	0.00	0.00	5007.8	5375.0	0.367	0.420	10	0.00	0.00	5444.9	5625.0	0.841	1.001	10
	5N	0.00	0.00	6109.7	6640.0	0.448	0.500	10	0.00	0.00	6763.6	6975.0	1.469	1.742	10
PS_8	1N	0.00	0.00	1344.6	1454.0	0.102	0.120	10	0.00	0.00	1540.6	1575.0	0.117	0.130	10
	2N	0.00	0.00	2577.9	2779.0	0.189	0.220	10	0.00	0.00	2848.1	2925.0	0.239	0.270	10
	3N	0.00	0.00	3754.1	4044.0	0.270	0.300	10	0.00	0.00	4161.7	4275.0	0.439	0.510	10
	4N	0.00	0.00	4878.4	5325.0	0.350	0.400	10	0.00	0.00	5521.8	5625.0	0.782	0.931	10
	5N	0.00	0.00	5849.9	6339.0	0.414	0.470	10	0.00	0.00	6783.9	6975.0	1.385	1.772	10
PS_9	1N	6.53	14.29	1524.8	1560.0	0.083	0.090	4	8.41	14.29	1575.0	1575.0	0.095	0.100	2
	2N	5.53	14.29	2845.0	2910.0	0.152	0.160	5	7.50	14.29	2925.0	2925.0	0.214	0.230	3
	3N	5.53	14.29	4160.0	4241.0	0.223	0.250	5	3.37	14.29	4275.0	4275.0	0.481	0.540	7
	4N	5.53	14.29	5472.8	5586.0	0.283	0.320	5	1.94	11.11	5625.0	5625.0	1.017	1.151	8
	5N	5.53	14.29	6769.3	6933.0	0.351	0.380	5	1.94	11.11	6975.0	6975.0	1.980	2.213	8
PS_{10}	1N	7.01	12.50	1545.4	1558.0	0.146	0.160	2	5.62	12.50	1575.0	1575.0	0.156	0.170	3
	2N	6.51	12.50	2883.0	2904.0	0.265	0.290	2	4.50	6.67	2925.0	2925.0	0.338	0.370	2
	3N	5.99	12.50	4217.6	4248.0	0.387	0.420	2	3.45	6.67	4275.0	4275.0	0.702	0.761	4
	4N	5.99	12.50	5539.4	5596.0	0.505	0.560	2	2.29	6.67	5625.0	5625.0	1.388	1.542	6
	5N	5.43	12.50	6835.8	6941.0	0.619	0.680	3	1.07	6.25	6975.0	6975.0	2.602	2.974	7
PS_{11}	1N	4.86	20.00	1486.9	1544.0	0.141	0.160	7	4.86	20.00	1564.5	1575.0	0.148	0.170	7
	2N	4.86	20.00	2757.3	2873.0	0.252	0.290	7	4.86	20.00	2900.4	2925.0	0.305	0.350	7
	3N	4.86	20.00	4033.2	4186.0	0.367	0.430	7	2.86	14.29	4252.5	4275.0	0.563	0.610	8
	4N	4.86	20.00	5247.9	5520.0	0.473	0.570	7	4.86	20.00	5580.3	5625.0	1.016	1.141	7
	5N	4.86	20.00	6483.7	6776.0	0.585	0.691	7	4.86	20.00	6929.1	6975.0	1.743	2.032	7
PS_{12}	1N	5.15	9.09	3581.6	3706.0	0.417	0.440	4	6.06	9.09	3800.0	3800.0	0.510	0.540	3
	2N	2.50	8.33	6744.0	6928.0	0.780	0.811	7	4.39	9.09	7200.0	7200.0	1.517	1.592	5
	3N	2.50	8.33	9885.3	10194.0	1.123	1.171	7	3.48	9.09	10600.0	10600.0	4.000	4.316	6
	4N	0.83	8.33	12855.7	13441.0	1.449	1.532	9	3.48	9.09	14000.0	14000.0	9.042	9.764	6
	5N	0.83	8.33	15810.7	16691.0	1.787	1.892	9	3.48	9.09	17400.0	17400.0	17.093	18.326	6

Table C.2 Effect of β on the Beam Search Algorithm using UB_2 as Beam Evaluation Function

		Parallel Beam Search with UB_2							Pooled Beam Search with UB_2						
		% dev.		Total # of Nodes		CPU Time (sec.)		#	% dev.		Total # of Nodes		CPU Time (sec.)		#
	β	Avg.	Max.	Avg.	Max.	Avg.	Max.	opt.	Avg.	Max.	Avg.	Max.	Avg.	Max.	opt.
PS_1	1N	3.14	8.33	429.5	440.0	0.008	0.010	6	0.77	7.69	448.6	450.0	0.010	0.010	9
	2N	3.14	8.33	766.5	790.0	0.016	0.020	6	0.77	7.69	795.4	800.0	0.020	0.020	9
	3N	3.14	8.33	1096.9	1140.0	0.020	0.030	6	0.77	7.69	1144.2	1150.0	0.044	0.120	9
	4N	3.14	8.33	1431.5	1484.0	0.025	0.030	6	0.77	7.69	1493.9	1500.0	0.060	0.070	9
	5N	3.14	8.33	1765.4	1815.0	0.035	0.040	6	0.77	7.69	1841.5	1850.0	0.095	0.100	9
PS_2	1N	7.97	15.63	1514.1	1560.0	0.078	0.090	1	5.39	12.50	1548.5	1575.0	0.095	0.110	1
	2N	6.48	15.63	2822.8	2896.0	0.148	0.170	1	4.65	10.71	2877.7	2925.0	0.233	0.270	2
	3N	6.48	15.63	4130.7	4243.0	0.212	0.240	1	4.30	9.38	4206.6	4275.0	0.552	0.610	2
	4N	6.48	15.63	5427.7	5591.0	0.274	0.300	1	4.30	9.38	5533.6	5625.0	1.238	1.351	2
	5N	6.48	15.63	6727.7	6927.0	0.336	0.370	1	3.95	9.38	6861.0	6975.0	2.427	2.623	3
PS_3	1N	6.31	13.64	1519.4	1558.0	0.076	0.080	3	5.86	13.64	1556.7	1575.0	0.090	0.100	3
	2N	5.44	13.64	2822.8	2908.0	0.138	0.150	3	5.38	13.64	2888.9	2925.0	0.226	0.250	4
	3N	5.04	13.64	4125.2	4242.0	0.200	0.230	3	4.58	13.64	4221.3	4275.0	0.542	0.570	4
	4N	4.11	9.09	5414.6	5575.0	0.257	0.280	3	4.13	10.00	5553.4	5625.0	1.218	1.281	4
	5N	4.11	9.09	6700.2	6905.0	0.319	0.350	3	4.13	10.00	6885.3	6975.0	2.386	2.543	4
PS_4	1N	0.00	0.00	435.7	440.0	0.008	0.010	10	6.01	16.67	446.2	450.0	0.010	0.010	6
	2N	1.67	16.67	782.7	790.0	0.016	0.020	9	8.10	16.67	789.4	800.0	0.019	0.020	5
	3N	1.67	16.67	1123.8	1140.0	0.027	0.030	9	6.67	16.67	1135.6	1150.0	0.028	0.030	6
	4N	0.00	0.00	1462.9	1490.0	0.029	0.030	10	6.67	16.67	1486.8	1500.0	0.041	0.050	6
	5N	0.00	0.00	1808.8	1840.0	0.035	0.040	10	5.00	16.67	1832.9	1850.0	0.063	0.080	7
PS_5	1N	8.29	15.38	1493.9	1547.0	0.081	0.090	2	5.59	12.50	1548.8	1575.0	0.094	0.100	4
	2N	8.36	15.38	2795.4	2883.0	0.147	0.160	2	4.34	10.00	2875.9	2925.0	0.220	0.250	5
	3N	7.46	15.38	4070.9	4225.0	0.215	0.230	3	4.69	12.50	4202.9	4275.0	0.480	0.560	5
	4N	7.46	15.38	5355.6	5530.0	0.274	0.300	3	3.44	10.00	5524.6	5625.0	1.031	1.211	6
	5N	7.46	15.38	6644.4	6874.0	0.344	0.400	3	3.44	10.00	6858.3	6975.0	2.016	2.413	6
PS_6	1N	8.08	12.50	1494.6	1534.0	0.078	0.080	3	14.19	25.00	1555.4	1575.0	0.086	0.090	1
	2N	11.83	25.00	2809.2	2879.0	0.139	0.150	2	13.08	25.00	2881.5	2925.0	0.202	0.220	1
	3N	9.33	25.00	4061.7	4175.0	0.204	0.230	3	12.08	25.00	4219.9	4275.0	0.453	0.510	2
	4N	8.08	12.50	5288.8	5511.0	0.265	0.280	3	9.72	25.00	5552.1	5625.0	0.957	1.111	4
	5N	8.08	12.50	6572.5	6789.0	0.328	0.350	3	9.58	25.00	6884.1	6975.0	1.841	2.193	3
PS_7	1N	9.00	25.00	1478.8	1558.0	0.079	0.090	6	11.00	25.00	1520.3	1575.0	0.081	0.090	5
	2N	9.50	25.00	2744.2	2854.0	0.139	0.150	6	15.50	40.00	2853.8	2925.0	0.152	0.160	4
	3N	9.50	25.00	4016.7	4173.0	0.205	0.230	6	13.50	25.00	4162.3	4275.0	0.261	0.310	4
	4N	9.50	25.00	5300.6	5515.0	0.263	0.280	6	13.50	40.00	5485.0	5625.0	0.425	0.480	5
	5N	9.50	25.00	6543.3	6844.0	0.324	0.350	6	13.50	25.00	6767.2	6975.0	0.723	0.941	4
PS_8	1N	5.00	25.00	1421.2	1554.0	0.074	0.080	8	15.00	25.00	1543.4	1575.0	0.079	0.090	4
	2N	5.83	33.33	2662.5	2787.0	0.138	0.160	8	18.33	33.33	2867.0	2925.0	0.155	0.170	3
	3N	5.83	33.33	3907.1	4121.0	0.196	0.210	8	18.33	33.33	4161.7	4275.0	0.257	0.290	3
	4N	5.83	33.33	5074.0	5466.0	0.258	0.300	8	13.33	33.33	5519.8	5625.0	0.413	0.470	5
	5N	5.83	33.33	6271.0	6632.0	0.310	0.330	8	13.33	33.33	6830.6	6975.0	0.681	0.821	5
PS_9	1N	11.48	22.22	1527.0	1558.0	0.062	0.070	1	9.98	28.57	1574.9	1575.0	0.074	0.080	2
	2N	9.77	22.22	2820.7	2902.0	0.114	0.130	2	7.62	28.57	2924.3	2925.0	0.175	0.190	4
	3N	8.66	14.29	4108.5	4230.0	0.164	0.210	2	7.76	28.57	4269.5	4275.0	0.398	0.470	4
	4N	7.55	14.29	5390.7	5570.0	0.205	0.230	3	7.30	22.22	5624.4	5625.0	0.850	1.011	4
	5N	7.55	14.29	6687.5	6828.0	0.254	0.280	3	8.87	28.57	6972.6	6975.0	1.691	1.962	3
PS_{10}	1N	11.82	20.00	1529.2	1552.0	0.096	0.110	0	10.36	20.00	1575.0	1575.0	0.111	0.130	0
	2N	11.94	18.75	2855.4	2901.0	0.180	0.200	0	8.88	15.79	2925.0	2925.0	0.257	0.280	0
	3N	10.76	15.79	4155.1	4249.0	0.255	0.290	0	7.80	12.50	4275.0	4275.0	0.588	0.650	1
	4N	10.24	15.00	5462.4	5586.0	0.332	0.370	0	6.75	12.50	5625.0	5625.0	1.283	1.452	1
	5N	9.71	15.00	6734.1	6870.0	0.411	0.460	0	7.08	12.50	6975.0	6975.0	2.443	2.703	2
PS_{11}	1N	8.19	20.00	1474.6	1533.0	0.094	0.110	5	12.95	20.00	1561.8	1575.0	0.107	0.120	2
	2N	8.19	20.00	2767.9	2829.0	0.171	0.190	5	11.29	20.00	2895.8	2925.0	0.225	0.250	3
	3N	8.19	20.00	3994.6	4119.0	0.240	0.270	5	12.95	20.00	4240.5	4275.0	0.473	0.540	2
	4N	8.19	20.00	5271.2	5441.0	0.320	0.360	5	11.52	20.00	5584.1	5625.0	0.928	1.051	3
	5N	6.52	20.00	6381.3	6754.0	0.390	0.430	6	12.62	16.67	6937.7	6975.0	1.699	1.962	2
PS_{12}	1N	7.80	16.67	3576.9	3702.0	0.315	0.320	2	13.03	25.00	3772.4	3800.0	0.419	0.430	0
	2N	6.97	9.09	6810.6	7025.0	0.600	0.610	2	12.27	18.18	7160.1	7200.0	1.461	1.552	0
	3N	6.97	9.09	9835.4	10228.0	0.855	0.881	2	11.36	18.18	10539.2	10600.0	4.256	4.476	0
	4N	6.97	9.09	13058.3	13408.0	1.116	1.141	2	11.36	18.18	13912.0	14000.0	9.927	10.445	0
	5N	6.97	9.09	16003.3	16637.0	1.368	1.412	2	11.36	18.18	17313.4	17400.0	19.420	20.269	0

Table C.3 Effect of β on the Beam Search Algorithm using LB_s as Beam Evaluation Function

