A COMPUTATIONAL STUDY ON A TIME-SENSITIVE MULTIOBJECTIVE FLEXIBLE JOB SHOP SCHEDULING PROBLEM

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

BY

CANER OĞUZKAN

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

NOVEMBER 2017

Approval of the thesis:

A COMPUTATIONAL STUDY ON A TIME-SENSITIVE MULTIOBJECTIVE FLEXIBLE JOB SHOP SCHEDULING PROBLEM

submitted by **CANER OĞUZKAN** in partial fulfillment of the requirements for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. M. Gülbin Dural Ünver ____________________
Director, Graduate School of **Natural & Applied Sciences**

Prof. Dr. Yasemin Serin ____________________
Head of Department, **Industrial Engineering**

Assist. Prof. Dr. Bahar Çavdar ____________________
Supervisor, **Industrial Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Meral Azizoğlu ____________________
Industrial Engineering Dept., METU

Assist. Prof. Dr. Bahar Çavdar ____________________
Industrial Engineering Dept., METU

Assist. Prof. Dr. Sakine Batun ____________________
Industrial Engineering Dept., METU

Assist. Prof. Dr. Serhat Gül ____________________
Industrial Engineering Dept., TED University

Prof. Dr. Ömer Kırca ____________________
Industrial Engineering Dept., METU

Date: November 13, 2017

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: Caner Oğuzkan

Signature :

ABSTRACT

A COMPUTATIONAL STUDY ON A TIME-SENSITIVE MULTIOBJECTIVE FLEXIBLE JOB SHOP SCHEDULING PROBLEM

Oğuzkan, Caner

M.Sc., Department of Industrial Engineering

Supervisor : Assist. Prof. Dr. Bahar Çavdar

November 2017, 119 pages

In this thesis we focus on a time-sensitive flexible job shop scheduling problem. In time-sensitive systems, the main concern is more on the total completion time, which includes time spent during the computation-only phase and the implementation of the solution, rather than finding a solution which will take the least time to implement. However, the conventional solution approaches do not directly address the computation time. In this study, we employ a Computation-Implementation Parallezation (CIP) approach, which was originally introduced for routing problems, in order to find more time-efficient solutions in a multiobjective flexible job shop scheduling problem.

This is the first study to implement the CIP approach on a multi-objective problem. Moreover, we implement the CIP approach on a problem where there are precedence relations between the operations. Therefore, our results provide a further understanding of the benefits of CIP under the tradeoff between different objectives, and also the limitations of embedding the computation time into the implementation in a more challenging problem setting.

We perform extensive computational experiments on many different scenarios based on different instance sizes, flexibility of the processing tools, processing times of operations and compare a base solution method with its CIP implementation. Our results show that CIP approach can provide considerable improvement in the solution quality without increasing the computation-only time or we can find better solutions using the same computation-only time in a multiobjective flexible job shop scheduling problem.

Keywords: Flexible Job Shop Scheduling, Computation Implementation Parallelization, CIP, Total Completion Time, Heuristic

ÖZ

ZAMAN DUYARLI ÇOK AMAÇ FONKSİYONLU ESNEK TİPLİ ATÖLYE ÇİZELGELEME PROBLEMLERİ ÜZERİNE HESAPLAMA ÇALIŞMASI

Oğuzkan, Caner

Yüksek Lisans, Endüstri Mühendisliği Bölümü

Tez Yöneticisi : Yrd. Doç. Dr. Bahar Çavdar

Kasım 2017, 119 sayfa

Bu tezde zaman duyarlı birden çok amaç fonksiyonlu esnek tipli atölye çizelgeleme problemleri üzerine çalışıldı. Zaman duyarlı sistemlerde asıl amaç, daha kısa bir uygulama zamanı olan çözümü bulmaktansa hesaplama zamanını ve çözümü uygulama zamanından oluşan toplam tamamlama zamanını kısaltmaktır. Geleneksel çözüm yöntemleri ise direkt olarak hesaplama zamanını hedef almamaktadır. Bu çalışmada çok amaç fonksiyonlu esnek tipli atölye çizelgeleme problemlerinde zaman-verimli çözümler elde etmek amacıyla Hesaplama ve Uygulama Zamanını Paralelleştirme adı verilen ve ilk olarak rotalama problemleri için uygulanan bir yaklaşım kullanılması önerilmiştir.

Bu çalışma, Hesaplama ve Uygulama Zamanını Paralelleştirme yöntemini birden çok amaç fonksiyonlu bir problem üzerinde ilk kez kullandığından önem taşımaktadır. Ayrıca bu tezde Hesaplama ve Uygulama Zamanını Paralelleştirme yöntemi öncelik ilişkisi barındıran esnek tipli atölye çizelgeleme problemlerinde kullanılmıştır. Bu sayede bu yöntemin birden fazla amaç fonksiyonunun arasındaki ilişkideki yararları ve daha zorlu bir problemde hesaplama zamanını uygulama zamanına yerleştirmekteki kısıtlamaları hakkında daha iyi bir anlayış sağlanacaktır.

Deneyler farklı problem büyüklüğü, farklı esnekliklere sahip araçlara sahip vb. örnekler üzerinde uygulanmıştır ve ana algoritma ve onun Hesaplama ve Uygulama Zamanını Paralelleştirme uygulaması arasında karşılaştırma yapılmıştır. Deneylerin sonucunda, Hesaplama ve Uygulama Zamanını Paralelleştirme uygulamasının tabu arama algoritması tarafından bulunan çözümleri çözüm zamanını artırmadan veya eşit çözüm zamanına sahip olarak önemli bir oranda iyileştirdiği görüldü.

Anahtar Kelimeler: Esnek Tipli Atölye Çizelgeleme, Hesaplama ve Uygulama Zamanını Paralelleştirme, Toplam Tamamlama Zamanı, Sezgisel Yaklaşımlar

To My Family and Friends

ACKNOWLEDGMENTS

I wish to express my deepest gratitude to my supervisor, Assist. Prof. Dr. Bahar Çavdar for her advice, criticism, encouragements and the insight she provided throughout the research.

I would like to thank to my family and friends for their support and encouragement throughout my study.

I also want to thank my managers Kaan Balkan and Akay Kerim İnce for motivating and encouraging me during the study.

This study was supported by the TÜBİTAK under 2211 National Graduate Fellowship Programme. I am grateful to TÜBİTAK for the support.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Progressively developing manufacturing sector has become more competitive in today's world due to high diversification of products. To increase their competitive power, companies focus on effective usage of their resources and try to maintain high customer satisfaction. Customers demand different products with different specifications and they demand products to be delivered within a very short amount of time. Consequently, production variety brings out flexible production and due dates become more strict in production systems. As a result, making the best use of the time between receiving the customer order and the delivery time becomes more crucial in time-sensitive flexible production systems. Manufacturers aim to minimize the time between receiving the customer order and the delivery time to satisfy customer due dates and maximize machine utilization. This duration can be minimized by a better schedule of jobs, so that jobs can be completed in a smaller amount of time. These concerns are addressed by scheduling to assign operations to machines and determine the start time and the sequence of each operation in its assigned machine. Even though the scheduling problems occur mainly in production environments, there are other many applications such as assigning computing jobs to machines and flight deck scheduling. These problems can be considered as job shop scheduling problem or flexible job shop scheduling problem.

Job shop scheduling problem is a classical operations research problem. In the classical job shop scheduling problem, jobs consist of different operations (tasks). Each operation can be performed on a single machine with a fixed processing time. Garey and Johnson (1979) claim that classical job shop scheduling problem having jobs with more than three operations is an NP-hard problem in the strong sense in terms of computational complexity.

Customized demands and high diversification of products could not be satisfied with mass production systems. Flexible job shop manufacturing gains more importance to provide flexibility to the manufacturing systems instead of job shop scheduling problem. In flexible job shop manufacturing, the machines are equipped with several tools, so operations having different tasks can be performed on a single machine. In flexible scheduling environments, production instances change substantially for every production phase and each phase needs to be solved. In the survey research by Chaudhry and Khan (2016), Flexible Job Shop Scheduling Problem (FJSSP) is defined as an extension of classical job shop problem that allows an operation to be processed in any machine that belongs to the alternative machine set of that operation. In a flexible job shop production area, every job may have different machine sequence that it can be processed and a route in the layout. There are operations of jobs and these operations required to be completed depending on a defined precedence relationship. The aim of FJSSP is to find the assignment of operations/tasks to a single machine and find the performing time of operations according to a pre-specified objective function. FJSSP is an NP-hard problem in strong sense. Due to the computational difficulty for finding optimal solutions in practice, there has been a great interest in developing heuristic methods for the problem. The literature for the FJSSP is quite rich. However, heuristics can also take considerably large amount of time to solve for large instances and it may be difficult to find reasonable solutions for time-sensitive systems. Time-sensitive systems are the applications which aim to minimize the total completion time. Total completion time is the total duration between receiving an instance and completing the implementation of the solution. In time-sensitive systems, the computation time of finding a solution can be comparable to the implementation time of the solution. For the time-sensitive systems, the computation time can become as important as the makespan or other traditional criteria and may need to be addressed directly.

In this study we focus on flexible job shop scheduling problems in time-sensitive environments. Contrary to the traditional FJSSP problems, the computation time that is allocated for finding the solution is also considered in the objective function in scheduling problems where the computation time is comparable to the total processing time. These scheduling problems can emerge in production systems where production is based on a specific customer order and/or there are strict due dates for orders. Another example scenario is assigning computing tasks with precedence relationships to computation nodes.