		Parallel Beam Search with all LBs							Pooled Beam Search with all LBs						
		% dev.		Total # of Nodes		CPU Time (sec.)		#	% dev.		Total # of Nodes		CPU Time (sec.)		#
	β	Avg.	Max.	Avg.	Max.	Avg.	Max.	opt.	Avg.	Max.	Avg.	Max.	Avg.	Max.	opt.
PS_1	1N	2.45	9.09	437.6	440.0	0.019	0.030	7	0.00	0.00	450.0	450.0	0.028	0.090	10
	2N	0.77	7.69	781.9	790.0	0.029	0.040	9	0.00	0.00	799.8	800.0	0.036	0.040	10
	3N	0.77	7.69	1127.5	1140.0	0.039	0.050	9	0.00	0.00	1149.9	1150.0	0.054	0.070	10
	4N	0.77	7.69	1472.8	1490.0	0.049	0.070	9	0.00	0.00	1499.3	1500.0	0.081	0.100	10
	5N	0.77	7.69	1820.4	1839.0	0.060	0.090	9	0.00	0.00	1849.5	1850.0	0.122	0.150	10
PS_2	1N	8.58	11.11	1541.3	1560.0	0.481	0.751	1	6.25	14.81	1572.3	1575.0	0.504	0.761	3
	2N	9.25	14.81	2876.1	2910.0	0.750	1.111	1	3.03	8.33	2916.5	2925.0	0.870	1.241	4
	3N	8.55	14.81	4206.5	4260.0	1.022	1.502	1	3.37	8.33	4260.6	4275.0	1.428	1.892	3
	4N	8.55	14.81	5540.5	5610.0	1.304	1.902	1	3.77	8.33	5615.2	5625.0	2.287	3.004	2
	5N	7.85	14.81	6870.8	6960.0	1.584	2.323	1	3.12	8.33	6962.9	6975.0	3.671	4.696	4
PS_3	1N	6.03	13.64	1539.5	1560.0	0.126	0.280	1	3.23	9.09	1572.3	1575.0	0.147	0.330	4
	2N	5.10	13.64	2866.9	2910.0	0.221	0.500	2	2.73	9.09	2919.1	2925.0	0.334	0.610	6
	3N	5.10	13.64	4195.2	4260.0	0.304	0.670	2	2.75	9.09	4264.2	4275.0	0.686	1.101	5
	4N	4.60	13.64	5522.7	5610.0	0.395	0.801	3	1.78	8.70	5613.2	5625.0	1.306	1.872	7
	5N	4.60	13.64	6852.1	6960.0	0.483	0.991	3	1.34	4.55	6911.7	6975.0	2.385	3.124	7
PS_4	1N	0.00	0.00	431.4	440.0	0.022	0.030	10	7.68	16.67	447.6	450.0	0.021	0.030	5
	2N	4.35	16.67	775.6	790.0	0.034	0.040	7	6.01	16.67	796.1	800.0	0.036	0.040	6
	3N	1.67	16.67	1110.6	1138.0	0.055	0.120	9	2.92	16.67	1147.7	1150.0	0.059	0.100	8
	4N	1.67	16.67	1451.5	1488.0	0.059	0.070	9	4.58	16.67	1494.4	1500.0	0.077	0.090	7
	5N	0.00	0.00	1782.0	1836.0	0.069	0.080	10	4.58	16.67	1827.4	1850.0	0.111	0.140	7
PS_5	1N	8.26	12.50	1526.1	1560.0	0.475	0.590	1	7.43	12.50	1569.5	1575.0	0.495	0.590	2
	2N	9.20	15.38	2855.2	2909.0	0.742	0.961	1	3.60	10.00	2918.3	2925.0	0.822	0.991	6
	3N	8.43	12.50	4165.4	4259.0	1.024	1.311	1	4.76	12.50	4258.5	4275.0	1.307	1.592	5
	4N	8.43	12.50	5482.4	5608.0	1.303	1.612	1	3.60	10.00	5600.1	5625.0	2.011	2.453	6
	5N	7.43	12.50	6793.1	6958.0	1.579	1.932	2	4.69	12.50	6961.8	6975.0	3.094	3.715	5
PS_6	1N	9.58	25.00	1523.2	1554.0	0.240	0.400	3	8.33	25.00	1554.1	1575.0	0.259	0.430	4
	2N	11.76	25.00	2848.3	2910.0	0.372	0.610	2	6.97	12.50	2887.9	2925.0	0.462	0.680	4
	3N	10.76	25.00	4157.3	4242.0	0.521	0.851	2	10.58	25.00	4266.5	4275.0	0.786	1.121	3
	4N	9.51	14.29	5458.6	5582.0	0.645	1.071	2	7.29	14.29	5613.5	5625.0	1.223	1.552	4
	5N	9.51	14.29	6764.4	6924.0	0.781	1.281	2	9.61	25.00	6954.2	6975.0	1.966	2.483	3
PS_7	1N	9.00	25.00	1469.0	1558.0	0.093	0.150	6	11.00	40.00	1533.0	1575.0	0.100	0.160	6
	2N	9.00	25.00	2725.4	2884.0	0.165	0.270	6	15.50	40.00	2837.0	2925.0	0.199	0.300	4
	3N	9.00	25.00	3986.4	4219.0	0.236	0.370	6	13.50	25.00	4128.0	4275.0	0.334	0.440	4
	4N	7.00	25.00	5214.4	5483.0	0.303	0.460	7	11.00	25.00	5463.9	5625.0	0.531	0.731	5
	5N	7.00	25.00	6451.1	6798.0	0.378	0.580	7	15.50	40.00	6793.2	6975.0	0.944	1.301	4
PS_8	1N	7.50	25.00	1418.3	1552.0	0.081	0.090	7	15.83	33.33	1549.4	1575.0	0.093	0.100	4
	2N	2.50	25.00	2585.7	2799.0	0.143	0.160	9	10.83	33.33	2873.5	2925.0	0.207	0.270	6
	3N	2.50	25.00	3772.0	4136.0	0.209	0.230	9	13.33	33.33	4171.1	4275.0	0.377	0.490	5
	4N	2.50	25.00	4850.2	5368.0	0.260	0.300	9	12.50	25.00	5533.4	5625.0	0.685	0.921	5
	5N	0.00	0.00	5872.9	6402.0	0.315	0.350	10	13.33	33.33	6871.6	6975.0	1.114	1.472	5
PS_9	1N	11.61	22.22	1545.2	1560.0	0.461	0.580	1	10.48	18.18	1574.6	1575.0	0.475	0.610	1
	2N	9.66	14.29	2873.9	2895.0	0.716	0.901	1	8.46	18.18	2923.2	2925.0	0.781	0.971	3
	3N	9.66	14.29	4204.3	4239.0	0.978	1.241	1	8.54	18.18	4274.0	4275.0	1.210	1.482	3
	4N	9.66	14.29	5521.4	5581.0	1.249	1.552	1	9.46	18.18	5624.1	5625.0	1.818	2.183	2
	5N	9.66	14.29	6842.9	6931.0	1.528	1.892	1	7.43	18.18	6973.0	6975.0	2.754	3.174	4
PS_{10}	1N	9.55	13.33	1557.8	1560.0	0.607	0.891	0	5.91	10.53	1575.0	1575.0	0.616	0.931	2
	2N	7.83	12.50	2905.3	2910.0	0.944	1.402	0	4.42	11.11	2925.0	2925.0	1.034	1.502	3
	3N	7.83	12.50	4251.3	4260.0	1.292	1.902	0	2.83	6.25	4275.0	4275.0	1.588	2.183	5
	4N	7.30	12.50	5589.2	5610.0	1.642	2.393	0	2.30	6.25	5625.0	5625.0	2.447	3.124	6
	5N	7.30	12.50	6934.9	6960.0	2.001	2.894	0	1.68	6.25	6975.0	6975.0	3.750	4.506	7
PS_{11}	1N	7.95	20.00	1539.3	1558.0	0.128	0.300	5	7.95	20.00	1573.0	1575.0	0.139	0.330	5
	2N	4.86	20.00	2846.3	2890.0	0.221	0.490	7	8.19	20.00	2919.0	2925.0	0.256	0.500	5
	3N	4.86	20.00	4156.0	4238.0	0.309	0.610	7	6.52	20.00	4264.6	4275.0	0.439	0.801	6
	4N	4.86	20.00	5479.3	5582.0	0.399	0.761	7	8.19	20.00	5625.0	5625.0	0.720	1.151	5
	5N	4.86	20.00	6779.4	6920.0	0.497	1.001	7	8.19	20.00	6967.0	6975.0	1.161	1.542	5
PS_{12}	1N	3.48	9.09	3726.3	3771.0	0.358	0.380	5	5.30	18.18	3800.0	3800.0	0.439	0.460	5
	2N	2.65	9.09	7010.9	7140.0	0.655	0.681	6	3.56	9.09	7200.0	7200.0	1.220	1.412	6
	3N	2.65	9.09	10339.5	10527.0	0.947	1.001	6	3.48	9.09	10594.8	10600.0	2.707	3.164	6
	4N	1.74	9.09	13597.9	13902.0	1.239	1.311	7	4.17	16.67	14000.0	14000.0	6.525	7.350	6
	5N	1.74	9.09	16935.1	17283.0	1.544	1.632	7	3.48	9.09	17400.0	17400.0	12.044	13.108	6

Table C.4 Effect of β on the Beam Search Algorithm using LB_1 as Beam Evaluation Function

		Parallel Beam Search with LB_1							Pooled Beam Search with LB_1						
		% dev.		Total # of Nodes		CPU Time (sec.)		#	% dev.		Total # of Nodes		CPU Time (sec.)		#
	β	Avg.	Max.	Avg.	Max.	Avg.	Max.	opt.	Avg.	Max.	Avg.	Max.	Avg.	Max.	opt.
PS_1	1N	3.22	9.09	438.3	440.0	0.010	0.010	6	0.77	7.69	450.0	450.0	0.011	0.020	9
	2N	1.54	7.69	783.7	790.0	0.019	0.020	8	0.00	0.00	799.8	800.0	0.021	0.030	10
	3N	1.54	7.69	1130.4	1140.0	0.031	0.070	8	0.00	0.00	1149.8	1150.0	0.039	0.050	10
	4N	1.54	7.69	1477.6	1490.0	0.031	0.040	8	0.00	0.00	1499.7	1500.0	0.063	0.070	10
	5N	1.54	7.69	1825.9	1840.0	0.039	0.050	8	0.00	0.00	1849.4	1850.0	0.100	0.110	10
PS_2	1N	9.64	15.63	1540.5	1560.0	0.087	0.100	1	5.55	12.50	1573.3	1575.0	0.096	0.110	3
	2N	9.64	15.63	2886.1	2910.0	0.160	0.180	1	4.16	11.11	2921.6	2925.0	0.266	0.290	3
	3N	8.19	14.81	4214.7	4260.0	0.231	0.280	1	3.80	11.11	4269.7	4275.0	0.619	0.711	3
	4N	8.19	14.81	5545.0	5610.0	0.296	0.330	1	4.56	11.11	5616.8	5625.0	1.342	1.492	3
	5N	7.80	14.81	6874.2	6960.0	0.366	0.400	1	4.56	11.11	6964.4	6975.0	2.644	2.994	3
PS_3	1N	6.58	13.64	1539.3	1560.0	0.080	0.090	1	2.77	9.09	1573.5	1575.0	0.095	0.100	5
	2N	5.10	13.64	2884.0	2910.0	0.150	0.160	2	2.77	9.09	2922.0	2925.0	0.259	0.280	5
	3N	5.10	13.64	4212.8	4260.0	0.211	0.230	2	2.77	9.09	4270.5	4275.0	0.635	0.711	5
	4N	5.10	13.64	5538.8	5610.0	0.281	0.290	2	1.80	9.09	5619.0	5625.0	1.371	1.522	7
	5N	5.10	13.64	6869.4	6960.0	0.339	0.360	2	1.80	9.09	6967.5	6975.0	2.660	2.994	7
PS_4	1N	0.00	0.00	433.2	440.0	0.010	0.010	10	6.67	16.67	446.9	450.0	0.012	0.020	6
	2N	1.67	16.67	780.7	790.0	0.018	0.020	9	8.33	16.67	796.8	800.0	0.019	0.020	5
	3N	1.67	16.67	1125.3	1140.0	0.027	0.030	9	3.33	16.67	1144.5	1150.0	0.032	0.040	8
	4N	0.00	0.00	1462.6	1490.0	0.033	0.040	10	5.00	16.67	1492.4	1500.0	0.046	0.050	7
	5N	0.00	0.00	1809.9	1840.0	0.045	0.050	10	3.33	16.67	1839.4	1850.0	0.070	0.140	8
PS_5	1N	9.03	15.38	1530.5	1560.0	0.086	0.090	1	9.86	16.67	1573.2	1575.0	0.094	0.100	1
	2N	9.03	15.38	2871.1	2910.0	0.160	0.170	1	6.04	15.38	2921.3	2925.0	0.224	0.260	4
	3N	8.20	15.38	4189.4	4260.0	0.229	0.250	2	6.52	12.50	4269.5	4275.0	0.485	0.610	3
	4N	7.43	12.50	5507.5	5609.0	0.300	0.330	2	3.60	10.00	5617.6	5625.0	1.013	1.241	6
	5N	7.43	12.50	6824.3	6958.0	0.371	0.410	2	5.85	12.50	6965.5	6975.0	1.952	2.673	4
PS_6	1N	9.58	25.00	1523.9	1554.0	0.079	0.080	3	8.08	25.00	1571.3	1575.0	0.092	0.110	4
	2N	11.76	25.00	2860.5	2910.0	0.152	0.160	2	5.72	12.50	2921.6	2925.0	0.210	0.230	5
	3N	10.76	25.00	4165.1	4242.0	0.213	0.230	2	10.58	25.00	4270.2	4275.0	0.447	0.560	3
	4N	9.51	14.29	5466.6	5582.0	0.282	0.300	2	6.04	14.29	5617.2	5625.0	0.855	1.181	5
	5N	9.51	14.29	6772.4	6924.0	0.347	0.360	2	7.11	25.00	6958.2	6975.0	1.600	2.253	5
PS_7	1N	9.00	25.00	1469.0	1558.0	0.081	0.100	6	11.00	40.00	1533.0	1575.0	0.086	0.090	6
	2N	9.00	25.00	2725.4	2884.0	0.143	0.160	6	15.50	40.00	2837.0	2925.0	0.176	0.200	4
	3N	9.00	25.00	3986.4	4219.0	0.208	0.220	6	13.50	25.00	4128.0	4275.0	0.310	0.380	4
	4N	7.00	25.00	5214.4	5483.0	0.273	0.310	7	11.00	25.00	5463.9	5625.0	0.509	0.650	5
	5N	7.00	25.00	6451.1	6798.0	0.334	0.360	7	15.50	40.00	6793.2	6975.0	0.918	1.311	4
PS_8	1N	7.50	25.00	1418.1	1552.0	0.076	0.090	7	13.33	33.33	1544.1	1575.0	0.085	0.090	5
	2N	2.50	25.00	2582.2	2799.0	0.142	0.150	9	10.83	33.33	2889.1	2925.0	0.195	0.220	6
	3N	2.50	25.00	3773.4	4136.0	0.197	0.210	9	13.33	33.33	4230.9	4275.0	0.380	0.490	5
	4N	2.50	25.00	4848.6	5368.0	0.260	0.310	9	15.00	25.00	5556.3	5625.0	0.719	0.911	4
	5N	0.00	0.00	5883.2	6406.0	0.308	0.340	10	15.83	33.33	6874.6	6975.0	1.191	1.492	4
PS_9	1N	11.68	22.22	1552.1	1560.0	0.070	0.070	1	9.50	22.22	1573.9	1575.0	0.077	0.090	3
	2N	10.57	18.18	2893.2	2910.0	0.125	0.140	1	6.64	14.29	2925.0	2925.0	0.170	0.190	4
	3N	10.57	18.18	4236.3	4260.0	0.180	0.200	1	7.55	14.29	4274.1	4275.0	0.352	0.410	3
	4N	9.66	14.29	5560.4	5608.0	0.236	0.270	1	8.66	22.22	5621.9	5625.0	0.718	0.961	3
	5N	9.66	14.29	6899.5	6956.0	0.290	0.310	1	6.64	14.29	6973.0	6975.0	1.433	2.002	4
PS_{10}	1N	11.63	18.75	1560.0	1560.0	0.102	0.110	1	7.42	15.00	1575.0	1575.0	0.113	0.120	1
	2N	12.01	18.75	2910.0	2910.0	0.191	0.210	0	6.86	15.79	2925.0	2925.0	0.245	0.260	2
	3N	11.49	18.75	4260.0	4260.0	0.276	0.300	0	6.89	15.00	4275.0	4275.0	0.497	0.550	1
	4N	9.81	15.00	5608.4	5610.0	0.358	0.390	1	5.33	10.53	5625.0	5625.0	1.003	1.111	2
	5N	9.18	15.00	6957.0	6960.0	0.436	0.480	1	5.26	10.53	6975.0	6975.0	1.830	2.073	2
PS_{11}	1N	7.95	20.00	1538.3	1558.0	0.099	0.110	5	7.95	20.00	1573.0	1575.0	0.107	0.120	5
	2N	4.86	20.00	2846.3	2890.0	0.181	0.200	7	8.19	20.00	2919.0	2925.0	0.212	0.230	5
	3N	4.86	20.00	4155.5	4238.0	0.260	0.290	7	6.52	20.00	4264.6	4275.0	0.373	0.410	6
	4N	4.86	20.00	5478.3	5582.0	0.343	0.410	7	8.19	20.00	5625.0	5625.0	0.635	0.701	5
	5N	4.86	20.00	6777.4	6920.0	0.427	0.470	7	8.19	20.00	6967.0	6975.0	1.102	1.592	5
PS_{12}	1N	3.48	9.09	3726.3	3771.0	0.341	0.350	5	5.30	18.18	3800.0	3800.0	0.406	0.420	5
	2N	2.65	9.09	7010.9	7140.0	0.624	0.640	6	3.56	9.09	7200.0	7200.0	1.169	1.321	6
	3N	2.65	9.09	10339.5	10527.0	0.912	0.951	6	3.48	9.09	10594.8	10600.0	2.660	3.154	6
	4N	1.74	9.09	13597.9	13902.0	1.190	1.241	7	4.17	16.67	14000.0	14000.0	6.468	7.280	6
	5N	1.74	9.09	16935.1	17283.0	1.478	1.542	7	3.48	9.09	17400.0	17400.0	12.230	13.419	6