There are many different objectives that are commonly considered in the literature for the flexible job shop scheduling problem, such as minimization of makespan, total tardiness, maximum lateness, average tardiness, total weighted tardiness. The objective may have single criterion as well as it may consist of two or more criteria. Minimization of makespan is generally used when the main goal is to increase the throughput whereas total tardiness, maximum lateness, average tardiness are more relevant when failing to meet the deadlines are very expensive (Mönch et al., 2011). While minimizing makespan considers the entire schedule, minimizing the maximum tardiness considers the due dates of each job. In this study, we use these two objectives together to achieve better schedules both overall and individually, and we also consider the time spent during the computation.

We are focusing on a time-sensitive multi-objective scheduling problem. The objectives are minimizing total completion time and total maximum tardiness. Total completion time consists of the sum of computation-only time and makespan. Makespan is defined as the time elapsed from beginning of performing the first job to the completion of the last job. Total maximum tardiness is the maximum delay of the jobs considering not only the time passing during the processing but also during the computation. Conventionally, when we are computing these performance measures, we only consider the time passing during implementing the solution. However, the time spent during the computation counts as well when the aim is minimizing the time between receiving the customer order and completing the production. Consequently, computation time becomes especially important for real-time problems and for the deadline concerns. On time-sensitive scheduling problems, taking the time spent during computation into consideration may be more appropriate to judge the solution quality. In this thesis, we study a time-sensitive FJSSP.

Since computation time is directly addressed in our flexible job shop scheduling problem, we bring a new perspective to the conventional FJJSP and implement the solution methods by using a different approach which embeds the computation time into the processing time of operations. To embed the computation time into the processing time, we use computation-implementation parallelization (CIP) approach introduced by Çavdar and Sokol (2014). The main idea of this approach is embedding the computation time into processing time by prematurely implementing some parts of the current solution and performing the computation in parallel with the implementation of the partial solution instead of making computation and implementation sequentially. The goal is to decrease the total completion time. As the base solution method, we use the tabu search algorithm used by Billaut and Vilcot (2011) to implement CIP approach on. This tabu search algorithm is developed for multi-objective FJSSP including makespan and maximum tardiness and it has shown to be effective on large instances.

This study is important in different aspects. First of all, it is the first implementation of a CIP approach on a multi-objective problem. The results are important to provide insights about the tradeoff between different objectives. Moreover, this is also the first time CIP has been implemented on a problem where there is a precedence relation between operations. Implementing CIP approach requires implementing some part of the solution in advance and that part of the solution cannot be changed later. Having the precedence relation between jobs increases the effect of the previously implemented decisions, which is a harder case for CIP to perform well. Therefore, the results in this study help us better understanding when CIP benefits as opposed to the traditional solution methods.

The remaining parts of this thesis are organized as follows. In Chapter 2, we present the general problem definition of FJSSP and we review the FJSSP literature. In Chapter 3, we present our problem and introduce our partial-solution freezing policies to implement the CIP approach and the structure of the tabu search algorithm that is used as the base solution method. In Chapter 4, we present and discuss our experimental results. In Chapter 5, we make concluding remarks.

CHAPTER 2

LITERATURE REVIEW

Flexible Job Shop Scheduling Problem is an extension of classical job shop problem where each operation can be processed on any machine that belongs to the alternative machine set of that operation. In this chapter, we present the general structure of the problem, we compare exact solution methods and heuristic approaches proposed for FJSSP and we review the literature for the solution techniques in terms of single objective and multi-objective problems.

2.1. The General Structure of FJSSP

In the problem setting, there is a set of m jobs, J, and a set of k machines, M. Each job contains n_j operations and the total number of operations is denoted by n. We denote operation i of job j by $O_{i,j}$. There is a precedence relationship between operations of each job such that operation $O_{i,j}$ has to be completed before starting operation $O_{i+1,j}$. In our problem setting, operations have a fixed processing time denoted by $p_{i,j}$ and each job has a strict due date denoted by d_j.

In flexible job shop problem, each operation can be performed on multiple machines. We denote the set of machines on which operation $O_{i,j}$ can be performed by $A_{i,j}$. The cardinality of $A_{i,j}$, $|A_{i,j}|$,is the number of machines operation $O_{i,j}$ can be performed on. As the machines get more flexible, $|A_{i,j}|$ increases. We assume that processing times are only operation dependent and do not change according the machine.

The first paper on FJSSP is by Brucker and Schlie (1990). They describe the problem's feasibility by two different conditions. They define the parameters $C_{i,j}$ as the completion times of each operation $O_{i,j}$ and $\mu(O_{i,j})$ as the assigned machine of $O_{i,j}$. The feasibility conditions of the problem are:

i. $C_{i,j} \leq C_{i+1,j} - p_{i+1,j}$ for $i = 1, \ldots, n_j - 1;\ j = 1, \ldots, m$

ii. $\mu(O_{i,j}) \neq \mu(O_{k,l})$ for $O_{i,j} \neq O_{k,l}$

where $[C_{i,j} - p_{i,j},\ C_{i,j}] \cap [C_{k,l} - p_{k,l},\ C_{k,l}] \neq \emptyset$

The first condition implies that the completion time of an operation minus its processing time should be greater than or equal to its predecessor's completion time. In other words, an operation can start only after its predecessor operation is completed. The second condition implies that two operations having overlapped processing intervals cannot be assigned to the same machine. Brucker and Schlie (1990) state that a feasible schedule should satisfy these two conditions and they provide the general assumptions of the FJSSP, which we include in our problem setting, as follows:

1. Each operation can be processed by one machine.
2. Each machine can process one operation at a time.
3. There is no preemption of operations.

In addition to these assumptions, Chaudhry and Khan (2016) provide the following three assumptions which we also consider in our problem:

4. All machines and all jobs are available at time t = 0.
5. Each job's operations are independent from the other job's operations. There is no precedence relationship among the operations of different jobs.
6. Transportation time of jobs between the machines and setup time for processing a particular operation, like tool changes, are included in the processing time.

2.2. Mathematical Model of FJSSP

The mathematical model of the FJSSP we present is constructed by adapting the model developed by Fattahi et al. (2007). We modify their mathematical model processing times are not machine dependent in our problem whereas their model is constructed for machine dependent processing times.

The parameters of the problem are as follows:

m = number of jobs

k = number of machines

$$a_{i,j,h} = \begin{cases} 1, \text{if operation } O_{i,j} \text{ can be performed on machine } h \\ 0, \text{otherwise} \end{cases}$$

$p_{i,j}$ = processing time of operation $O_{i,j}$

L = a big number.

The decision variables of the mathematical model are shown below:

$$y_{i,j,h} = \begin{cases} 1, \text{if operation } O_{i,j} \text{ is assigned to machine } h \\ 0, \text{otherwise} \end{cases}$$

$$x_{i,j,h,p} = \begin{cases} 1, \text{if operation } O_{i,j} \text{ is performed on machine } h \text{ with sequence } p \\ 0, \text{otherwise} \end{cases}$$

$t_{i,j}$ = start time of the operation $O_{i,j}$

$Tm_{h,p}$ = start time of the machine h for the operation in sequence p

k_h = number of assigned operations to machine h

T = total computation time spent to solve the problem

TC_{max} = total completion time of the solution

TL_{max} = total maximum tardiness of the solution.

We provide the assumptions of our FJSSP in the general structure of FJSSP subsection. Under those assumptions and the notations given above, a mixed integer linear programming (MILP) formulation of the problem is the following:

$Min\ C_{max}$

$s.t.$

$$C_{max} \geq t_{n_j,j} + p_{n_j,j} \ \text{ for } \ j = 1, \dots , m; \tag{1}$$

$$t_{i,j} - p_{i,j} \leq t_{i+1,j} \ \text{ for } \ i = 1, \dots , n_j - 1 \,; j = 1, \dots , m; \tag{2}$$

$$Tm_{h,p} + p_{i,j} * x_{i,j,h,p} \leq Tm_{h,p+1} \ \text{ for } \ i = 1, \dots , n_j; j = 1, \dots , m; h = 1, \dots , k; \; p = 1, \dots , k_{h-1}; \tag{3}$$

$$Tm_{h,p} \leq t_{i,j} + \left(1 - x_{i,j,h,p}\right) * L \ \text{ for } \ i = 1, \dots , n_j; j = 1, \dots , m; h = 1, \dots , k; \; p = 1, \dots , k_h; \tag{4}$$

$$Tm_{h,p} + \left(1 - x_{i,j,h,p}\right) * L \geq t_{i,j} \ \text{ for } \ i = 1, \dots , n_j; j = 1, \dots , m; h = 1, \dots , k; \; p = 1, \dots , k_h; \tag{5}$$

$$y_{i,j,h} \leq a_{i,j,h} \text{ for } \ i = 1, \dots , n_j; j = 1, \dots , m; h = 1, \dots , k; \tag{6}$$

$$\textstyle\sum_j \sum_i x_{i,j,h,p} = 1 \ \text{ for } \ h = 1, \dots , k; p = 1, \dots , k_h; \tag{7}$$

$$\textstyle\sum_h y_{i,j,h} = 1 \text{ for } \ i = 1, \dots , n_j; j = 1, \dots , m; \tag{8}$$

$$\textstyle\sum_P x_{i,j,h,p} = y_{i,j,h} \text{ for } \ i = 1, \dots , n_j; j = 1, \dots , m; \ h = 1, \dots , k; \tag{9}$$

$$t_{i,j} \geq 0 \ \text{ for } \ i = 1, \dots , n_j; j = 1, \dots , m; \tag{10}$$

$$Tm_{h,p} \geq 0 \ \text{ for } h = 1, \dots , k; p = 1, \dots , k_h; \tag{11}$$

$$x_{i,j,h,p} \in \{0, 1\} \text{ for } \ i = 1, \dots , n_j; j = 1, \dots , m; h = 1, \dots , k; \ p = 1, \dots , k_h; \tag{12}$$

$$y_{i,j,h} \in \{0, 1\} \text{ for } \ i = 1, \dots , n_j; j = 1, \dots , m; h = 1, \dots , k; \tag{13}$$

$$TL_{max} \geq 0 \tag{14}$$

The objective in this model is to minimize the makespan. However, in our problem, we include the computation time in the objective function and our aim is minimizing two objectives, namely total completion time and total maximum tardiness.

2.3. Exact Solution Methods for FJSSP

FJSSP is an NP-hard problem in strong sense as it is stated in the introduction part. A few exact methods are constructed to solve small size FJSSP in literature, but many studies propose heuristic methods since exact methods are not eligible to solve medium and large size problems.

As it is stated before, Brucker and Schlie (1990) define the FJSSP for the first time and the objective function in their study is minimizing the makespan. They show that general FJSSP with two jobs can be reconstructed as a shortest path problem in a network having polynomial number of vertices. After reducing the problem to a shortest path problem, they constructed an algorithm to find the shortest path. They draw attention to size of the problem and they claim that mixed-integer linear programming (MILP) is effective for the problems having two jobs and it is not effective for the problems having three or more jobs.

In the literature, there are MILP formulations proposed for the FJSSP to find lower and upper bounds, however the MILP model constructed by Roshanaei et al. (2013) can solve the problems with eight jobs on seven machines at most and they assert that the previous mathematical models, such as MILP by Fattahi et al. (2007), Ozguven et al. (2010) can solve the instances having four jobs and four machines. In the recent production environments, the scheduling instances are larger and many papers in the literature try to focus on larger instances of FJSSP. Since the exact solution methods cannot solve large problems, many studies propose heuristic methods for FJSSP, such as tabu search algorithm, evolutionary algorithms like genetic algorithm, particle swarm optimization etc. In our study, our focus is on larger instances, which have at least 30 jobs and 20 machines, since the problems in the industry consist of large number of jobs, operations and machines.

2.4. Review on the Heuristics for FJSSP

There are many heuristic methods constructed to solve the FJSSP. The mostly used heuristic algorithms for the FJSSP in the literature are tabu search and evolutionary algorithms.

Tabu search is a local heuristic search method that uses adaptive memory. The adaptive memory feature allows tabu search to explore the neighborhood in an effective way (Glover et al., 2007). The main difference between tabu search and some of the local search techniques is to allow moves which worsen the objective function value with the aim of exploring different neighborhoods. This adaptive memory feature is constructed by using a list, called tabu list, to remember the recently made moves and avoiding going back to local optimums.

Evolutionary algorithms are metaheuristics based on natural evolution, and they use biological evolution mechanisms, such as crossover, selection, reproduction and mutation. Each candidate solution represents an individual of the population and individuals' quality is determined by a fitness function. Genetic algorithm is the most popular evolutionary approach which starts with an initial solution and uses genetic operators to produce offsprings which are expected to have better solutions from their ancestors. Harmony search is another population-based evolutionary stochastic algorithm that is inspired by the behavior of a music orchestra. The harmony in music is analogous to the optimization solution vector whereas the musician's improvisations are analogous to local and global search. (Gao et al., 2016).

In our literature review, we group the heuristic methods according to the objective functions. The objective functions are mainly divided into two subgroups as single criterion and multi criteria objectives.

2.4.1. Single Criterion Problems

The studies on single criterion FJSSP generally focus on minimizing makespan. Chaudhry and Khan (2016) assert that out of 197 research papers, makespan is used as the single objective of a FJSSP in 88 papers, whereas in 78 papers makespan is used in combination with another criterion. Maximum tardiness, mean lateness and total number of tardy jobs are other objectives used in single criterion FJSSP since the main aim is to finish the problem earlier. Since we are focusing on makespan and maximum tardiness in this study, we review the papers with these objectives.

2.4.1.1. Makespan

Brandimarte (1993) constructs a hierarchical algorithm based on a tabu search. The study considers two different objectives; makespan and total tardiness. The solution approach can be applicable for both objectives separately. Therefore, we classify this paper as single objective. The FJSSP is decomposed into two sub problems as routing and job shop scheduling problem and these subproblems are solved with tabu search algorithm. He proposes a two-way information flow between these subproblems. For the initial solution creation, he introduces some dispatching rules, which are constructed on assigning priorities to operations, and he uses the shortest processing time and most work remaining rules in the experiments. He defines four different neighborhoods for the tabu search algorithm and after the experimental results, he concludes that the neighborhood that changes operations on the critical path works best for the makespan objective. In the paper, exact computation times are not provided.

Hurink et al. (1994) also aim to minimize makespan using a tabu-search based algorithm. To create an initial solution, they use a fast heuristic based on insertion techniques. In this fast heuristic, operations of the longest job are assigned to the machines with the minimum workload iteratively. The remaining operations are inserted to the schedule in the order of non-increasing processing time. After assigning all the operations, all possible sequences of operations on the assigned machines are calculated and the sequence resulting in the lowest makespan is chosen. To improve the quality of the initial solution, they use beam-search technique. The main idea of this technique is to examine a fixed number of feasible partial schedules in parallel and improve the existing solution. The initial solution creation techniques are applied to start from a better initial solution, because the initial solution quality plays an important role in the quality of the final solution. They use tabu search algorithm with two different neighborhood structures. Their experimental results show that both neighborhood structures give similar results, and they conclude that an application of tabu search techniques to FJSSP provides excellent results. The CPU times of their proposed algorithms is between 1 hour 40 minutes to 2 hours for instances having 30 jobs and 300 operations. This shows that for larger instances, the computation time requirements are quite demanding.

Pezzella et al. (2008) develop a genetic algorithm for minimizing makespan in FJSSP. Firstly, they assign operations to machines with a localization approach and sequence these assigned operations by the Most Work Remaining, the Most Operation Remaining and the random selection of the next job dispatching rules. After having an initial solution, makespan is computed for each generated chromosomes that corresponds to a feasible schedule, and best of them are chosen by one of three different methods, which are binary tournament, n-size tournament and linear ranking. From the selected schedules, the new generation is created by changing assignment of the operations and changing the sequence of operations in their assigned machine. This algorithm continues iteratively until a pre-defined number of generations is reached. They conclude that their genetic algorithm performs better than the genetic algorithms proposed by Chen et al. (1999), Ho and Tay(2004), Jia et al. (2003) and they claim that their algorithm gives results comparable to the tabu search algorithm constructed by Mastrolilli and Gambardella(1996) in terms of solution quality and computational effort, however they do not provide the computational times of the algorithm.

Pitts and Ventura (2009) present a two-stage tabu search algorithm, which they denoted as TS^2, to minimize makespan of FJSSP. They develop MILP model and they claim that medium and large size problems cannot be solved in reasonable amount of computation time. Therefore, they propose a two stage algorithm with a construction phase at first stage and improvement phase at second stage. Stage I constructs the initial solution by determining the initial feasible routings and initial job sequences. A rescheduling heuristic based on smallest processing time of operations is used to generate initial feasible routings. In order to provide the initial job sequences, a critical path-based heuristic is utilized. At Stage II, tabu search heuristic is used along with efficient pairwise interchange method, linear programming sub-problem formulations, and two job reassignment procedures. Their experimental results show that their proposed algorithm provides solutions close to the optimal solutions and the algorithm improves the solution with a small computation time being equal to 7.2 seconds. However, they work on instances having at most 10 jobs, which is relatively small that we want to focus on.

Li et al. (2011) propose a hybrid tabu search algorithm with a fast public critical block neighborhood structure to minimize makespan in FJSSP. A mix of four machine assignment rules and four operation scheduling rules are developed to improve initial solution quality. They claim that changing the assignment of an operation can lead to near-optimal solution and according to that, they propose three different rules to change machine assignment which are random rule, top-k most work rule and last processing role. They provide an adaptive proportion formula including this three rules aiming that their hybrid algorithm preserves balance between global exploration and local exploitation. For the scheduling neighborhood structure, which includes changing operations of a machine, three insert and swap functions based on public critical block theory are introduced. Their computational results give the result that their algorithm is comparable with the other contemporary algorithms in the literature with regard to the solution quality and computational effort. The running time of their algorithm is small, such as 90 seconds for an instance containing 20 jobs and 15 machines. However, in their instances each job has small a number of operations, and this is one of the main reasons for small computation times.

Zhang et al. (2011) propose a genetic algorithm for minimum makespan FJSSP. They construct Global Selection and Local Selection methods to create a good initial solution by assigning operations to machines due to the processing times of operations and workload of machines. Machine Selection and Operation Sequence method, which is an improved chromosome representation method, is used for reducing the cost of decoding and avoiding repair mechanism. Furthermore, they use different methods for crossover and mutation operator, including the changes of machine selection and operation sequence, to create chromosomes having better objective value from the current ones. Their experimental results show that their genetic algorithm generates same level of solutions or in some cases better solutions in comparison with other genetic algorithms in the literature like the genetic algorithm proposed by Ho and Tay (2004).