Table C.5 Effect of β on the Beam Search Algorithm using LB_2 as Beam Evaluation Function

		Parallel Beam Search with LB_2							Pooled Beam Search with LB_2						
		% dev.		Total # of Nodes		CPU Time (sec.)		#	% dev.		Total # of Nodes		CPU Time (sec.)		#
	β	Avg.	Max.	Avg.	Max.	Avg.	Max.	opt.	Avg.	Max.	Avg.	Max.	Avg.	Max.	opt.
PS_1	1N	11.41	18.18	425.8	440.0	0.011	0.020	2	11.41	18.18	437.3	450.0	0.011	0.020	2
	2N	11.41	18.18	761.7	788.0	0.017	0.020	2	11.41	18.18	777.5	800.0	0.021	0.030	2
	3N	10.70	18.18	1098.5	1134.0	0.023	0.030	2	11.41	18.18	1124.1	1150.0	0.035	0.040	2
	4N	10.70	18.18	1438.4	1484.0	0.035	0.040	2	11.41	18.18	1467.8	1500.0	0.055	0.060	2
	5N	10.70	18.18	1776.5	1834.0	0.039	0.040	2	11.41	18.18	1820.2	1850.0	0.093	0.110	2
PS_2	1N	15.28	20.83	1516.8	1560.0	0.092	0.100	0	15.70	25.00	1557.8	1575.0	0.102	0.110	0
	2N	15.28	20.83	2825.5	2910.0	0.167	0.190	0	15.70	25.00	2890.2	2925.0	0.234	0.250	0
	3N	15.28	20.83	4137.8	4260.0	0.235	0.260	0	15.70	25.00	4229.7	4275.0	0.537	0.600	0
	4N	15.28	20.83	5449.8	5610.0	0.311	0.340	0	15.70	25.00	5576.1	5625.0	1.147	1.271	0
	5N	15.28	20.83	6756.7	6960.0	0.386	0.430	0	15.70	25.00	6912.6	6975.0	2.205	2.433	0
PS_3	1N	15.38	33.33	1471.6	1556.0	0.084	0.090	0	15.38	33.33	1492.9	1575.0	0.095	0.110	0
	2N	15.38	33.33	2744.0	2901.0	0.158	0.200	0	15.38	33.33	2824.8	2925.0	0.215	0.240	0
	3N	14.88	33.33	4009.4	4208.0	0.219	0.250	0	15.38	33.33	4135.3	4275.0	0.482	0.520	0
	4N	14.88	33.33	5271.5	5549.0	0.287	0.330	0	15.38	33.33	5445.0	5625.0	1.027	1.111	0
	5N	14.88	33.33	6542.9	6893.0	0.356	0.380	0	15.38	33.33	6762.9	6975.0	1.979	2.183	0
PS_4	1N	3.33	16.67	439.9	440.0	0.011	0.020	8	9.76	16.67	449.3	450.0	0.012	0.020	4
	2N	1.67	16.67	787.2	790.0	0.017	0.020	9	5.00	16.67	798.7	800.0	0.023	0.030	7
	3N	1.67	16.67	1134.7	1140.0	0.027	0.030	9	3.33	16.67	1147.7	1150.0	0.036	0.040	8
	4N	0.00	0.00	1469.8	1490.0	0.035	0.040	10	3.33	16.67	1496.1	1500.0	0.057	0.060	8
	5N	0.00	0.00	1812.2	1840.0	0.041	0.050	10	3.33	16.67	1844.7	1850.0	0.096	0.180	8
PS_5	1N	14.30	23.08	1516.0	1560.0	0.094	0.100	0	14.30	23.08	1547.5	1575.0	0.097	0.100	0
	2N	14.30	23.08	2812.9	2910.0	0.168	0.190	0	14.30	23.08	2873.2	2925.0	0.220	0.260	0
	3N	14.30	23.08	4120.1	4260.0	0.245	0.280	0	14.30	23.08	4183.1	4275.0	0.454	0.540	0
	4N	13.30	23.08	5423.7	5610.0	0.315	0.350	0	14.30	23.08	5524.7	5625.0	0.900	1.121	0
	5N	13.30	23.08	6727.4	6960.0	0.392	0.450	0	14.30	23.08	6826.3	6975.0	1.656	2.243	0
PS_6	1N	17.87	25.00	1484.1	1548.0	0.086	0.090	0	19.12	25.00	1540.2	1575.0	0.095	0.100	0
	2N	16.87	25.00	2767.2	2879.0	0.158	0.180	0	19.12	25.00	2861.0	2925.0	0.199	0.220	0
	3N	15.62	25.00	4009.6	4193.0	0.226	0.240	0	19.12	25.00	4156.3	4275.0	0.394	0.470	0
	4N	15.62	25.00	5280.1	5529.0	0.294	0.320	0	19.12	25.00	5497.9	5625.0	0.791	0.981	0
	5N	15.62	25.00	6549.4	6859.0	0.363	0.390	0	18.01	25.00	6835.6	6975.0	1.417	1.842	0
PS_7	1N	11.50	25.00	1406.1	1528.0	0.083	0.090	5	13.50	25.00	1481.4	1575.0	0.088	0.100	4
	2N	13.50	25.00	2603.9	2835.0	0.149	0.190	4	15.50	40.00	2746.0	2925.0	0.169	0.190	4
	3N	11.50	25.00	3772.4	4124.0	0.220	0.240	5	13.50	40.00	4012.0	4275.0	0.291	0.360	5
	4N	11.50	25.00	4965.8	5436.0	0.287	0.320	5	15.50	40.00	5347.1	5625.0	0.506	0.670	4
	5N	11.50	25.00	6160.2	6741.0	0.349	0.370	5	15.50	40.00	6638.4	6975.0	0.838	1.321	4
PS_8	1N	18.33	33.33	1403.7	1540.0	0.082	0.090	3	20.83	33.33	1481.9	1545.0	0.085	0.090	2
	2N	18.33	33.33	2597.1	2853.0	0.150	0.160	3	20.83	33.33	2773.0	2925.0	0.172	0.190	2
	3N	18.33	33.33	3815.6	4188.0	0.220	0.240	3	20.83	33.33	4030.2	4275.0	0.277	0.340	2
	4N	18.33	33.33	5028.5	5504.0	0.283	0.310	3	20.83	33.33	5339.3	5625.0	0.471	0.550	2
	5N	18.33	33.33	6238.5	6827.0	0.353	0.380	3	18.33	33.33	6760.3	6975.0	0.758	1.211	3
PS_9	1N	19.53	33.33	1548.8	1560.0	0.070	0.070	1	22.45	42.86	1569.0	1575.0	0.077	0.080	0
	2N	19.33	28.57	2887.6	2910.0	0.128	0.140	0	21.44	33.33	2921.0	2925.0	0.172	0.190	0
	3N	17.42	28.57	4221.9	4258.0	0.181	0.200	1	20.19	33.33	4271.8	4275.0	0.357	0.400	0
	4N	17.42	28.57	5555.3	5608.0	0.237	0.280	1	21.44	33.33	5616.4	5625.0	0.724	0.821	0
	5N	17.42	28.57	6892.6	6958.0	0.288	0.310	1	20.87	42.86	6967.6	6975.0	1.375	1.632	0
PS_{10}	1N	17.18	25.00	1551.4	1560.0	0.111	0.120	0	17.71	25.00	1575.0	1575.0	0.121	0.130	0
	2N	15.89	21.05	2892.5	2910.0	0.202	0.240	0	17.71	25.00	2925.0	2925.0	0.247	0.290	0
	3N	15.36	21.05	4237.4	4260.0	0.287	0.310	0	16.56	21.05	4275.0	4275.0	0.491	0.580	0
	4N	15.36	21.05	5576.7	5610.0	0.378	0.410	0	16.15	25.00	5625.0	5625.0	0.947	1.041	0
	5N	15.36	21.05	6920.4	6960.0	0.466	0.510	0	16.10	25.00	6975.0	6975.0	1.763	1.902	0
PS_{11}	1N	12.48	28.57	1496.4	1548.0	0.108	0.120	4	15.81	28.57	1539.9	1575.0	0.110	0.120	2
	2N	12.48	28.57	2795.4	2894.0	0.191	0.230	4	15.81	28.57	2851.9	2925.0	0.220	0.260	2
	3N	11.05	28.57	4058.0	4236.0	0.276	0.320	4	15.81	28.57	4159.3	4275.0	0.390	0.480	2
	4N	9.38	28.57	5271.3	5580.0	0.357	0.410	5	15.81	28.57	5493.4	5625.0	0.679	0.871	2
	5N	9.38	28.57	6568.6	6928.0	0.444	0.510	5	15.81	28.57	6888.9	6975.0	1.112	1.522	2
PS_{12}	1N	17.35	25.00	3663.0	3769.0	0.355	0.380	0	19.17	27.27	3767.0	3800.0	0.398	0.420	0
	2N	16.44	25.00	6946.5	7165.0	0.648	0.681	0	19.17	27.27	7129.2	7200.0	0.960	1.031	0
	3N	16.44	25.00	10239.6	10561.0	0.945	0.991	0	18.33	27.27	10467.7	10600.0	2.258	2.503	0
	4N	16.44	25.00	13537.3	13950.0	1.240	1.301	0	19.17	27.27	13899.9	14000.0	4.959	5.588	0
	5N	16.44	25.00	16839.3	17344.0	1.537	1.612	0	18.26	27.27	17235.3	17400.0	9.120	10.855	0

Table C.6 Effect of β on the Beam Search Algorithm using LB_3 as Beam Evaluation Function