2.4.1.2. Total Tardiness

Scrich et al. (2004) develop two different algorithms based on tabu search to minimize the total tardiness in FJSSP. The first algorithm is called hierarchical algorithm and the other one is named as multi-start tabu search algorithm. Both of the developed algorithms use four different dispatching rules to create an initial solution. For the scheduling problem, the neighborhood includes all solutions created by arc inversions on the critical path of a job whereas for the routing subproblem, the neighborhood consists of all solutions created by reassigning an operation to another machine. In the scheduling problem, the best move is determined by the one that provides largest reduction in total tardiness. On the other hand, the best move is selected by the weighted sum of the total tardiness and the total load of the solution that results from the reassignment in the routing subproblem. Furthermore, some diversification techniques based on the frequencies of the sequences which operations occupy in each machine are developed by them. Their computational results illustrate that hierarchical approach performs well in general whereas the multi-start tabu search algorithm performs well in large instances with a lower computational effort.

Na and Park (2014) propose a genetic algorithm to minimize total tardiness in FJSSP with multi-level job structures. They mention that after production plans are determined by material requirement planning system in companies, the FJSSP with multi-level job structures arises. In their genetic algorithm, compositive chromosomes consist of machine selection, job sequencing and operation prioritization are preferred. They create solution populations to improve their objective value and the convergence speed of the suggested genetic algorithm. They used for different genetic operators, such as selection, crossover, mutation and replacement. They test their algorithm on randomly generated instances and conclude that their algorithm is effective on problems having multi-level job structures.

2.4.2. Multi-objective Problems

There are many papers that focus on multi-objective FJSSP due to the need of optimizing several criteria simultaneously. When multiple objectives are considered, the FJSSP becomes even more complex to solve, and computation takes longer.

Kacem et al. (2002) develop a Pareto-optimality approach based on the hybridization of Fuzzy Logic and evolutionary algorithms. Their multi-objective problem considers minimizing the makespan, total workload of machines and the workload of most loaded machine. Total workload of machines is tried to be minimized when machine efficiencies of performing an operation differs in FJSSP problems. In order to evaluate the solution quantity, they reduce the objective function into a single fitness function. They utilize fuzzy multi-objective evaluation, which starts by determining a lower-bound set for the objectives, in order to decide the selection of the new individuals of evolutionary algorithm. They conclude that their computational results show that they get good solutions in reasonable amount of times.

Billaut and Vilcot (2008) focus on a two-criteria objective in FJSSP consisting of makespan and maximum lateness and they aim to find an approximation of the Pareto frontier. For this aim, they propose two genetic algorithms based on NSGA-II framework, which was used in previous studies. The first algorithm generates initial population randomly whereas the second algorithm generates partially initialization by tabu search algorithm. In the NSGA-II algorithm, all initial solutions are evaluated due to the non-dominated level method and binary tournament is applied for selection whereas crossover and mutation operators are used to create offsprings. They test their proposed algorithm on Hurink (1994) instances and they conclude that the algorithm that uses tabu search algorithm into the initial population of the genetic algorithm performs better in terms of solution quality and computational effort.

Similar to the objective of Kacem et al. (2002), Gao et al. (2008) aim to minimize makespan, the workload of the most loaded machine and total workload of the machines by proposing a new hybridized approach, which consists of genetic algorithm and variable neighborhood descent algorithm. In their genetic algorithm, two-vector representation of solutions is preferred, which are machine assignment vector and operation sequence vector. They use a priority-based decoding, which allocates operations to a machine one by one in the order represented by the operation sequence vector to translate chromosomes into feasible schedules and crossover operators, allele-based mutation and immigration mutation operators are implemented to adapt to the special chromosome structure. After generating new offsprings, i.e. new schedules, variable neighborhood descent is used to enhance the quality of these new schedules by moving one operation or moving two operations. They test their algorithm on 181 benchmark instances. For 119 instances, they find same results as found in the previous studies and better solutions are found for 39 instances. For Hurink (1994) data, their algorithm spends a total of 70745 seconds for 129 instances and 548.5 seconds on average for each instance. Hurink data set consists of instances having 50 operations to 300 operations and the largest instance has 30 jobs and 10 machines. They claim that their proposed genetic algorithm is time consuming since it is a multipoint stochastic search method.

Billaut and Vilcot (2011) aim to minimize makespan and maximum lateness in FJSPP and also they add a third criterion as maximum tardiness in their experimental results. They try to find a set of Pareto optimal solutions. They propose two different tabu search algorithms. For both tabu heuristics, they use a two-step greedy algorithm to generate initial solution. The first tabu search algorithm, named as ϵ-constraint approach, sets a bound for maximum tardiness as the value ϵ and aims to minimize makespan with trying to find a maximum lateness lower than or equal to the value ϵ. The second tabu search algorithm, which we use in our study as our base solution method, evaluates each solution by a linear combination of two criteria makespan and maximum lateness. Two coefficients are used to normalize the linear combination and if the algorithm cannot find any improvement for some iterations, these coefficients are changed and it starts to search the neighborhood from the best known solution. They test their algorithm on Hurink (1994) instances and they illustrate that linear combination tabu search algorithm performs better than the ϵ-constraint tabu search algorithm in terms of solution quality. The average time spent for Hurink (1994) instances is 369 seconds with the maximum iteration parameter is equal to 500. In comparison to genetic algorithm of Gao et al. (2008), Billaut and Vilcot's tabu search algorithm is a faster algorithm.

Chen et al. (2012) develop a scheduling algorithm based on genetic algorithm and grouping genetic algorithm for multi-objective FJSSP. They take a weapon producing company as their case study and they focus on minimizing total tardiness, total machine idle time and makespan regarding to the industry requirements. In their problem structure, precedence relationship depends on bill of material so that operations having the same parent node can be performed parallel on different machines. Their algorithm is composed of two major algorithms. Grouping genetic algorithm is developed to find machine assignments of operations whereas genetic algorithm is applied to find the sequence of operations at their assigned machines. The computational experiments show that their algorithm outperforms the current algorithm of the company but they do not discuss and provide their algorithm's computation time.

Gao et al. (2016) construct a discrete harmony search algorithm for two-criteria FJSSP. Their objective is to minimize the weighted combination of makespan and mean of earliness and tardiness. To create the initial solution, they use several rules. In order to determine the machine assignment, they use random rule, global minimum-processing time rule of Pezzella et al. (2008), two-step greedy rule of Billaut and Vilcot (2011) and hybridization of minimum-processing time and local minimum-processing time rule. Random rule, most work remaining rule, most operations remaining rule and shortest processing time rules are used for operation scheduling. A new rule for the improvisation to produce a new harmony, which is creating a new feasible solution, is developed by changing machine assignment and operation sequencing. Moreover, they use several local search methods to improve the local exploitation ability of the algorithm. They test the proposed algorithm on 49 benchmark instances and they claim that their algorithm is very competitive in comparison to existing algorithms in the literature in terms of solution quality, however they do not provide their CPU time.

For our knowledge, there is no study on time-sensitive FJSSP with the aim of minimizing total completion time and minimizing total maximum tardiness. In this study, we focus on time-sensitive FJSSP. Our study focuses on not only finding better results on classical objectives, but also getting good results in a small amount of computation time. Our study provides a notion for time-sensitive FJSSP by including computation time in the objective function. In order to minimize total completion time and total maximum tardiness, we use an approach called Computation Implementation Parallelization (CIP) which embeds the computation time into the implementation of the solution. Using CIP, we can improve an existing solution method. As our base solution method, we choose Billaut and Vilcot's (2011) linear combination tabu search algorithm because it is fast and efficient. Furthermore, it considers both makespan and maximum lateness which we consider to be crucial objectives in time-sensitive scheduling problems.

CHAPTER 3

PROBLEM DEFINITION

In this thesis, we focus on a multi-objective flexible job shop scheduling problem where the computation time is directly addressed. Unlike the FJSSP problems defined in the literature, we focus on time-sensitive systems, where there is a limited time between receiving the orders and delivery of the products or services by including the computation time in our objective. Time-sensitive flexible job shop scheduling systems may emerge when jobs are processed after customers have placed the orders with strict due dates. There are multipurpose machines equipped with several tools so that some operations can be performed on more than a single machine.

In time-sensitive applications, how we compute the makespan and maximum tardiness in our scheduling problem differs from the classical way since we also consider the computation-only time in the objective function. We refer the addition of makespan and computation-only time as total completion time. Similarly, we name the maximum tardiness including computation time as total maximum tardiness. In our thesis, the objective is to minimize total completion time and total maximum tardiness where we include computation time in both of these objectives. Completion time of job J_j is denoted by C_j. The formula of total completion time, TC_{max}, including the computation-only time, T, is:

$$TC_{max} = T + C_{max} = T + max_{J_j \epsilon J}(C_j) \qquad (1)$$

Total maximum tardiness is related to the due dates of the jobs. The computation-only time is a time interval that directly affects the total maximum tardiness, since implementation starts after the idle-computation time. In our problem, total maximum tardiness, TL_{max}, is computed as follows:

$$TL_{max} = \ max_{J_j \epsilon J}\left(T + \ C_j - d_j, 0\right) \qquad (2)$$

In this study, the general approach aims to shorten the total completion time. Conventional solution approaches, where the computation is not directly addressed, are more suitable when the computing requirements are not that tight. If the system can allow time for computation, conventional solution methods can be used as they are to find better solutions. However, in time-sensitive systems the tradeoff between the computation time and the solution quality becomes more important since production needs to be started as the instance is received. To handle this tradeoff, we propose to use computation-implementation parallelization approach.

CIP approach is proposed to decrease total completion time by embedding the computation into the implementation (Çavdar and Sokol, 2014). Instead of a single computing phase, computation is done in smaller parts and these computation parts are in parallel with the implementation of the solution except for the first computing step. This approach is illustrated in Figure 1. As shown in the figure, the first computation is called as computation-only time or idle-computation time. After computation-only time is passed, some part of solution is finalized and started to be implemented. Correspondingly, the first implementation phase and the second computation phase start simultaneously (Çavdar and Sokol, 2015). Each iteration's elapsed time is equal to the maximum of computation and implementation time in Çavdar and Sokol's CIP approach. In our flexible job shop scheduling problem, we make computation until the implementation time ends. Consequently, in our approach elapsed time of each iteration equals to implementation time of that iteration.

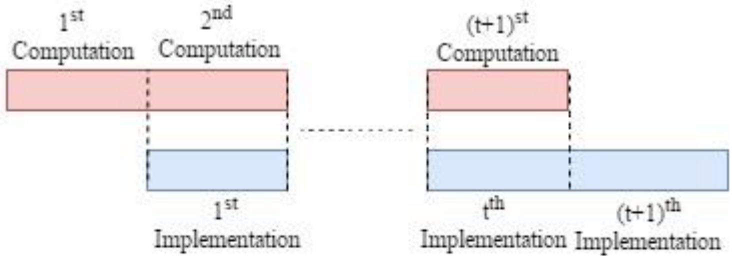

Figure 1 – Computation Implementation Parallelization. Adapted from Çavdar, Bahar, and Joel Sokol. "TSP Race: Minimizing completion time in time-sensitive applications." European Journal of Operational Research 244.1 (2015): 47-54.

Basically, using CIP we can re-construct the implementation of any solution method while obeying the underlying solution mechanism to make better use of the total available time. Once a base solution method is chosen, computation-implementation parallelization can be done as follows:

First computation time may be used to create an initial solution and improve the solution on hand. After creating the initial solution, we need to decide to allocate some more computation time in order to improve the initial solution or implement some part of the initial solution directly. If we allocate some time to base solution algorithm to improve the initial solution on hand, we need to choose which operations are started to be processed and which are not after that time amount runs out. The operations that are started to be processed are called fixed or frozen operations. These fixed operations are eliminated from the computation set and we cannot use them in further computations.

We start to implement fixed operations and simultaneously, we start to use our base algorithm for the remaining unfixed operations in parallel. The second computation time is equal to the first implementation time of fixed solutions. This method continues iteratively until the pre-determined number of iterations is reached. By this approach, just computation-only time, denoted by T, is spent to make just for computation. The following computations are parallelized to the production process and the computation time of them are embedded to implementation. The pseudocode of this method is visualized in Figure 2.

The CIP approach has two goals. The first aim is to reach the same solution quality of the base solution method by shortening its computation-only time. The second goal is to improve the solution quality found by the base solution method without increasing the idle-computation time.

The implementation of the CIP approach (illustrated in Figure 2) on our time-sensitive FJSSP can be represented by a multi-state mathematical model. Assume that there are k fixing decisions (i.e., finalizing a part of the current solution). After the fixing decision, we run the base solution method without allowing any changes on the fixed part. Fixing decision includes the assignment of the operations to machines and determining their start time. Fixing decision in the i^{th} stage is denoted by fd^i and cumulative fixing decisions including that of i^{th} stage are denoted by cfd^i. The feasible solution set of the problem is determined by the fixing decisions and the set of all feasible solutions given the fixing decision is denoted by $S \mid cfd^i$. In our CIP implementation, we run the base solution method on $S \mid cfd^i$ for the time allocated for i^{th} stage and this time interval is denoted by T^i. The objective of our modified FJSSP problem is minimizing the linear combination of total completion time and total maximum tardiness where allocated computation time is determined. The problem is expressed by $Min_x\,(\propto TC_{max} + \beta TL_{max} \mid T^i)$ subject to $x \in S \mid cfd^i$ where x is a solution to the underlying scheduling problem. To solve this problem optimally requires being able to compute the value of the objective function under different computation times. Therefore, we follow a heuristic procedure.

In this study, we use the tabu search algorithm proposed by Billaut and Vilcot (2011) as our base solution method and we propose several partial – solution freezing policies to parallelize the computation time and the implementation time.

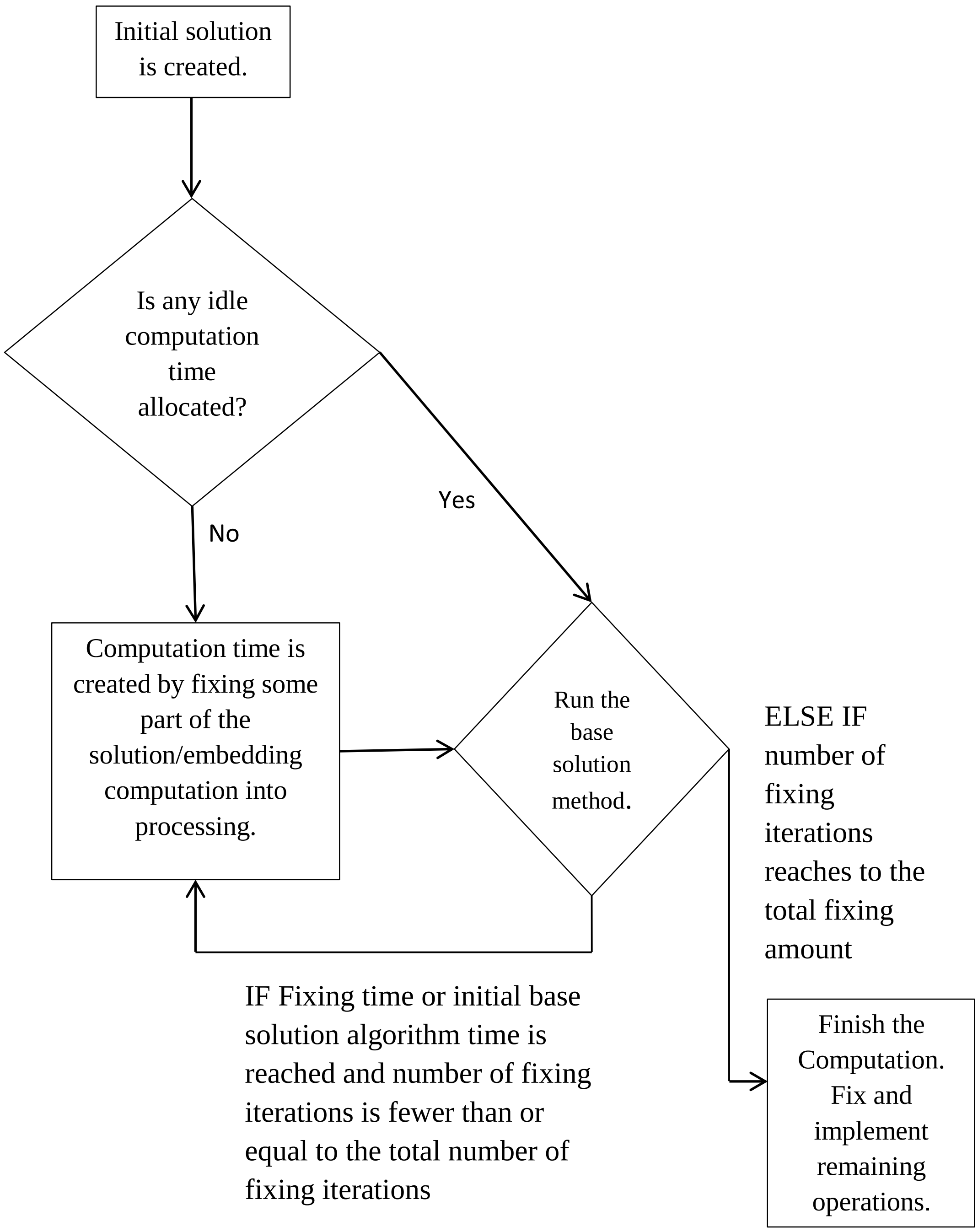

Figure 2 – Illustration of the CIP approach on time-sensitive FJSSP

As it is mentioned before in this chapter, to embed the computation into the implementation, we need to freeze some part of the solution during the computation and implement it. While implementing the partial solution, we continue computing. This is how the parallelization is performed. We call this as partial-solution freezing or partial-solution fixing, and different rules can be used for this. Fixing some of the operations creates additional time to make computation in parallel to their implementation time. In that time duration, base solution method tries to improve the solution among the unfixed operations while the fixed operations are being processed. Partial-solution freezing methods affect the solution quality directly due to the following reasons:

(1) Fixed operations are not allowed to move to another machine or another sequence in the same machine.

(2) After fixing some operations, the succeeding operations may need to be shifted further due to the precedence constraints. Idle times can occur as a result of fixing, so the unfixed operations may start later than their current position.

(3) After fixing some operations, the maximum completion time of fixed operations is the total fixed time for that iteration. If there are some operations started before that maximum completion time and have not finished until that time, these operations need to be shifted due to the no preemption constraint. Similar to (2), idle times can occur as a result of fixing decision.

If the fixing decisions are done in a poor manner, in the future computations the local search mechanism may be stuck in a local optimal solution and not be able to explore others. This can prevent the CIP approach from providing any benefits we are aiming. Therefore, the quality of the solution-freezing rules is important regarding the final solution quality.