		Parallel Beam Search with LB_3							Pooled Beam Search with LB_3						
		% dev.		Total # of Nodes		CPU Time (sec.)		#	% dev.		Total # of Nodes		CPU Time (sec.)		#
	β	Avg.	Max.	Avg.	Max.	Avg.	Max.	opt.	Avg.	Max.	Avg.	Max.	Avg.	Max.	opt.
PS_1	1N	1.54	7.69	437.6	440.0	0.025	0.030	8	4.83	15.38	446.3	450.0	0.025	0.030	5
	2N	0.00	0.00	784.7	790.0	0.034	0.040	10	3.23	9.09	790.8	800.0	0.039	0.050	6
	3N	0.00	0.00	1127.9	1140.0	0.043	0.050	10	2.39	9.09	1131.8	1150.0	0.060	0.070	7
	4N	0.00	0.00	1470.8	1490.0	0.055	0.060	10	2.39	9.09	1477.2	1500.0	0.093	0.150	7
	5N	0.00	0.00	1815.0	1840.0	0.068	0.080	10	2.39	9.09	1821.3	1850.0	0.132	0.140	7
PS_2	1N	7.30	12.50	1538.1	1560.0	0.618	0.821	1	8.13	16.67	1554.4	1575.0	0.619	0.821	1
	2N	7.44	12.50	2870.2	2910.0	0.918	1.201	1	6.37	16.67	2885.6	2925.0	1.013	1.341	1
	3N	6.67	11.54	4199.8	4260.0	1.266	1.662	1	5.70	16.67	4215.2	4275.0	1.607	2.052	1
	4N	6.29	8.33	5528.6	5610.0	1.601	2.113	1	4.95	16.67	5546.0	5625.0	2.551	3.164	1
	5N	6.29	8.33	6858.5	6960.0	1.940	2.543	1	5.35	16.67	6875.3	6975.0	3.988	4.766	1
PS_3	1N	5.95	10.00	1538.5	1560.0	0.327	0.731	2	8.64	19.05	1555.6	1575.0	0.330	0.711	3
	2N	6.75	12.00	2867.0	2910.0	0.490	1.021	2	7.73	19.05	2888.5	2925.0	0.565	1.131	3
	3N	6.75	12.00	4197.4	4260.0	0.679	1.422	2	7.73	19.05	4222.0	4275.0	0.941	1.672	3
	4N	5.95	10.00	5527.0	5610.0	0.871	1.772	2	7.27	19.05	5552.9	5625.0	1.614	2.513	3
	5N	5.95	10.00	6853.3	6960.0	1.058	2.173	2	6.82	19.05	6885.1	6975.0	2.684	3.885	3
PS_4	1N	0.00	0.00	435.9	440.0	0.020	0.020	10	9.35	16.67	446.7	450.0	0.021	0.030	4
	2N	4.35	16.67	780.4	790.0	0.031	0.040	7	11.01	16.67	791.6	800.0	0.036	0.040	3
	3N	1.67	16.67	1113.0	1140.0	0.043	0.050	9	7.68	16.67	1136.7	1150.0	0.047	0.050	5
	4N	1.67	16.67	1456.7	1490.0	0.055	0.060	9	6.01	16.67	1482.4	1500.0	0.071	0.080	6
	5N	0.00	0.00	1793.1	1840.0	0.061	0.070	10	2.92	16.67	1822.3	1850.0	0.104	0.120	8
PS_5	1N	8.26	12.50	1529.8	1560.0	0.531	0.660	1	7.59	12.50	1548.3	1575.0	0.535	0.630	2
	2N	8.20	15.38	2841.9	2907.0	0.842	1.051	2	9.03	15.38	2876.8	2925.0	0.876	1.051	1
	3N	7.43	12.50	4152.5	4247.0	1.149	1.382	2	7.52	12.50	4175.1	4275.0	1.351	1.582	2
	4N	5.59	12.50	5453.3	5593.0	1.454	1.722	4	7.10	16.67	5523.6	5625.0	2.076	2.453	3
	5N	5.59	12.50	6766.6	6936.0	1.765	2.083	4	7.18	10.00	6870.5	6975.0	3.263	3.965	2
PS_6	1N	8.08	12.50	1518.6	1554.0	0.449	0.741	3	12.26	25.00	1555.6	1575.0	0.453	0.721	1
	2N	9.51	14.29	2844.7	2896.0	0.684	1.071	2	12.08	25.00	2887.6	2925.0	0.735	1.081	3
	3N	9.51	14.29	4157.8	4240.0	0.949	1.492	2	9.90	25.00	4219.6	4275.0	1.093	1.602	3
	4N	9.51	14.29	5454.7	5582.0	1.200	1.882	2	9.33	25.00	5551.6	5625.0	1.588	2.243	4
	5N	9.51	14.29	6767.7	6921.0	1.462	2.263	2	10.76	25.00	6883.6	6975.0	2.284	3.074	2
PS_7	1N	9.00	25.00	1478.8	1558.0	0.133	0.300	6	11.00	25.00	1520.3	1575.0	0.132	0.320	5
	2N	9.50	25.00	2744.2	2854.0	0.232	0.510	6	15.50	40.00	2853.6	2925.0	0.240	0.490	4
	3N	9.50	25.00	4016.5	4173.0	0.329	0.701	6	13.50	25.00	4162.3	4275.0	0.375	0.701	4
	4N	9.50	25.00	5300.4	5515.0	0.423	0.871	6	13.50	40.00	5485.0	5625.0	0.578	0.961	5
	5N	9.50	25.00	6543.5	6844.0	0.509	1.051	6	13.50	25.00	6769.0	6975.0	0.913	1.592	4
PS_8	1N	5.00	25.00	1421.2	1554.0	0.075	0.080	8	15.00	25.00	1543.4	1575.0	0.081	0.090	4
	2N	5.83	33.33	2662.5	2787.0	0.135	0.150	8	18.33	33.33	2867.0	2925.0	0.157	0.190	3
	3N	5.83	33.33	3907.1	4121.0	0.202	0.230	8	18.33	33.33	4161.7	4275.0	0.258	0.290	3
	4N	5.83	33.33	5074.0	5466.0	0.254	0.280	8	13.33	33.33	5519.8	5625.0	0.412	0.470	5
	5N	5.83	33.33	6271.0	6632.0	0.316	0.340	8	13.33	33.33	6830.6	6975.0	0.679	0.811	5
PS_9	1N	9.66	14.29	1549.0	1560.0	0.535	1.041	1	10.48	22.22	1572.5	1575.0	0.534	1.061	2
	2N	9.66	14.29	2879.0	2910.0	0.802	1.582	1	12.11	28.57	2907.0	2925.0	0.866	1.642	2
	3N	9.66	14.29	4214.5	4260.0	1.108	2.153	1	8.46	18.18	4267.4	4275.0	1.295	2.233	3
	4N	9.66	14.29	5544.3	5608.0	1.435	2.703	1	11.48	18.18	5625.0	5625.0	2.017	2.994	0
	5N	9.66	14.29	6873.8	6958.0	1.745	3.034	1	12.91	28.57	6962.8	6975.0	2.965	3.735	0
PS_{10}	1N	8.11	12.50	1556.2	1560.0	0.585	0.871	1	8.79	18.75	1575.0	1575.0	0.585	0.851	2
	2N	8.73	12.50	2901.4	2910.0	0.906	1.341	0	5.51	12.50	2925.0	2925.0	0.988	1.442	1
	3N	8.11	12.50	4236.9	4260.0	1.235	1.852	1	4.49	12.50	4275.0	4275.0	1.567	2.183	3
	4N	8.11	12.50	5578.0	5610.0	1.574	2.303	1	4.49	12.50	5625.0	5625.0	2.461	3.214	3
	5N	7.58	12.50	6924.6	6958.0	1.919	2.804	2	4.01	12.50	6974.8	6975.0	3.785	4.786	4
PS_{11}	1N	7.95	20.00	1541.7	1558.0	0.142	0.530	5	14.62	20.00	1572.8	1575.0	0.146	0.530	1
	2N	9.62	20.00	2875.8	2910.0	0.245	0.831	4	12.71	28.57	2925.0	2925.0	0.270	0.871	3
	3N	7.95	20.00	4186.9	4258.0	0.343	1.111	5	9.62	20.00	4267.8	4275.0	0.439	1.181	4
	4N	7.95	20.00	5520.1	5606.0	0.439	1.361	5	11.29	20.00	5620.6	5625.0	0.708	1.692	3
	5N	7.95	20.00	6845.2	6956.0	0.548	1.692	5	8.19	20.00	6975.0	6975.0	1.163	2.433	5
PS_{12}	1N	6.14	9.09	3724.7	3767.0	0.333	0.350	3	11.36	18.18	3799.6	3800.0	0.402	0.420	0
	2N	7.88	9.09	7102.8	7176.0	0.636	0.691	1	9.62	18.18	7183.2	7200.0	1.150	1.211	1
	3N	7.88	9.09	10478.5	10572.0	0.915	0.961	1	9.62	18.18	10596.4	10600.0	3.004	3.194	0
	4N	7.88	9.09	13817.8	13962.0	1.202	1.271	1	10.45	18.18	13999.0	14000.0	6.674	7.460	0
	5N	6.97	9.09	17149.4	17334.0	1.479	1.552	2	9.62	18.18	17400.0	17400.0	12.982	14.270	0

Table C.7 Effect of β on the Beam Search Algorithm using LB_4 as Beam Evaluation Function

		Parallel Beam Search with LB_4							Pooled Beam Search with LB_4						
		% dev.		Total # of Nodes		CPU Time (sec.)		#	% dev.		Total # of Nodes		CPU Time (sec.)		#
	β	Avg.	Max.	Avg.	Max.	Avg.	Max.	opt.	Avg.	Max.	Avg.	Max.	Avg.	Max.	opt.
PS_1	1N	4.05	9.09	438.2	440.0	0.011	0.020	5	5.74	15.38	449.4	450.0	0.011	0.020	4
	2N	0.77	7.69	784.4	790.0	0.020	0.050	9	4.14	9.09	796.6	800.0	0.023	0.030	5
	3N	0.77	7.69	1129.2	1138.0	0.025	0.030	9	2.39	9.09	1142.5	1150.0	0.038	0.040	7
	4N	0.77	7.69	1467.0	1483.0	0.033	0.040	9	2.39	9.09	1489.6	1500.0	0.060	0.060	7
	5N	0.77	7.69	1810.4	1833.0	0.041	0.050	9	2.39	9.09	1836.8	1850.0	0.103	0.120	7
PS_2	1N	6.46	11.11	1541.3	1560.0	0.092	0.100	1	5.99	11.11	1548.1	1575.0	0.100	0.110	0
	2N	6.88	12.50	2876.0	2910.0	0.165	0.180	1	5.99	11.11	2883.1	2925.0	0.238	0.270	0
	3N	6.14	12.50	4207.0	4260.0	0.235	0.260	1	4.84	10.71	4215.8	4275.0	0.549	0.660	2
	4N	5.37	9.52	5538.5	5610.0	0.305	0.340	1	4.56	7.41	5536.6	5625.0	1.211	1.482	1
	5N	4.98	9.52	6866.4	6960.0	0.373	0.420	1	4.08	7.41	6855.3	6975.0	2.351	2.914	2
PS_3	1N	7.28	14.29	1538.9	1560.0	0.080	0.090	2	7.73	19.05	1555.6	1575.0	0.091	0.100	3
	2N	6.35	13.64	2863.2	2910.0	0.151	0.160	2	8.66	23.81	2888.5	2925.0	0.214	0.230	3
	3N	6.35	13.64	4192.3	4260.0	0.214	0.250	2	8.26	23.81	4222.0	4275.0	0.475	0.530	3
	4N	5.95	13.64	5518.8	5610.0	0.279	0.300	2	5.90	15.00	5552.9	5625.0	1.030	1.191	3
	5N	5.95	13.64	6843.4	6960.0	0.341	0.360	2	5.92	15.00	6885.1	6975.0	1.993	2.333	3
PS_4	1N	0.00	0.00	430.9	440.0	0.012	0.020	10	6.25	16.67	443.1	450.0	0.012	0.020	6
	2N	2.68	14.29	771.7	790.0	0.017	0.020	8	7.26	16.67	796.4	800.0	0.020	0.020	5
	3N	1.43	14.29	1114.2	1140.0	0.025	0.030	9	6.01	16.67	1145.1	1150.0	0.046	0.120	6
	4N	1.43	14.29	1437.5	1490.0	0.033	0.040	9	7.68	16.67	1493.1	1500.0	0.054	0.070	5
	5N	0.00	0.00	1776.0	1840.0	0.039	0.040	10	6.01	16.67	1841.8	1850.0	0.086	0.100	6
PS_5	1N	9.26	12.50	1512.8	1558.0	0.087	0.090	0	9.35	20.00	1546.3	1575.0	0.092	0.110	2
	2N	8.26	12.50	2810.4	2906.0	0.159	0.170	1	10.26	20.00	2862.2	2925.0	0.215	0.250	1
	3N	7.35	12.50	4103.5	4256.0	0.227	0.250	2	9.26	20.00	4205.1	4275.0	0.464	0.580	2
	4N	7.35	12.50	5387.1	5604.0	0.298	0.330	2	7.43	12.50	5514.9	5625.0	1.024	1.412	2
	5N	7.35	12.50	6674.3	6927.0	0.366	0.390	2	8.35	12.50	6832.5	6975.0	1.974	2.784	1
PS_6	1N	8.33	25.00	1518.0	1560.0	0.082	0.090	4	10.90	25.00	1555.6	1575.0	0.093	0.120	2
	2N	10.76	25.00	2842.9	2898.0	0.150	0.160	2	8.47	25.00	2887.6	2925.0	0.182	0.210	4
	3N	9.65	25.00	4150.1	4246.0	0.212	0.230	3	11.04	25.00	4219.6	4275.0	0.352	0.400	3
	4N	9.65	25.00	5455.7	5596.0	0.278	0.310	3	6.97	12.50	5551.6	5625.0	0.655	0.801	4
	5N	9.65	25.00	6770.8	6908.0	0.342	0.360	3	9.90	25.00	6883.6	6975.0	1.157	1.512	4
PS_7	1N	9.00	25.00	1478.8	1558.0	0.081	0.090	6	11.00	25.00	1520.3	1575.0	0.083	0.090	5
	2N	9.50	25.00	2744.2	2854.0	0.143	0.160	6	15.50	40.00	2854.0	2925.0	0.159	0.170	4
	3N	9.50	25.00	4016.7	4173.0	0.210	0.250	6	13.50	25.00	4162.3	4275.0	0.265	0.290	4
	4N	9.50	25.00	5300.6	5515.0	0.271	0.290	6	13.50	40.00	5485.0	5625.0	0.431	0.490	5
	5N	9.50	25.00	6543.3	6844.0	0.333	0.360	6	13.50	25.00	6767.2	6975.0	0.732	0.951	4
PS_8	1N	5.00	25.00	1421.2	1554.0	0.079	0.090	8	15.00	25.00	1543.4	1575.0	0.081	0.090	4
	2N	5.83	33.33	2662.5	2787.0	0.141	0.150	8	18.33	33.33	2867.0	2925.0	0.156	0.170	3
	3N	5.83	33.33	3907.1	4121.0	0.202	0.220	8	18.33	33.33	4161.7	4275.0	0.266	0.300	3
	4N	5.83	33.33	5074.0	5466.0	0.262	0.290	8	13.33	33.33	5519.8	5625.0	0.418	0.470	5
	5N	5.83	33.33	6271.0	6632.0	0.321	0.360	8	13.33	33.33	6830.6	6975.0	0.686	0.801	5
PS_9	1N	11.48	18.18	1542.5	1560.0	0.065	0.080	0	10.80	28.57	1567.2	1575.0	0.075	0.080	2
	2N	11.48	18.18	2854.9	2900.0	0.119	0.130	0	14.13	28.57	2923.8	2925.0	0.175	0.210	1
	3N	9.66	14.29	4161.1	4248.0	0.170	0.190	1	9.32	14.29	4275.0	4275.0	0.401	0.590	1
	4N	8.66	14.29	5467.5	5598.0	0.223	0.260	2	10.43	22.22	5623.4	5625.0	0.926	1.482	1
	5N	8.66	14.29	6759.9	6946.0	0.275	0.310	2	14.02	28.57	6973.3	6975.0	1.911	2.964	0
PS_{10}	1N	11.23	18.75	1555.9	1560.0	0.109	0.120	0	7.15	15.79	1575.0	1575.0	0.118	0.130	2
	2N	9.41	15.79	2897.6	2910.0	0.191	0.210	0	4.96	15.79	2922.2	2925.0	0.257	0.280	4
	3N	9.41	15.79	4240.3	4260.0	0.279	0.310	0	3.45	12.50	4275.0	4275.0	0.573	0.660	6
	4N	9.41	15.79	5581.2	5608.0	0.363	0.410	0	3.31	6.25	5625.0	5625.0	1.233	1.452	4
	5N	8.88	12.50	6918.2	6958.0	0.449	0.500	0	4.49	12.50	6975.0	6975.0	2.344	2.914	3
PS_{11}	1N	7.95	20.00	1541.7	1558.0	0.101	0.110	5	14.62	20.00	1573.0	1575.0	0.103	0.110	1
	2N	9.62	20.00	2875.8	2910.0	0.186	0.200	4	12.71	28.57	2925.0	2925.0	0.209	0.220	3
	3N	7.95	20.00	4186.9	4258.0	0.269	0.300	5	9.62	20.00	4269.2	4275.0	0.370	0.420	4
	4N	7.95	20.00	5520.1	5606.0	0.350	0.380	5	11.29	20.00	5620.8	5625.0	0.613	0.721	3
	5N	7.95	20.00	6845.2	6956.0	0.432	0.460	5	8.19	20.00	6975.0	6975.0	1.047	1.301	5
PS_{12}	1N	6.14	9.09	3724.7	3767.0	0.339	0.350	3	11.36	18.18	3799.6	3800.0	0.410	0.420	0
	2N	7.88	9.09	7102.8	7176.0	0.645	0.681	1	9.62	18.18	7183.2	7200.0	1.166	1.241	1
	3N	7.88	9.09	10478.5	10572.0	0.932	0.991	1	9.62	18.18	10596.4	10600.0	3.023	3.234	0
	4N	7.88	9.09	13817.8	13962.0	1.216	1.281	1	10.45	18.18	13999.0	14000.0	6.690	7.480	0
	5N	6.97	9.09	17149.4	17334.0	1.501	1.582	2	9.62	18.18	17400.0	17400.0	13.018	14.310	0

Table C.8 Effect of β on the Beam Search Algorithm using F_1 & F_2 as Beam Evaluation Function

		Parallel Beam Search with F_1&F_2							Pooled Beam Search with F_1&F_2						
		% dev.		Total # of Nodes		CPU Time (sec.)		#	% dev.		Total # of Nodes		CPU Time (sec.)		#
	β	Avg.	Max.	Avg.	Max.	Avg.	Max.	opt.	Avg.	Max.	Avg.	Max.	Avg.	Max.	opt.
PS_1	1N	10.50	18.18	410.7	434.0	0.009	0.010	2	11.41	18.18	429.5	450.0	0.011	0.020	2
	2N	8.68	16.67	717.3	767.0	0.016	0.020	3	11.41	18.18	762.5	800.0	0.019	0.020	2
	3N	8.68	16.67	1031.3	1093.0	0.020	0.020	3	11.41	18.18	1088.6	1150.0	0.037	0.050	2
	4N	8.68	16.67	1333.4	1418.0	0.027	0.030	3	11.41	18.18	1412.5	1500.0	0.069	0.160	2
	5N	7.85	15.38	1635.0	1740.0	0.030	0.030	3	11.41	18.18	1747.5	1850.0	0.108	0.140	2
PS_2	1N	15.70	25.00	1424.8	1524.0	0.079	0.100	0	15.70	25.00	1461.2	1555.0	0.093	0.100	0
	2N	15.70	25.00	2610.8	2842.0	0.142	0.160	0	15.70	25.00	2734.7	2877.0	0.237	0.260	0
	3N	15.70	25.00	3794.8	4145.0	0.202	0.220	0	15.70	25.00	4003.0	4203.0	0.616	0.691	0
	4N	15.28	20.83	4985.3	5352.0	0.265	0.290	0	15.70	25.00	5256.6	5519.0	1.404	1.542	0
	5N	15.28	20.83	6190.9	6608.0	0.323	0.350	0	15.70	25.00	6540.3	6855.0	2.778	3.084	0
PS_3	1N	14.88	33.33	1444.0	1528.0	0.077	0.080	0	15.38	33.33	1476.2	1575.0	0.092	0.120	0
	2N	14.88	33.33	2679.0	2876.0	0.138	0.150	0	15.38	33.33	2739.4	2925.0	0.234	0.250	0
	3N	14.88	33.33	3892.2	4188.0	0.196	0.210	0	15.38	33.33	4008.8	4233.0	0.614	0.680	0
	4N	14.88	33.33	5125.5	5513.0	0.258	0.270	0	15.38	33.33	5277.7	5585.0	1.411	1.562	0
	5N	14.88	33.33	6353.6	6822.0	0.315	0.340	0	15.38	33.33	6566.8	6943.0	2.807	3.144	0
PS_4	1N	4.17	16.67	418.7	436.0	0.009	0.010	7	11.85	16.67	441.9	450.0	0.012	0.020	2
	2N	2.92	16.67	738.3	784.0	0.016	0.020	8	9.17	16.67	782.2	800.0	0.021	0.030	4
	3N	2.92	16.67	1068.5	1131.0	0.023	0.030	8	7.92	16.67	1109.9	1150.0	0.048	0.120	5
	4N	1.67	16.67	1392.4	1479.0	0.030	0.030	9	9.35	16.67	1443.1	1500.0	0.068	0.090	4
	5N	1.67	16.67	1706.3	1825.0	0.036	0.040	9	10.60	16.67	1780.1	1850.0	0.122	0.170	3
PS_5	1N	13.30	23.08	1383.5	1488.0	0.075	0.080	0	14.30	23.08	1424.8	1506.0	0.090	0.100	0
	2N	13.30	23.08	2522.9	2713.0	0.141	0.150	0	14.30	23.08	2673.9	2820.0	0.237	0.260	0
	3N	13.30	23.08	3664.0	4012.0	0.202	0.210	0	14.30	23.08	3905.5	4197.0	0.602	0.660	0
	4N	13.30	23.08	4807.4	5085.0	0.260	0.280	0	14.30	23.08	5166.8	5527.0	1.368	1.532	0
	5N	13.30	23.08	5971.0	6278.0	0.322	0.350	0	14.30	23.08	6393.7	6810.0	2.694	2.984	0
PS_6	1N	17.87	25.00	1417.5	1531.0	0.079	0.090	0	19.12	25.00	1478.3	1575.0	0.090	0.100	0
	2N	17.87	25.00	2610.3	2769.0	0.138	0.150	0	19.12	25.00	2739.1	2925.0	0.240	0.280	0
	3N	17.87	25.00	3795.7	4090.0	0.201	0.230	0	19.12	25.00	4030.7	4271.0	0.616	0.721	0
	4N	17.87	25.00	4989.1	5220.0	0.258	0.280	0	19.12	25.00	5408.4	5625.0	1.419	1.652	0
	5N	17.87	25.00	6196.7	6511.0	0.321	0.360	0	19.12	25.00	6699.5	6975.0	2.807	3.244	0
PS_7	1N	15.50	40.00	1321.7	1494.0	0.076	0.090	4	15.50	40.00	1417.5	1545.0	0.089	0.100	4
	2N	15.50	40.00	2429.7	2780.0	0.136	0.170	4	15.50	40.00	2628.6	2925.0	0.227	0.270	4
	3N	15.50	40.00	3533.6	4025.0	0.192	0.220	4	15.50	40.00	3890.9	4275.0	0.573	0.670	4
	4N	15.50	40.00	4665.1	5295.0	0.247	0.280	4	15.50	40.00	5094.4	5625.0	1.293	1.532	4
	5N	15.50	40.00	5770.3	6582.0	0.308	0.350	4	15.50	40.00	6375.9	6975.0	2.532	3.034	4
PS_8	1N	18.33	33.33	1377.9	1477.0	0.076	0.080	3	20.83	33.33	1442.4	1545.0	0.088	0.100	2
	2N	18.33	33.33	2538.1	2711.0	0.136	0.150	3	18.33	33.33	2708.9	2925.0	0.233	0.270	3
	3N	15.83	33.33	3641.8	3997.0	0.199	0.220	4	18.33	33.33	4029.5	4275.0	0.599	0.731	3
	4N	15.83	33.33	4709.3	5261.0	0.249	0.280	4	18.33	33.33	5278.7	5625.0	1.367	1.682	3
	5N	15.83	33.33	5856.3	6445.0	0.310	0.350	4	18.33	33.33	6548.1	6975.0	2.692	3.314	3
PS_9	1N	17.76	42.86	1478.4	1554.0	0.060	0.070	1	22.59	42.86	1527.1	1575.0	0.080	0.090	0
	2N	18.49	42.86	2724.8	2864.0	0.111	0.130	1	22.70	42.86	2867.1	2925.0	0.229	0.260	0
	3N	15.22	28.57	3925.8	4120.0	0.153	0.170	1	22.70	42.86	4184.0	4275.0	0.651	0.821	0
	4N	15.22	28.57	5149.7	5444.0	0.198	0.220	1	21.27	33.33	5501.9	5625.0	1.564	2.022	0
	5N	15.22	28.57	6350.5	6731.0	0.243	0.270	1	21.27	33.33	6822.6	6975.0	3.146	4.065	0
PS_{10}	1N	18.23	25.00	1393.2	1513.0	0.096	0.110	0	18.23	25.00	1499.0	1575.0	0.112	0.120	0
	2N	18.23	25.00	2576.4	2807.0	0.170	0.190	0	18.23	25.00	2762.8	2925.0	0.264	0.290	0
	3N	18.23	25.00	3761.6	4094.0	0.248	0.270	0	18.23	25.00	4033.4	4275.0	0.646	0.721	0
	4N	18.23	25.00	4958.7	5401.0	0.318	0.360	0	18.23	25.00	5340.6	5625.0	1.452	1.682	0
	5N	18.23	25.00	6149.5	6697.0	0.391	0.430	0	18.23	25.00	6613.6	6975.0	2.805	3.274	0
PS_{11}	1N	16.05	28.57	1437.4	1514.0	0.092	0.110	1	17.48	28.57	1511.5	1575.0	0.107	0.130	1
	2N	16.05	28.57	2674.1	2816.0	0.172	0.200	1	17.48	28.57	2816.4	2925.0	0.258	0.320	1
	3N	16.05	28.57	3902.2	4127.0	0.247	0.280	1	17.48	28.57	4139.1	4275.0	0.616	0.771	1
	4N	16.05	28.57	5137.9	5401.0	0.321	0.370	1	17.48	28.57	5465.0	5625.0	1.357	1.692	1
	5N	16.05	28.57	6376.2	6722.0	0.393	0.460	1	17.48	28.57	6780.4	6975.0	2.639	3.294	1
PS_{12}	1N	18.33	27.27	3567.7	3755.0	0.319	0.330	0	20.83	33.33	3694.2	3800.0	0.427	0.460	0
	2N	19.17	27.27	6731.4	7117.0	0.613	0.681	0	19.17	27.27	7029.5	7200.0	1.664	1.832	0
	3N	19.17	27.27	9929.9	10483.0	0.875	0.941	0	19.17	27.27	10341.1	10600.0	5.113	5.908	0
	4N	18.33	27.27	13040.8	13804.0	1.128	1.181	0	20.00	27.27	13747.4	14000.0	12.095	13.940	0
	5N	18.33	27.27	16273.5	17153.0	1.392	1.432	0	20.00	27.27	17080.8	17400.0	23.601	27.589	0

APPENDIX D

COMPUTATIONAL RESULTS FOR FILTERED BEAM SEARCH EXPERIMENTS

In this appendix, we provide the experimental results for different beam and filter evaluation functions employed. Moreover, the effect of the filter width is tested. The parameter settings for the problems tested in these experiments are given in Table 6.16.