We introduce four different partial-solution freezing policies and each of these policies focuses on different aspects of our multi-objective problem. Before introducing our partial-solution freezing policies, we will first discuss the details on the base tabu search algorithm by Billaut and Vilcot (2011) in two parts. In the first part, we will explain the initial solution creation method. In the second part, we will explain the tabu search mechanism. Then, the partial-solution freezing policies which are constructed based on the problem attributes and the tabu search mechanism will be presented.

3.1. Creating the Initial Solution

Billaut and Vilcot (2011) introduced a two-step greedy algorithm to create an initial solution. The algorithm is based on assigning each operation to a machine and then determining the sequence of operations in each machine.

In the assignment step, all operations are sorted by $|A_{i,j}|$ in non-decreasing order. If there are any ties between some operations, these tied operations are sorted by $p_{i,j}$ in non-decreasing order. This sorted list is denoted by S_{op}. The machines are also sorted by their workload in non-decreasing order. Workload of a machine is the total processing time of assigned operations to that machine. Initially, the workloads of all machines are equal to 0, and we update workload of machines after we make an operation assignment. The assignment is done by taking the first operation of S_{op} and assigning it to a machine that belongs to $A_{i,j}$ and has the minimum workload. If machines has the same workload, we choose the assigned machine arbitrarily from the set $A_{i,j}$. After an operation has been assigned to a machine, it is removed from the set S_{op} and this assignment process continues until $S_{op} = \emptyset$.

The second step is determining the operation sequence on the machines. Billaut and Vilcot (2011) name the sorting rule as ‘slack’ rule. At this step, the greedy algorithm works by taking the precedence relationship into account. Initially a set of candidate operations S_{cand} is created by the first operations of each job and these operations are sorted by two parameters. The first parameter is the release time. Operations are sorted in non-decreasing order of their release times. Release time of an operation is the initial time that we can start processing that operation $O_{i,j}$. Release time is equal to the maximum value of the time that the machine becomes available to process

$O_{i,j}$ and the time that operation $O_{i-1,j}$ is completed if $O_{i,j}$ has a predecessor operation. The second parameter to break ties in the sorted list is the 'slack' of operation $O_{i,j}$, and it is the difference between the due date d_j and the remaining time of the job J_j as if all remaining operations after $O_{i,j}$ were processed immediately after $O_{i,j}$. After an operation $O_{i,j}$ is scheduled, the set S_{cand} is updated by eliminating $O_{i,j}$ from the set and adding $O_{i+1,j}$ if $O_{i,j}$ has a successor. At the same time, release times of all operations are calculated again. This sequence determination continues until all operations are scheduled, in other words, $S_{cand} = \emptyset$.

3.2. Tabu Search Algorithm Constructed by Billaut and Vilcot (2011)

Billaut and Vilcot (2011) propose a tabu search algorithm which tries to minimize a linear combination of makespan and maximum tardiness in FJSSP. In the algorithm, the neighborhood of an operation constructed as two different ways. An operation $O_{i,j}$ can be swapped with the previous operation in the sequence on the same machine where $i \neq 1$. The other way to swap $O_{i,j}$ is to insert it at another possible machine that is involved in the set $A_{i,j}$. The insertion point at the other machine is the first available point that satisfies precedence constraints. We call the swap of $O_{i,j}$ with the operation at the previous sequence of it as an intra-machine move and the insertion of $O_{i,j}$ to another possible machine as an inter-machine move.

3.2.1. Tabu List

In the algorithm, the tabu list has a fixed size and we add the most recently implemented moves to the tabu list. These moves are stored as vectors in the list and each element of the list is a vector. The tabu list starts empty at the beginning of the algorithm and if the number of stored elements reaches to the tabu size, the oldest element of the list is removed and new elements are inserted into it.

Billaut and Vilcot (2011) use a similar tabu list structure suggested by Dauzère-Pérès et al. (1997). There are three types of tabu lists constructed in the article by Dauzère-Pérès et al. (1997) and Billaut and Vilcot use the one that provides best results. For the simplicity of the notation, the operation which is swapped is denoted by $O_{i,j}$, and its predecessor and successor before the movement are denoted by v and w respectively. The new predecessor and successor of $O_{i,j}$ are denoted by *x* and *y*. Since $O_{i,j}$ is moved between the operations *x* and *y*, this move is denoted by $\{O_{i,j},\ x,\ y\}$. After this move $(x, O_{i,j})$ and $(O_{i,j}, y)$ are added to the tabu list. A move $\{O'_{i,j},\ x',\ y'\}$ is forbidden if $(x', O_{i,j}{}') \in TL$ or $(O'_{i,j}, y') \in TL$.

The size of tabu list, *TLS*, is a parameter to decide in the algorithm. We use a tabu list with fixed size and *TLS* is considered to set to 100 because Billaut and Vilcot (2011) try all values of *TLS* ∈ {10, 40, 70, 100, 150} and they state that their results are better with *TLS* being equal to 100.

3.2.2. Evaluation of Neighborhood

Tabu search algorithm searches the described neighborhood entirely and at each iteration, it chooses the best non-tabu solution denoted by $S^{i}{}_{best}$, where i is the iteration number. Solutions are evaluated according to the linear combination of makespan and maximum tardiness as shown below:

$$Z(S) = \frac{\alpha * C_{max}(S)}{\max(C_{max}) - \min(C_{max})} + \frac{\beta * L_{max}(S)}{\max(L_{max}) - \min(L_{max})} \qquad (3)$$

where α and β are two coefficients. In the Equation (3) $\alpha \in \{0, 0.1, 0.2, 0.3, \ldots 1\}$ and $\beta = 1 - \alpha$. The $\max(C_{max})$ value stands for the maximum value of makespan, and $\min(C_{max})$ value stands for the minimum value of makespan. Similarly, $\max(L_{max})$ denotes the maximum of maximum tardiness value and $\min(L_{max})$ is the minimum of it. These values are found by considering all the best solutions found so far. They are used to normalize makespan and maximum tardiness values in a multi-objective criteria evaluation so that the objective values are comparable. The current iteration is denoted by b and these values are calculated as below:

$$Y = \left\{ S^{i}{}_{best}, \forall\, i, i < b \right\} where\ b > 1 \qquad (4)$$

$$\max(C_{max}) = \max_{y \in Y} C_{max}(y) \tag{5}$$

$$\min(C_{max}) = \min_{y \in Y} C_{max}(y) \tag{6}$$

$$\max(L_{max}) = \max_{y \in Y} L_{max}(y) \tag{7}$$

$$\min(L_{max}) = \min_{y \in Y} L_{max}(y) \tag{8}$$

This evaluation method for each tabu iteration is taken from Billaut and Vilcot (2011). For initialization, we make an assumption so that $\max(C_{max})$ and $\max(L_{max})$ values are equal to the makespan and maximum tardiness values of initial solution and $\min(C_{max}) = \max(C_{max}) - 1$ and $\min(L_{max}) = \max(L_{max}) - 1$. If the solution is improved at the first iteration, we update $\min(C_{max})$ and $\min(L_{max})$ values due to the computation defined above since $b > 1$.

α and β coefficients are chosen randomly in the algorithm and these values can be changed after a number of iterations where no improvement occur in $Z(S)$ value. The number of iterations without improvement is denoted by Max_{iter}. α and β values are changed and chosen randomly again from the defined set after algorithm reaches to the value Max_{iter}. This method is used due to keep tabu search from falling into a local optimum. Since α and β values change during the local search, the algorithm can explore different neighborhoods. This may cause one of the criteria gets worse whereas the other criteria may be improved. After Max_{iter} is reached and α and β values are changed, Max_{iter} value is set to 0 and this method is used again until the algorithm stops.

3.3. Partial-Solution Freezing Rules

Partial-solution freezing rules are used to embed computation time into the processing time of operations. The partial-solution freezing rules take a feasible solution, and fix operations using a pre-defined rule. The fixed operations are started to be processed on their assigned machines instantly. When the fixing decision is made, the unfixed operations are shifted to the time equal to the maximum of the finish time of the fixed operations. Therefore, an idle time may occur on machines and the solution quality may get worse. After a partial-solution freezing rule is applied, the processing time of the fixed jobs determines the computation time until the next fixing step. While the fixed operations are processed, computation continues. The jobs that are fixed are labeled, and they are not allowed to be involved in any moves in the future computations.

Partial – solution freezing policies are iterative methods. We denote the number of fixing steps by K. In each iteration the operations that are fixed are decided using a partial-solution freezing policy. Then, the tabu search algorithm computes further until we use the time allocated for the current step, then another fixing is made until K is reached.

To make partial-solution freezing decisions, we introduce four different policies. All policies are mentioned in detail under the sections below later in this section. These four policies are (i) average process time threshold fixing policy, (ii) tardiness-based fixing policy, (iii) machine cardinality fixing policy and (iv) remaining time fixing policy. Each fixing policy relies on different aspects of the problem into consideration. As it is stated before, fixing may create idle time on the machines and the average process time threshold fixing policy and remaining time fixing policy take into consideration that idle time creation and they are constructed to create smaller idle time by attempting to fix operations of different machines instead of fixing operations assigned to a certain machine. Moreover, we construct the policies according to our performance measures, total completion time and total maximum tardiness. Furthermore, the flexibility of an operation depends on the number of machines it can be processed on, and machine cardinality fixing policy is constructed to fix fewer flexible operations in order to decrease the negative effect on the tabu search algorithm.

The general solution-freezing mechanism and the four policies are explained as follows:

3.3.1. General Solution-Freezing Mechanism

Fixing process can be implemented once or it can be implemented iteratively after the time allocated for base solution method has run out. All policies that we defined in this paper are iterative and the current iteration of the fixing is denoted by k. Each policy finds the first available unfixed operation of each machine and these operations are added to the set O_{avail}. An available operation to fix means that operation has no predecessor or its predecessor is fixed and it is the first operation of its machine sequence. Operations belonging to the set O_{avail} may be decided to fix due to a given rule. After determining which operations to fix, the maximum finish time of the fixed operations is computed, and it is denoted by F_{max}. The algorithm removes the precedence relationship between the fixed operations and their successors. Subsequently, the beginning time of first unfixed operations of each machine is shifted to F_{max} and then all other operations are shifted according to their predecessors in the machine sequence or their predecessors defined in the job sequence. We allocate F_{max} for tabu search in our case. After the search time is finished, if the current iteration number $k \leq K$, we use our fixing policies again for the unfixed operations. For partial-solution freezing, we propose and test the following policies.

3.3.2. Fixing Policies

In this thesis, four fixing policies are introduced and the rules created in these policies are explained in detail in the following sections.

3.3.2.1. Average Process Time Threshold Fixing Policy

This fixing policy uses a threshold-based rule. In each fixing iteration, we compute a threshold in order to determine which operations to fix. The threshold for making the fixing decision according to the average processing time, Th_k, is calculated as:

$$Th_k = k(\frac{1}{n}\sum_{j=1}^{m}\sum_{i=1}^{n_j} p_{i,j}) \quad where\ k \leq\ K \tag{9}$$

For each machine, the first available unfixed operation, i, is found and we fix i if the completion time of that operation is under the threshold value. The condition that we check is formulated as:

$$C_{i,j}\ \leq\ Th_k \tag{10}$$

The algorithm continues to search that machine until it finds an operation satisfying the condition $C_{i,j} >\ Th_k$, and then continues with other machines. Average process time threshold fixing policy is named as Policy 1.

3.3.2.2. Tardiness-Based Fixing Policy

Total maximum tardiness is a part of the objective function. This fixing policy, called as Policy 2, takes the due dates of the jobs into consideration to determine which operations to fix. The aim is to fix $\theta * n$ operations in each iteration, where θ is the proportion of operations to fix. θ is a pre-determined coefficient, where $0\ < \theta <\ 1$. As defined above, the first available unfixed operation of each machine is found and these operations are added to the set O_{avail}. For each operation, $O_{i,j}\ \in\ O_{avail}$, L_j value of J_j is found, where $O_{i,j}$ is an operation of job J_j. The operations are sorted by their job's L_j value in descending order and the operation which belongs to the latest job is fixed. The algorithm updates the set O_{avail} after an operation is fixed and it continues fixing until $\theta * n$ operations are fixed.

3.3.2.3. Machine Cardinality Fixing Policy

Machine cardinality of the operation $O_{i,j}$ is denoted by $|A_{i,j}|$. Since $|A_{i,j}|$ value gets larger, there are more moves for $O_{i,j}$ in tabu search algorithm. Similar to Policy 2, there is a pre-determined coefficient θ, where $0\ < \theta <\ 1$, and each iteration fixes $\theta * n$ number of operations. This policy sorts the available operations, $O_{i,j}\ \in\ O_{avail}$, by their machine cardinality $|A_{i,j}|$ in ascending order and it fixes the operation with the lowest $|A_{i,j}|$ value. It continues fixing until $\theta * n$ operations are fixed as we do in Policy 2. This rule is denoted by Policy 3.

3.3.2.4. Remaining Time Fixing Policy

After each fixing step, the beginning time of unfixed operations are shifted to F_{max} and there may be an idle time on the machines as a result. In order to decrease this idle time, Remaining Time Fixing Policy can be applied where we check the remaining time of the operation to decide to fix the operation or not. Remaining time is calculated as the completion time of the job minus fixed time: $R_{i,j} = C_{i,j} - F_{max}$. The policy initially finds $O_{i,j} \in O_{avail}$ that has shortest processing time and fixes it. F_{max} becomes equal to $p_{i,j}$. Afterwards O_{avail} is updated and a coefficient q is used as a threshold coefficient to determine to fix the operation where q is the percentage of processing time of the operation that we determine to fix. For each $O_{i,j} \in O_{avail}$, the policy checks the condition if $R_{i,j} \leq (p_{i,j} * q)$ and if the operations that satisfy this condition are fixed. After scanning all $O_{i,j} \in O_{avail}$, F_{max} value is updated depending on the completion time of fixed operations. The algorithm continues if any fixing is made after each update of O_{avail} and it stops when there is no operation to fix belonging to O_{avail}. This policy is named as Policy 4.

3.4. Proposed CIP Implementation to Solve FJSSP

We will now explain how the partial solution fixing policies can be used to implement CIP on the base solution method: An initial solution is created by the two-step greedy algorithm and we run base solution method if there is any computation-only time allocated. Afterwards, we choose one of the partial-solution freezing policies and number of iterations of freezing, and initial parameters are set due to our instance properties. The processing starts with the fixed operations on their assigned machines. Simultaneously, unfixed operations are included in the tabu search neighborhood and tabu search algorithm tries to improve the solution with this neighborhood. Tabu search continues until the maximum completion time of fixed operations is reached. When the computation time of tabu search is reached and the current fixing iteration is lower or equal to the total number of fixing iterations, our chosen policy determines another set of operations to be produced. After each fixing iteration, the makespan and maximum tardiness are updated due to the shifting of operations. Tabu search starts again with the new neighborhood and its $\min(C_{max})$, $\max(C_{max})$, $\min(L_{max})$ and $\max(L_{max})$ values are initialized as defined above. The current iteration value b is also initialized as 1. This process continues until the maximum number of fixing iterations is reached. After the last phase of the computation finishes, the remaining jobs are processed according to the final solution.

3.5. An Illustrative Example of the Proposed CIP Implementation

Before continuing with more details on the computational experiments, we present an illustration of the CIP approach on the average process time threshold fixing policy, i.e. Policy1, on an example instance to make the mechanism of the CIP implementation clear. The purpose of this illustration is to demonstrate how the computation starts if there is an available computation-only time is allocated for the base algorithm, how the jobs are fixed on the machines during the computation and how some of the successor operations are delayed if needed. The random example instance we use here, which is Instance36, has 36 jobs, 650 operations that can be run on 30 machines. The number of operations of each job is uniformly distributed between 10 and 30, and the processing time of operations is uniformly distributed on [400, 1400] seconds. The cardinality of operations is uniformly distributed on [5, 15]. The whole data is given in Table 14 in Appendix B. Timing measure is in seconds.

All machines are available to start processing at time 0. In this illustrative example, we allow an initial 300 seconds to our CIP implementation to create initial solution and make tabu search to improve initial solution before the partial solution fixing process. The initial time that we allocate is the computation-only time of our CIP approach. After the computation-only time has finished, we perform two partial-solution fixings iteratively.

The initial total completion time is equal to 46653 seconds and total maximum tardiness is 5885. After 300 seconds of computation, the first and second operations of each machine are shown in Figure 3. The values on the bars are the processing times of operations. The total completion time and total maximum tardiness become equal to 41764 and 4671 at the end of 300 seconds.

As it is mentioned before, Policy1 determines which operations to fix according to the average value of process times of operations. The first threshold Th_1 is calculated as 929 seconds, and consequently, the algorithm fixes the operations which have completion time before 929 seconds. At the first fixing iteration, the start times and process times of the first two assigned operations of each machine are shown in Table 1 and illustrated in Figure 3. Fixed operation ID's are 51, 132, 158, 191, 231, 265, 286, 387, 405 and 526 at the first iteration. These operations are illustrated with bold characters in Table 1 and their start time and process time can be also seen on that table. The succeeding operations of the fixed ones or preempting operations may need to be shifted further. Shifting operations can be seen in Figure 4. After the shifting, total completion time and total maximum tardiness become 42547 and 5481 seconds. It should be emphasized that total completion time and total maximum tardiness get worse after this iteration of the freezing policy because some operations are shifted. The effect of shifting on first two operations of each machine can be observed in Figure 4.

Table 1 – Start and Process Times of First Two Operations of Each Machine (Fixing Iteration 1)

	1st operation on the machine			2nd operation on the machine		
Machine	Oper. ID	Start Time	Process Time	Oper. ID	Start Time	Process Time
M1	**191**	0	519	211	519	497
M2	501	0	1321	512	1321	598
M3	357	0	1150	389	1650	1346
M4	**265**	0	441	633	1771	927
M5	266	441	1137	406	1578	611

Table 1 Continued

M6	388	829	821	94	1650	444
M7	**286**	0	925	21	925	1220
M8	194	3251	943	361	4194	750
M9	**231**	0	863	103	863	718
M10	74	0	1288	502	1321	431
M11	93	0	1123	574	1315	481
M12	232	863	1220	193	2083	1168
M13	**158**	0	500	632	500	1271
M14	**526**	0	688	1	688	1198
M15	358	1150	780	634	2698	801
M16	**51**	0	791	489	942	802
M17	488	0	942	594	943	650
M18	635	3499	573	23	4110	1153
M19	573	0	1315	332	1315	1397
M20	474	0	1104	168	1104	1362
M21	593	0	943	527	943	1392
M22	552	0	1013	421	1013	914
M23	**132**	0	681	490	1744	780
M24	192	519	1047	554	2406	1082
M25	134	1193	1018	513	2211	955
M26	**387**	0	829	435	829	1094
M27	**405**	0	910	504	3028	739
M28	11	0	1058	376	1058	614
M29	243	0	1020	287	1020	859
M30	133	681	512	359	1930	802

The maximum finish time of fixed operations is 925 seconds. So, our implementation continues to the next iteration of tabu search which will be run for another 925 seconds eliminating the fixed jobs, i.e., those jobs cannot be involved in any moves. After 925 seconds of computation, the total completion time and total maximum tardiness become 39349 and 3251 seconds. It should be emphasized that the total completion time and total maximum tardiness are both improved from the objectives that the algorithm finds at 300 seconds even though the current solution temporarily gets worse with the first partial-solution fixing decision.