Table D.1 Performance of Parallel vs. Pooled Filtered Beam Search using F_1 & F_2 as Filter Function and LB_s as Beam Function

	Parallel					Pooled				
	% dev. Avg.	% dev. Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal	% dev. Avg.	% dev. Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal
PS_1	8.76	15.38	1831.80	0.047	2	4.63	14.29	1842.50	0.129	5
PS_2	15.70	25.00	6876.90	1.053	0	15.70	25.00	6947.30	3.306	0
PS_3	13.91	28.57	6877.10	0.338	0	15.38	33.33	6961.80	3.075	0
PS_4	6.01	16.67	1830.70	0.058	6	8.93	16.67	1833.10	0.129	4
PS_5	13.30	23.08	6876.80	1.260	0	14.30	23.08	6947.70	3.353	0
PS_6	17.87	25.00	6877.10	0.432	0	19.12	25.00	6960.70	3.093	0
PS_7	15.50	40.00	6877.10	0.332	4	15.50	40.00	6960.30	3.055	4
PS_8	15.00	25.00	6877.10	0.324	4	20.83	33.33	6960.30	3.057	2
PS_9	18.42	33.33	6960.00	1.141	1	22.27	33.33	6965.00	3.568	0
PS_{10}	17.71	25.00	6960.00	1.381	0	17.71	25.00	6974.40	3.329	0
PS_{11}	12.48	28.57	6960.00	0.396	4	17.48	28.57	6975.00	2.914	1
PS_{12}	13.79	25.00	17380.00	1.444	0	20.00	27.27	17400.00	25.600	0

Table D.2 Performance of Parallel vs. Pooled Filtered Beam Search using F_1 & F_2 as Filter Function and UB_1 as Beam Function

	Parallel					Pooled				
	% dev. Avg.	Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal	% dev. Avg.	Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal
PS_1	5.46	15.38	1829.10	0.047	4	8.90	18.18	1835.10	0.130	2
PS_2	14.47	18.75	6831.10	0.419	0	15.70	25.00	6878.60	3.024	0
PS_3	14.43	33.33	6877.10	0.398	0	13.88	33.33	6946.40	3.035	0
PS_4	9.35	16.67	1821.50	0.049	4	6.25	16.67	1820.60	0.130	6
PS_5	10.03	20.00	6855.70	0.439	1	14.30	23.08	6843.10	3.009	0
PS_6	17.87	25.00	6877.10	0.423	0	17.87	25.00	6898.00	3.028	0
PS_7	15.50	40.00	6877.10	0.425	4	15.50	40.00	6903.50	3.042	4
PS_8	15.83	33.33	6877.10	0.417	4	18.33	33.33	6897.40	3.027	3
PS_9	14.54	22.22	6957.30	0.321	0	21.70	42.86	6975.00	3.325	0
PS_{10}	16.18	25.00	6960.00	0.530	0	17.18	25.00	6975.00	2.973	0
PS_{11}	10.81	28.57	6960.00	0.520	5	12.71	28.57	6975.00	2.851	4
PS_{12}	13.94	18.18	17380.00	1.650	0	18.18	27.27	17400.00	26.001	0

Table D.3 Performance of Parallel vs. Pooled Filtered Beam Search using $cost$ as Filter Function and LB_s as Beam Function

	Parallel					Pooled				
	% dev. Avg.	Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal	% dev. Avg.	Max.	Avg. # of Nodes	Avg. CPU Time (sec.)	# optimal
PS_1	10.50	18.18	1786.30	0.065	2	2.44	8.33	1845.40	0.117	7
PS_2	15.70	25.00	6825.60	1.041	0	6.60	11.11	6965.60	2.896	1
PS_3	14.88	33.33	6880.70	0.416	0	4.09	9.09	6967.50	2.722	3
PS_4	13.10	16.67	1810.70	0.070	1	3.33	16.67	1829.00	0.147	8
PS_5	14.30	23.08	6842.50	1.248	0	10.09	16.67	6959.50	4.366	0
PS_6	19.12	25.00	6880.70	0.516	0	12.94	25.00	6967.20	4.189	2
PS_7	15.50	40.00	6880.70	0.423	4	13.50	40.00	6967.50	4.975	5
PS_8	20.83	33.33	6880.70	0.412	2	18.33	33.33	6967.50	5.026	3
PS_9	20.53	33.33	6949.70	1.090	1	21.27	33.33	6970.50	4.243	0
PS_{10}	18.23	25.00	6959.30	1.440	0	12.24	18.75	6975.00	3.667	0
PS_{11}	14.14	28.57	6960.00	0.515	3	14.38	28.57	6975.00	4.405	2
PS_{12}	15.53	25.00	17380.00	1.712	0	6.97	9.09	17400.00	32.420	2

Table D.4 Performance of Parallel vs. Pooled Filtered Beam Search using $cost$ as Filter Function and UB_1 as Beam Function

	Parallel					Pooled				
	% dev.		Avg. # of	Avg. CPU	#	% dev.		Avg. # of	Avg. CPU	#
	Avg.	Max.	Nodes	Time (sec.)	optimal	Avg.	Max.	Nodes	Time (sec.)	optimal
PS_1	7.77	16.67	1818.80	0.056	4	1.54	7.69	1837.10	0.114	8
PS_2	15.36	25.00	6815.60	0.504	0	5.60	10.71	6886.30	2.536	2
PS_3	14.88	33.33	6880.70	0.481	0	3.13	9.09	6905.10	2.637	5
PS_4	14.76	16.67	1800.90	0.058	0	1.67	16.67	1828.90	0.144	9
PS_5	13.47	23.08	6841.80	0.536	0	4.60	10.00	6873.60	3.733	5
PS_6	19.12	25.00	6880.70	0.518	0	4.86	12.50	6883.60	4.034	6
PS_7	15.50	40.00	6880.70	0.522	4	9.00	40.00	6884.00	4.900	7
PS_8	20.83	33.33	6880.70	0.508	2	8.33	33.33	6884.00	4.942	7
PS_9	23.70	42.86	6951.60	0.376	0	7.69	25.00	6973.40	3.939	4
PS_{10}	17.71	25.00	6960.00	0.649	0	3.43	12.50	6974.90	3.417	5
PS_{11}	15.81	28.57	6960.00	0.654	2	4.86	20.00	6975.00	4.483	7
PS_{12}	16.44	27.27	17380.00	1.930	1	1.74	9.09	17400.00	32.338	8

Table D.5 Performance of Parallel vs. Pooled Filtered Beam Search using UB_1 as Filter Function and LB_s as Beam Function

	Parallel					Pooled				
	% dev.		Avg. # of	Avg. CPU	#	% dev.		Avg. # of	Avg. CPU	#
	Avg.	Max.	Nodes	Time (sec.)	optimal	Avg.	Max.	Nodes	Time (sec.)	optimal
PS_1	9.59	16.67	1771.40	0.069	2	5.73	15.38	1846.60	0.136	4
PS_2	15.28	20.83	6795.70	1.129	0	10.32	16.67	6958.80	3.182	1
PS_3	14.91	28.57	6879.50	0.493	0	7.68	14.29	6961.80	3.058	2
PS_4	13.10	16.67	1805.50	0.081	1	7.68	16.67	1833.80	0.165	5
PS_5	14.30	23.08	6834.10	1.366	0	10.76	16.67	6955.60	3.928	1
PS_6	19.12	25.00	6881.90	0.611	0	15.62	25.00	6967.50	3.901	0
PS_7	15.50	40.00	6885.50	0.517	4	11.50	25.00	6967.50	4.622	5
PS_8	15.83	33.33	6884.30	0.501	4	18.33	33.33	6967.50	4.867	3
PS_9	23.70	42.86	6937.10	1.171	0	18.22	28.57	6975.00	3.738	0
PS_{10}	18.23	25.00	6959.30	1.609	0	16.04	25.00	6975.00	3.531	0
PS_{11}	12.48	28.57	6960.00	0.638	4	12.71	28.57	6975.00	4.648	3
PS_{12}	16.44	27.27	17380.00	2.004	0	12.12	18.18	17400.00	34.842	2

APPENDIX E

PERFORMANCE COMPARISON OF BEAM SEARCH VS. TRUNCATED B&B

The performance comparison of the best Beam Search algorithms to the truncated Branch-and-Bound algorithm with a time limit of 2 hours for full parameter settings presented in Table 6.2 are given in this appendix. We use 4 different number of job, N values for each of the combination presented in Table 6.2 and we generate 10 problem instances for each setting. The results for larger N values (i.e. $N = 20, 25$) are given in main text, so we only provide the results for $N = 10, 15$ in this appendix.

Table E.1 Performance Comparison of Parallel Beam Search vs. Truncated B&B for N=10

					Parallel							
Setting					LB_s				UB_1			
N	*T*	*(min, max)*	*C*	*D*	% dev.	# best	CPU (sec.)	Avg. Nodes	% dev.	# best	CPU (sec.)	Avg. Nodes
10	10	(2,5)	5	2	5.22	6	0.067	1812.2	1.43	9	0.052	1753.1
10	10	(2,5)	5	4	2.50	9	0.066	1767.9	0.00	10	0.056	1721.3
10	15	(2,5)	5	2	5.97	5	0.070	1790.3	1.25	9	0.063	1717.6
10	15	(2,5)	5	4	3.33	8	0.074	1773.3	0.00	10	0.068	1685.3
10	15	(2,10)	10	2	2.25	8	0.077	1833.6	1.25	9	0.066	1655.1
10	15	(2,10)	10	5	0.00	10	0.095	1828.4	0.00	10	0.070	1649.6
10	15	(2,10)	10	8	2.50	9	0.095	1820.2	0.00	10	0.072	1631.6
10	15	(5,10)	10	2	1.82	8	0.085	1813.2	2.65	7	0.078	1764.9
10	15	(5,10)	10	5	0.00	10	0.086	1809.9	0.00	10	0.075	1762.1
10	15	(5,10)	10	8	0.00	10	0.083	1788.3	0.00	10	0.078	1773.1
10	20	(2,5)	5	2	4.06	6	0.065	1790.8	2.11	8	0.067	1737.0
10	20	(2,5)	5	4	1.43	9	0.070	1737.2	0.00	10	0.069	1675.5
10	20	(2,10)	10	2	5.41	5	0.090	1838.8	2.59	7	0.091	1769.5
10	20	(2,10)	10	5	3.10	8	0.097	1829.2	1.43	9	0.090	1745.0
10	20	(2,10)	10	8	6.00	7	0.100	1818.4	0.00	10	0.091	1752.3
10	20	(5,10)	10	2	3.20	6	0.099	1836.2	3.44	5	0.098	1817.4
10	20	(5,10)	10	5	0.00	10	0.094	1830.2	0.00	10	0.105	1811.0
10	20	(5,10)	10	8	0.00	10	0.093	1826.5	0.00	10	0.108	1805.2
10	20	(2,15)	15	2	3.21	7	0.087	1816.2	2.39	7	0.082	1706.6
10	20	(2,15)	15	5	0.00	10	0.091	1775.5	1.67	9	0.079	1623.1
10	20	(2,15)	15	8	0.00	10	0.093	1749.9	0.00	10	0.081	1598.6
10	20	(5,15)	15	2	6.29	3	0.103	1838.0	5.52	4	0.084	1729.2
10	20	(5,15)	15	5	0.00	10	0.105	1827.5	0.00	10	0.091	1744.2
10	20	(5,15)	15	8	6.17	7	0.105	1814.1	6.17	7	0.093	1723.1
10	20	(2,5)	15	2	0.00	10	0.059	1794.9	0.00	10	0.066	1792.8
10	20	(2,5)	15	4	0.00	10	0.052	1665.0	0.00	10	0.066	1670.0
10	20	(2,10)	15	2	0.00	10	0.057	1788.6	0.00	10	0.072	1780.1
10	20	(2,10)	15	5	2.50	9	0.052	1639.2	0.00	10	0.069	1582.5
10	20	(2,10)	15	8	3.33	9	0.051	1594.7	3.33	9	0.070	1587.9
10	25	(2,5)	5	2	2.68	7	0.068	1791.0	2.68	7	0.080	1742.4
10	25	(2,5)	5	4	3.33	8	0.072	1735.5	0.00	10	0.079	1683.6
10	25	(2,10)	10	2	1.43	8	0.092	1839.6	0.71	9	0.099	1792.3
10	25	(2,10)	10	5	3.10	8	0.093	1825.1	3.10	8	0.096	1731.4
10	25	(2,10)	10	8	2.00	9	0.103	1821.0	0.00	10	0.102	1733.7
10	25	(5,10)	10	2	3.14	5	0.112	1839.8	3.65	4	0.113	1825.8
10	25	(5,10)	10	5	0.00	10	0.101	1808.4	1.25	9	0.119	1797.7
10	25	(5,10)	10	8	0.00	10	0.105	1824.8	0.00	10	0.123	1815.9
10	25	(2,15)	15	2	4.07	4	0.099	1838.8	4.69	4	0.102	1763.5
10	25	(2,15)	15	5	5.68	6	0.107	1833.2	0.00	10	0.107	1689.8
10	25	(2,15)	15	8	9.00	6	0.106	1826.8	2.00	9	0.111	1710.6
10	25	(5,15)	15	2	1.21	8	0.113	1837.8	2.85	5	0.116	1802.8
10	25	(5,15)	15	5	0.00	10	0.114	1822.1	0.00	10	0.117	1776.5
10	25	(5,15)	15	8	2.92	8	0.110	1818.3	1.67	9	0.128	1782.3
10	25	(2,5)	15	2	0.00	10	0.061	1779.1	0.00	10	0.072	1779.1
10	25	(2,5)	15	4	0.00	10	0.057	1713.1	0.00	10	0.072	1707.9
10	25	(2,10)	15	2	0.83	9	0.068	1823.6	0.00	10	0.089	1803.6
10	25	(2,10)	15	5	0.00	10	0.069	1800.3	0.00	10	0.096	1739.3
10	25	(2,10)	15	8	0.00	10	0.067	1732.6	0.00	10	0.086	1626.2
10	15	(2,5)	10	2	0.00	10	0.046	1789.5	0.00	10	0.063	1782.5
10	15	(2,5)	10	4	0.00	10	0.043	1617.9	0.00	10	0.052	1622.0
10	20	(2,5)	10	2	0.00	10	0.054	1818.7	0.00	10	0.069	1819.6
10	20	(2,5)	10	4	0.00	10	0.051	1777.7	0.00	10	0.069	1748.0
10	25	(2,5)	10	2	0.00	10	0.058	1800.7	0.00	10	0.071	1798.9
10	25	(2,5)	10	4	0.00	10	0.054	1723.6	0.00	10	0.071	1720.7
10	25	(2,5)	20	2	0.00	10	0.056	1766.0	0.00	10	0.073	1764.2
10	25	(2,5)	20	4	0.00	10	0.054	1667.2	0.00	10	0.068	1667.5
10	25	(5,10)	20	2	0.00	10	0.070	1802.0	0.00	10	0.088	1791.4
10	25	(5,10)	20	5	0.00	10	0.056	1465.2	0.00	10	0.075	1523.5
10	25	(5,10)	20	8	0.00	10	0.063	1529.5	0.00	10	0.076	1457.1
10	25	(2,10)	20	2	0.00	10	0.065	1777.4	0.00	10	0.083	1776.9
10	25	(2,10)	20	5	0.00	10	0.060	1634.0	0.00	10	0.079	1651.7
10	25	(2,10)	20	8	0.00	10	0.058	1445.4	0.00	10	0.072	1452.2

Table E.2 Performance Comparison of Pooled Beam Search vs. Truncated B&B for N=10

Setting					Pooled								Best of all	
					LB_s				UB_1					
N	*T*	*(min, max)*	*C*	*D*	% dev.	# best	CPU (sec.)	Avg. Nodes	% dev.	# best	CPU (sec.)	Avg. Nodes	% dev.	# best
10	10	(2,5)	5	2	5.04	6	0.112	1849.0	1.43	9	0.109	1842.3	1.43	9
10	10	(2,5)	5	4	9.50	5	0.101	1847.2	0.00	10	0.099	1840.0	0.00	10
10	15	(2,5)	5	2	2.50	8	0.125	1830.2	1.25	9	0.110	1791.4	1.25	9
10	15	(2,5)	5	4	3.67	8	0.117	1822.2	0.00	10	0.102	1769.3	0.00	10
10	15	(2,10)	10	2	0.00	10	0.117	1849.8	0.00	10	0.131	1850.0	0.00	10
10	15	(2,10)	10	5	0.00	10	0.125	1850.0	0.00	10	0.114	1840.0	0.00	10
10	15	(2,10)	10	8	9.50	6	0.118	1849.6	0.00	10	0.118	1842.4	0.00	10
10	15	(5,10)	10	2	2.73	7	0.127	1850.0	1.82	8	0.126	1850.0	1.82	8
10	15	(5,10)	10	5	0.00	10	0.114	1840.0	0.00	10	0.113	1840.8	0.00	10
10	15	(5,10)	10	8	0.00	10	0.120	1840.0	0.00	10	0.111	1842.0	0.00	10
10	20	(2,5)	5	2	0.00	10	0.117	1831.3	1.00	9	0.116	1803.5	0.00	9
10	20	(2,5)	5	4	5.33	7	0.105	1779.2	0.00	10	0.107	1770.9	0.00	10
10	20	(2,10)	10	2	2.59	7	0.135	1849.7	0.00	10	0.145	1848.7	0.00	10
10	20	(2,10)	10	5	1.43	9	0.129	1848.6	1.43	9	0.125	1842.6	1.43	9
10	20	(2,10)	10	8	5.67	7	0.136	1847.1	2.00	9	0.132	1842.9	0.00	9
10	20	(5,10)	10	2	1.34	8	0.165	1850.0	1.19	8	0.159	1850.0	0.00	8
10	20	(5,10)	10	5	0.00	10	0.122	1849.8	0.00	10	0.130	1848.2	0.00	10
10	20	(5,10)	10	8	2.00	9	0.138	1850.0	0.00	10	0.136	1850.0	0.00	10
10	20	(2,15)	15	2	2.25	7	0.134	1846.6	0.77	9	0.143	1850.0	0.77	9
10	20	(2,15)	15	5	0.00	10	0.117	1840.0	0.00	10	0.115	1820.2	0.00	10
10	20	(2,15)	15	8	5.00	8	0.113	1808.6	0.00	10	0.120	1808.1	0.00	10
10	20	(5,15)	15	2	3.99	5	0.143	1850.0	3.22	6	0.148	1850.0	3.22	6
10	20	(5,15)	15	5	1.25	9	0.143	1850.0	0.00	10	0.130	1842.2	0.00	10
10	20	(5,15)	15	8	9.83	5	0.143	1850.0	6.17	7	0.132	1845.4	4.50	7
10	20	(2,5)	15	2	0.00	10	0.092	1828.6	0.00	10	0.090	1834.1	0.00	10
10	20	(2,5)	15	4	0.00	10	0.068	1721.6	0.00	10	0.077	1735.7	0.00	10
10	20	(2,10)	15	2	0.00	10	0.100	1834.8	0.00	10	0.106	1832.0	0.00	10
10	20	(2,10)	15	5	4.50	8	0.069	1791.6	0.00	10	0.091	1796.4	0.00	10
10	20	(2,10)	15	8	3.33	9	0.066	1739.9	0.00	10	0.093	1756.6	0.00	10
10	25	(2,5)	5	2	2.68	7	0.130	1835.3	1.77	8	0.139	1816.0	1.00	8
10	25	(2,5)	5	4	3.10	8	0.127	1822.8	0.00	10	0.120	1780.7	0.00	10
10	25	(2,10)	10	2	1.43	8	0.141	1849.8	0.71	9	0.155	1847.4	0.71	9
10	25	(2,10)	10	5	3.10	8	0.133	1848.2	3.10	8	0.144	1843.2	3.10	8
10	25	(2,10)	10	8	4.00	8	0.133	1844.0	0.00	10	0.143	1825.7	0.00	10
10	25	(5,10)	10	2	1.18	8	0.178	1850.0	0.63	9	0.171	1850.0	0.00	9
10	25	(5,10)	10	5	0.00	10	0.137	1847.3	1.25	9	0.155	1850.0	0.00	9
10	25	(5,10)	10	8	1.43	9	0.163	1850.0	0.00	10	0.157	1850.0	0.00	10
10	25	(2,15)	15	2	4.02	5	0.143	1850.0	4.07	4	0.170	1850.0	3.35	4
10	25	(2,15)	15	5	0.00	10	0.138	1850.0	1.25	9	0.157	1849.0	0.00	9
10	25	(2,15)	15	8	13.92	3	0.146	1850.0	2.00	9	0.150	1841.4	2.00	9
10	25	(5,15)	15	2	1.18	8	0.166	1850.0	0.00	10	0.178	1850.0	0.00	10
10	25	(5,15)	15	5	0.00	10	0.146	1850.0	0.00	10	0.164	1835.2	0.00	10
10	25	(5,15)	15	8	8.27	5	0.146	1850.0	1.67	9	0.158	1850.0	1.67	9
10	25	(2,5)	15	2	0.00	10	0.093	1815.0	0.00	10	0.099	1815.0	0.00	10
10	25	(2,5)	15	4	0.00	10	0.080	1753.0	0.00	10	0.092	1791.3	0.00	10
10	25	(2,10)	15	2	0.00	10	0.121	1850.0	0.00	10	0.135	1846.8	0.00	10
10	25	(2,10)	15	5	0.00	10	0.098	1838.0	0.00	10	0.117	1836.8	0.00	10
10	25	(2,10)	15	8	2.50	9	0.094	1850.0	0.00	10	0.122	1830.4	0.00	10
10	15	(2,5)	10	2	0.00	10	0.076	1840.0	0.00	10	0.083	1841.4	0.00	10
10	15	(2,5)	10	4	0.00	10	0.062	1742.9	0.00	10	0.069	1756.4	0.00	10
10	20	(2,5)	10	2	0.00	10	0.087	1843.4	0.00	10	0.095	1842.6	0.00	10
10	20	(2,5)	10	4	0.00	10	0.072	1818.6	0.00	10	0.083	1808.8	0.00	10
10	25	(2,5)	10	2	0.00	10	0.093	1837.1	0.00	10	0.102	1837.1	0.00	10
10	25	(2,5)	10	4	0.00	10	0.075	1782.3	0.00	10	0.090	1791.6	0.00	10
10	25	(2,5)	20	2	0.00	10	0.095	1779.8	0.00	10	0.098	1779.4	0.00	10
10	25	(2,5)	20	4	0.00	10	0.073	1716.0	0.00	10	0.080	1734.4	0.00	10
10	25	(5,10)	20	2	0.00	10	0.114	1849.2	0.00	10	0.120	1845.4	0.00	10
10	25	(5,10)	20	5	0.00	10	0.080	1745.6	0.00	10	0.104	1745.0	0.00	10
10	25	(5,10)	20	8	0.00	10	0.101	1850.0	0.00	10	0.111	1822.1	0.00	10
10	25	(2,10)	20	2	0.00	10	0.107	1843.2	0.00	10	0.110	1841.2	0.00	10
10	25	(2,10)	20	5	0.00	10	0.083	1691.7	0.00	10	0.088	1732.4	0.00	10
10	25	(2,10)	20	8	0.00	10	0.070	1679.6	0.00	10	0.090	1705.2	0.00	10

Table E.3 Performance Comparison of Parallel Beam Search vs. Truncated B&B for N=15

Setting					Parallel							
					LB_s				UB_1			
N	***T***	***(min, max)***	***C***	***D***	**% dev.**	**# best**	**CPU (sec.)**	**Avg. Nodes**	**% dev.**	**# best**	**CPU (sec.)**	**Avg. Nodes**
15	10	(2,5)	5	2	9.66	1	1.534	6842.9	5.53	5	0.351	6769.3
15	10	(2,5)	5	4	0.00	9	1.409	6836.8	0.00	9	0.362	6754.0
15	15	(2,5)	5	2	14.49	0	1.842	6917.3	9.84	0	0.438	6825.8
15	15	(2,10)	10	2	9.97	0	2.570	6922.2	5.85	4	0.454	6470.4
15	15	(5,10)	10	2	6.33	4	2.116	6888.8	3.73	5	0.488	6638.9
15	20	(2,5)	5	2	6.14	3	1.390	6912.8	2.97	6	0.471	6785.8
15	20	(2,5)	5	4	2.50	8	1.509	6782.8	0.00	10	0.483	6670.7
15	20	(2,10)	10	2	7.41	1	2.247	6951.5	6.24	2	0.560	6704.9
15	20	(5,10)	10	2	7.30	0	2.015	6934.9	5.43	3	0.622	6835.8
15	20	(2,15)	15	2	8.71	2	1.858	6932.6	3.99	5	0.534	6393.2
15	20	(5,15)	15	2	7.36	1	2.060	6923.4	5.41	2	0.563	6556.1
15	20	(2,5)	15	2	0.00	10	0.349	6825.4	0.00	10	0.422	6830.7
15	20	(2,5)	15	4	0.00	10	0.323	6311.7	0.00	10	0.400	6350.7
15	20	(2,10)	15	2	3.82	6	0.382	6762.9	0.00	10	0.464	6393.5
15	20	(2,10)	15	5	0.00	10	0.399	6682.5	0.00	10	0.467	6054.0
15	20	(2,10)	15	8	3.33	9	0.406	6608.1	0.00	10	0.457	5924.9
15	25	(2,5)	5	2	8.12	0	1.500	6907.3	5.94	2	0.541	6779.3
15	25	(2,5)	5	4	2.02	8	1.360	6775.2	1.11	9	0.546	6634.7
15	25	(2,10)	10	2	9.52	0	1.829	6955.6	4.59	2	0.604	6809.1
15	25	(2,10)	10	5	14.06	0	1.967	6943.4	7.81	4	0.631	6759.2
15	25	(2,10)	10	8	3.10	8	1.987	6917.9	3.10	8	0.702	6806.7
15	25	(5,10)	10	2	5.44	2	2.014	6936.0	4.53	3	0.727	6907.4
15	25	(5,10)	10	5	0.83	9	1.452	6868.6	0.91	9	0.775	6893.0
15	25	(5,10)	10	8	-0.91	10	1.240	6924.4	-0.91	10	0.785	6895.3
15	25	(2,15)	15	2	8.96	0	2.163	6953.1	7.00	1	0.665	6693.1
15	25	(2,15)	15	5	5.69	5	2.430	6906.7	4.44	6	0.692	6563.9
15	25	(2,5)	15	2	0.00	10	0.390	6899.2	0.00	10	0.473	6890.0
15	25	(2,5)	15	4	0.00	10	0.371	6612.8	0.00	10	0.474	6587.6
15	25	(2,10)	15	2	5.80	3	0.442	6917.9	3.65	5	0.580	6756.7
15	25	(2,10)	15	5	4.86	7	0.502	6779.4	4.86	7	0.585	6483.7
15	25	(2,10)	15	8	11.50	5	0.554	6769.2	0.00	10	0.571	6099.7
15	15	(2,5)	10	2	0.00	10	0.305	6667.0	0.00	10	0.383	6634.6
15	15	(2,5)	10	4	5.00	8	0.287	6062.9	7.50	7	0.349	5917.2
15	20	(2,5)	10	2	1.00	9	0.341	6745.0	0.00	10	0.447	6691.6
15	20	(2,5)	10	4	10.00	5	0.340	6588.1	8.00	6	0.439	6428.1
15	25	(2,5)	10	2	0.91	9	0.362	6853.1	0.00	10	0.482	6816.3
15	25	(2,5)	10	4	5.00	7	0.354	6605.7	0.00	10	0.466	6325.1
15	25	(2,5)	20	2	0.00	10	0.390	6810.6	0.00	10	0.469	6815.7
15	25	(2,5)	20	4	0.00	10	0.386	6605.7	0.00	10	0.464	6615.1
15	25	(5,10)	20	2	0.00	10	0.450	6716.9	0.00	10	0.584	6632.2
15	25	(5,10)	20	5	0.00	10	0.459	6426.1	2.00	9	0.572	6173.8
15	25	(5,10)	20	8	5.00	8	0.433	5913.9	0.00	10	0.541	5727.3
15	25	(2,10)	20	2	0.00	10	0.417	6702.4	0.00	10	0.539	6748.5
15	25	(2,10)	20	5	0.00	10	0.374	5728.0	0.00	10	0.465	5642.5
15	25	(2,10)	20	8	3.33	9	0.423	6524.5	3.33	9	0.536	6292.5

Table E.4 Performance Comparison of Pooled Beam Search vs. Truncated B&B for N=15

Setting					Pooled								Best of all	
					LB_s				UB_1					
N	*T*	*(min, max)*	*C*	*D*	% dev.	# best	CPU (sec.)	Avg. Nodes	% dev.	# best	CPU (sec.)	Avg. Nodes	% dev.	# best
15	10	(2,5)	5	2	7.43	4	2.772	6973.0	1.94	8	1.971	6975.0	0.00	8
15	10	(2,5)	5	4	7.35	4	2.520	6973.9	0.00	9	1.759	6972.8	-1.25	9
15	15	(2,5)	5	2	7.19	2	3.217	6969.9	7.19	2	2.270	6948.7	7.19	2
15	15	(2,10)	10	2	6.11	5	3.799	6974.9	3.39	6	2.361	6975.0	0.71	6
15	15	(5,10)	10	2	3.83	5	3.475	6975.0	1.46	8	2.365	6975.0	1.46	8
15	20	(2,5)	5	2	1.48	8	3.306	6961.1	2.20	7	2.296	6928.0	0.71	7
15	20	(2,5)	5	4	5.86	5	2.887	6827.9	0.00	10	1.868	6794.2	0.00	10
15	20	(2,10)	10	2	5.82	4	3.528	6974.9	1.48	8	2.524	6974.8	1.48	8
15	20	(5,10)	10	2	1.68	7	3.755	6975.0	1.07	7	2.606	6975.0	0.07	7
15	20	(2,15)	15	2	5.20	4	3.018	6975.0	4.17	5	2.421	6960.0	2.10	5
15	20	(5,15)	15	2	5.52	3	3.372	6975.0	4.98	4	2.532	6975.0	3.55	4
15	20	(2,5)	15	2	0.00	10	1.467	6855.3	0.00	10	1.214	6868.8	0.00	10
15	20	(2,5)	15	4	0.00	10	1.160	6458.0	0.00	10	0.913	6568.1	0.00	10
15	20	(2,10)	15	2	1.00	9	1.571	6974.8	0.91	9	1.972	6960.0	0.00	9
15	20	(2,10)	15	5	10.00	5	0.893	6930.4	0.00	10	1.477	6830.5	0.00	10
15	20	(2,10)	15	8	11.67	6	0.759	6967.0	3.33	9	1.369	6847.2	0.00	9
15	25	(2,5)	5	2	4.70	5	3.573	6961.1	4.06	4	2.367	6866.5	2.68	4
15	25	(2,5)	5	4	3.02	7	2.779	6833.4	1.11	9	1.626	6621.3	1.11	9
15	25	(2,10)	10	2	3.96	4	3.518	6975.0	1.36	8	2.650	6974.9	0.59	8
15	25	(2,10)	10	5	8.50	3	2.827	6974.4	5.69	6	2.082	6972.1	2.92	6
15	25	(2,10)	10	8	5.12	6	2.999	6974.6	1.85	8	2.007	6975.0	1.85	8
15	25	(5,10)	10	2	1.37	7	4.136	6975.0	1.73	6	2.679	6975.0	0.92	6
15	25	(5,10)	10	5	2.65	7	2.534	6975.0	0.91	9	2.051	6975.0	0.00	9
15	25	(5,10)	10	8	0.00	9	2.547	6974.8	-0.91	10	1.955	6974.8	-0.91	10
15	25	(2,15)	15	2	4.89	2	3.511	6974.5	5.01	1	2.635	6974.3	3.04	1
15	25	(2,15)	15	5	6.94	4	3.475	6975.0	2.36	8	2.290	6975.0	2.36	8
15	25	(2,5)	15	2	0.00	10	1.673	6941.0	0.00	10	1.419	6942.4	0.00	10
15	25	(2,5)	15	4	0.00	10	1.250	6677.3	0.00	10	1.163	6768.9	0.00	10
15	25	(2,10)	15	2	1.38	8	1.984	6975.0	3.65	5	2.268	6960.2	1.38	5
15	25	(2,10)	15	5	8.19	5	1.163	6967.0	4.86	7	1.725	6929.1	4.86	7
15	25	(2,10)	15	8	14.83	3	1.503	6958.6	0.00	10	1.634	6960.0	0.00	10
15	15	(2,5)	10	2	0.00	10	1.383	6922.5	0.00	10	1.620	6898.2	0.00	10
15	15	(2,5)	10	4	15.00	4	0.896	6952.5	7.50	7	1.291	6836.3	5.00	7
15	20	(2,5)	10	2	0.00	10	1.679	6932.5	1.00	9	1.857	6883.5	0.00	9
15	20	(2,5)	10	4	6.00	7	1.227	6802.8	6.00	7	1.359	6791.0	6.00	7
15	25	(2,5)	10	2	0.00	10	1.791	6938.3	0.00	10	1.887	6942.7	0.00	10
15	25	(2,5)	10	4	5.00	7	1.299	6904.3	0.00	10	1.547	6876.3	0.00	10
15	25	(2,5)	20	2	0.00	10	1.574	6878.4	0.00	10	1.369	6878.4	0.00	10
15	25	(2,5)	20	4	0.00	10	1.285	6762.5	0.00	10	0.953	6741.8	0.00	10
15	25	(5,10)	20	2	0.00	10	1.880	6915.1	0.00	10	1.937	6856.2	0.00	10
15	25	(5,10)	20	5	4.00	8	0.917	6801.7	2.00	9	1.352	6725.3	0.00	9
15	25	(5,10)	20	8	7.50	7	1.116	6551.1	0.00	10	1.408	6722.2	0.00	10
15	25	(2,10)	20	2	0.00	10	1.804	6835.5	0.00	10	1.630	6867.2	0.00	10
15	25	(2,10)	20	5	8.00	6	1.003	6872.8	0.00	10	1.445	6825.3	0.00	10
15	25	(2,10)	20	8	0.00	10	1.130	6743.6	0.00	10	0.898	6573.0	0.00	10

APPENDIX F

PERFORMANCE OF BEAM SEARCH ON LARGE-SIZED INSTANCES

In this appendix, we provide the performances of Beam Search algorithms using different search strategies and evaluation functions on the instances taken from Hertz. et al. (1998). The results are further compared with the best algorithms presented in Hertz. et al. (1998). The discussions are given in main text.

Table F.1 Performances of Beam Searches using UB_1 as Beam Evaluation Function

Setting					Parallel				Pooled			
N	T	(min,max)	C	D	% dev.	# best	CPU Time (sec.)	Avg. Nodes	% dev.	# best	CPU Time (sec.)	Avg. Nodes
10	10	(2, 4)	4	1	1.60	8	0.052	1811.4	0.00	10	0.110	1844.0
10	10	(2, 4)	5	1	0.00	10	0.046	1793.1	0.00	10	0.101	1850.0
10	10	(2, 4)	6	1	0.00	10	0.046	1791.1	0.00	10	0.096	1850.0
10	10	(2, 4)	7	1	0.00	10	0.043	1806.7	0.00	10	0.076	1827.4
15	20	(2, 6)	6	1	1.51	6	0.500	6848.8	0.74	7	2.522	6909.1
15	20	(2, 6)	8	1	1.75	6	0.466	6805.7	0.00	9	2.481	6919.4
15	20	(2, 6)	10	1	2.08	6	0.453	6770.8	1.58	7	2.362	6920.7
15	20	(2, 6)	12	1	0.00	10	0.443	6796.5	0.00	10	2.084	6937.0
30	40	(5, 15)	15	1	5.04	0	23.009	61109.1	2.34	0	500.159	61650.0
30	40	(5, 15)	17	1	5.57	0	22.272	61058.5	2.68	1	491.235	61650.0
30	40	(5, 15)	20	1	7.03	0	21.469	60797.3	4.18	1	493.789	61650.0
30	40	(5, 15)	25	1	4.74	1	20.300	60168.0	3.65	3	488.128	61650.0
40	60	(7, 20)	20	1	4.49	1	123.444	149088.0	2.17	2	4776.452	149600.0
40	60	(7, 20)	22	1	5.06	0	120.433	148868.8	3.23	1	4712.461	149600.0
40	60	(7, 20)	25	1	5.21	0	117.090	148635.2	3.11	0	4656.559	149600.0
40	60	(7, 20)	30	1	6.22	0	112.519	148232.9	3.79	0	4514.957	149600.0

Table F.2 Performances of Beam Searches using all LB_s as Beam Evaluation Function

Setting					Parallel				Pooled			
N	T	(min,max)	C	D	% dev.	# best	CPU Time (sec.)	Avg. Nodes	% dev.	# best	CPU Time (sec.)	Avg. Nodes
10	10	(2, 4)	4	1	0.77	9	0.060	1820.4	0.00	10	0.118	1849.5
10	10	(2, 4)	5	1	0.91	9	0.045	1815.7	0.91	9	0.089	1850.0
10	10	(2, 4)	6	1	0.00	10	0.038	1794.5	0.00	10	0.076	1850.0
10	10	(2, 4)	7	1	0.00	10	0.037	1789.1	0.00	10	0.060	1809.3
15	20	(2, 6)	6	1	6.25	2	1.582	6870.8	1.60	6	3.659	6962.9
15	20	(2, 6)	8	1	3.17	4	0.484	6852.1	-0.04	9	2.367	6911.7
15	20	(2, 6)	10	1	2.05	6	0.347	6827.9	0.02	9	2.009	6948.3
15	20	(2, 6)	12	1	0.50	9	0.343	6805.1	0.00	10	2.006	6939.0
30	40	(5, 15)	15	1	10.21	0	1172.592	61616.0	8.47	0	1857.768	61650.0
30	40	(5, 15)	17	1	10.00	0	1231.070	61606.7	8.23	1	1863.682	61650.0
30	40	(5, 15)	20	1	8.21	0	442.893	61572.1	6.89	0	1046.981	61650.0
30	40	(5, 15)	25	1	4.51	0	17.915	61470.6	3.13	3	614.325	61650.0
40	60	(7, 20)	20	1	7.65	0	15473.119	149543.6	7.65	0	25195.374	149600.0
40	60	(7, 20)	22	1	8.37	0	17839.147	149543.4	6.20	0	28403.259	149600.0
40	60	(7, 20)	25	1	7.76	0	12550.690	149523.2	6.64	0	23011.490	149600.0
40	60	(7, 20)	30	1	6.47	0	99.977	149451.2	4.35	1	11787.061	149600.0

Table F.3 Performances of Beam Searches using all F_1 & F_2 as Beam Evaluation Function

Setting					Parallel				Pooled			
N	T	(min,max)	C	D	% dev.	# best	CPU Time (sec.)	Avg. Nodes	% dev.	# best	CPU Time (sec.)	Avg. Nodes
10	10	(2, 4)	4	1	7.85	3	0.031	1635.0	11.41	2	0.110	1747.5
10	10	(2, 4)	5	1	6.38	3	0.031	1705.3	10.12	2	0.115	1834.8
10	10	(2, 4)	6	1	4.00	6	0.033	1726.4	4.91	5	0.115	1840.0
10	10	(2, 4)	7	1	0.00	10	0.033	1757.3	1.00	9	0.104	1823.2
15	20	(2, 6)	6	1	13.59	0	0.330	6190.9	13.99	0	2.789	6540.3
15	20	(2, 6)	8	1	13.30	0	0.319	6353.6	13.78	0	2.807	6566.8
15	20	(2, 6)	10	1	10.21	3	0.316	6483.2	12.25	2	2.864	6697.1
15	20	(2, 6)	12	1	5.21	3	0.321	6590.1	7.30	3	2.905	6738.3
30	40	(5, 15)	15	1	17.28	0	15.658	53259.6	17.28	0	496.492	55091.8
30	40	(5, 15)	17	1	20.78	0	15.593	54441.0	20.78	0	499.039	55936.7
30	40	(5, 15)	20	1	24.31	0	15.498	55776.1	24.31	0	502.080	57245.7
30	40	(5, 15)	25	1	24.75	0	15.369	57485.9	24.75	0	505.117	58708.9
40	60	(7, 20)	20	1	15.26	0	86.217	134785.2	15.26	0	4243.111	136161.0
40	60	(7, 20)	22	1	17.41	0	85.612	135940.6	17.41	0	4253.202	136845.5
40	60	(7, 20)	25	1	19.60	0	84.886	137282.4	19.60	0	4264.585	139188.5
40	60	(7, 20)	30	1	21.01	0	83.387	137371.3	21.01	0	4250.770	138648.5