Since $K = 2$, another fixing is performed. Before the fixing, the new threshold of Policy1 to determine fixed operations, Th_2 is calculated as 1858 seconds and the second fixing decision is made following the same fixing rule. The tabu search algorithm runs for the allocated time for one more time and after that time we reach the final solution and process all the remaining operations according to the final solution. In the final solution, the total completion time and total maximum tardiness are 37986 and 2650 seconds. These values were 41764 and 4671 seconds respectively at the end of initial 300 seconds. By allocating 300 seconds of computation-only time to the base tabu search algorithm and our CIP approach, our approach gives 9.05% better total completion time and 43.27% better total maximum tardiness.

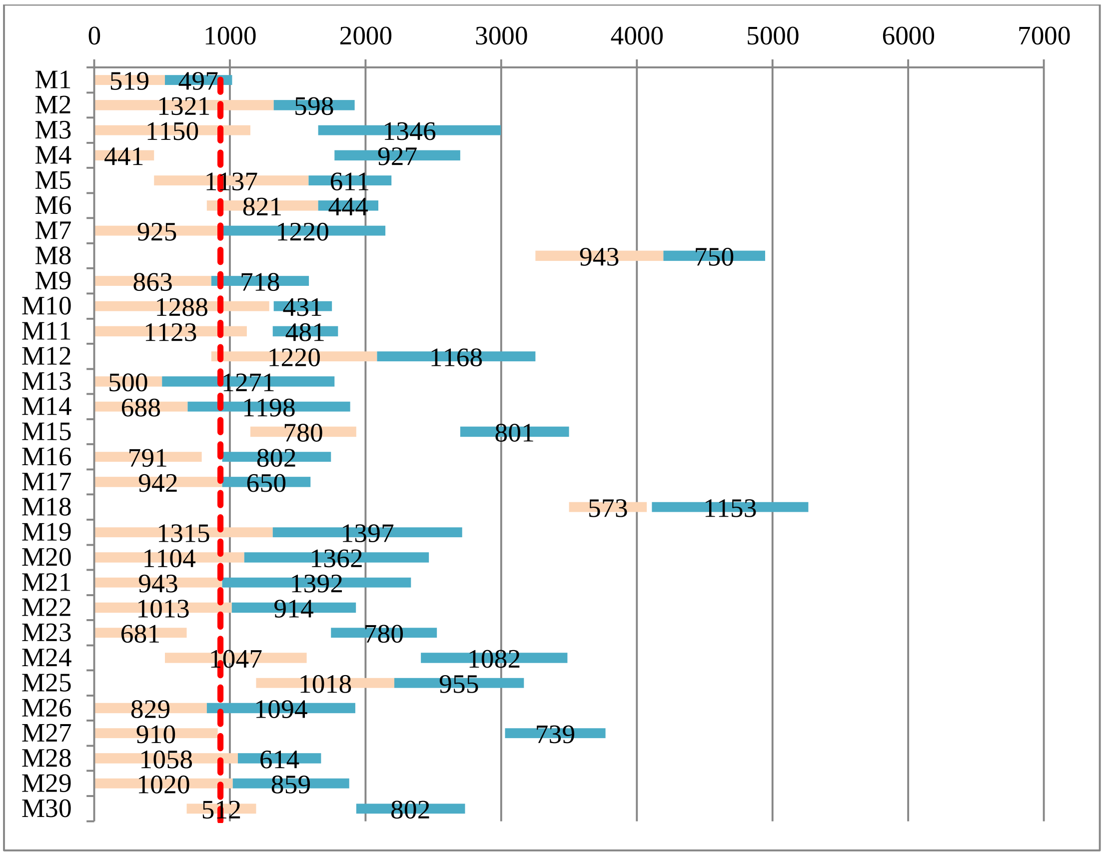

Figure 3 – First Two Operation Assignment of Machines before First Freezing Decision

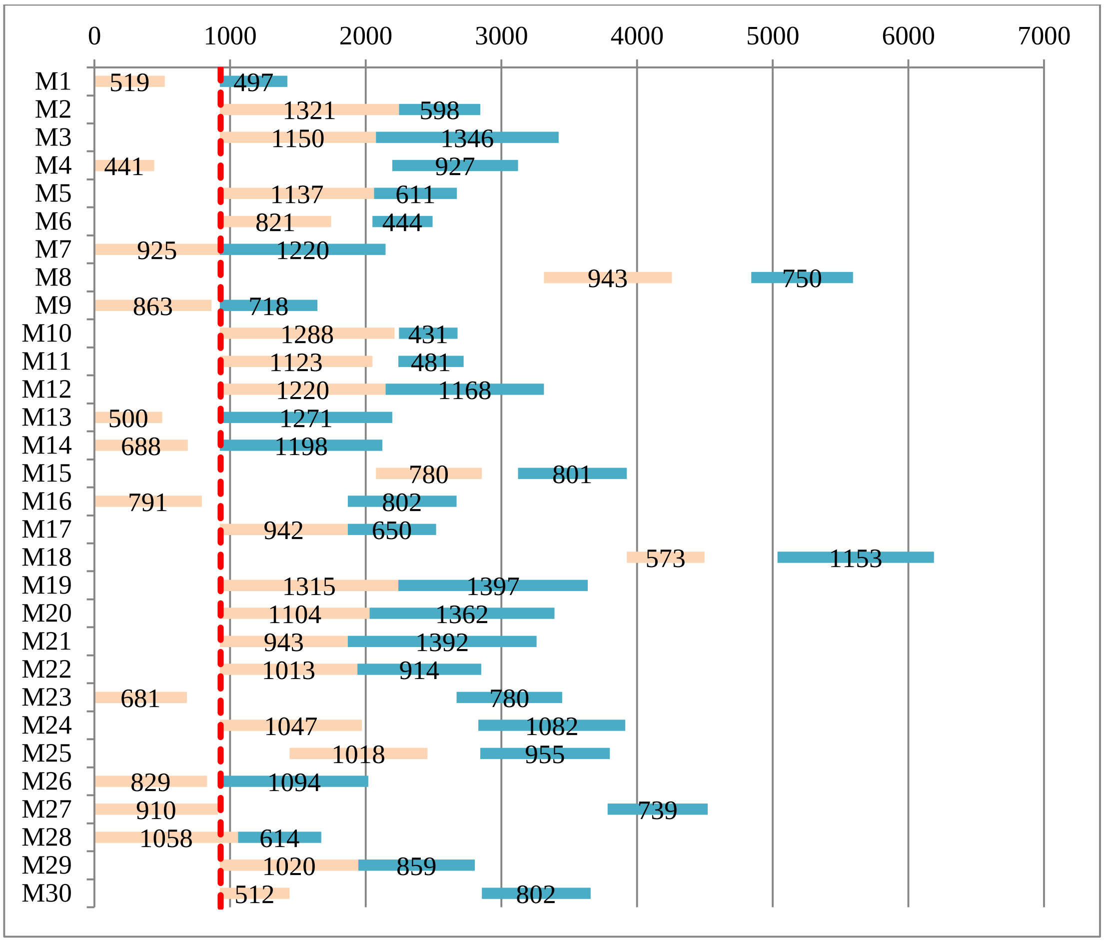

Figure 4 – First Two Operation Assignment of Machines after the First Freezing Decision

CHAPTER 4

COMPUTATIONAL EXPERIMENTS

In this chapter, we present and discuss the computational results of our CIP implementation for the time-sensitive FJSSP. In our computational experiments we compare the conventional implementation of the base solution method (tabu search algorithm by Billaut and Vilcot (2011)) with the CIP implementation of the same algorithm. The conventional implementation of the base algorithm is compute-first and implement-later method whereas in our CIP implementation, computation is performed in parallel with the implementation of operations. Our comparison bases are the total completion time and the total maximum tardiness in a time-sensitive environment.

As it was mentioned in the previous chapter, the CIP approach can provide benefits in two different ways. First, we can shorten the computation-only time of the base solution method without worsening the solution quality. Second, we can improve the solution quality of the base solution method under the same computation time. The experimental settings are determined based on these two goals. We compare the base tabu search algorithm and our CIP implementation on randomly generated instances, which will be described in detail later in this chapter.

Since computation time is also addressed directly in our problem, we include the computation time in the makespan and the maximum tardiness in the CIP implementation. To be able to compare the CIP implementation of the base tabu search algorithm with the original implementation on the same basis, we take the corresponding computation time into account for the original implementation to compute the makespan and maximum tardiness as well.

The main goal of the computational experiments is to test the partial-solution freezing policies and develop an understanding the performances of these policies for different objectives in different scenarios. Additionally, the effect of our CIP implementation on the FJSSP can be observed with the experimental results and consequently, we can report the effects of shifts of operations occurring due to the freezing decisions.

In our computational experiments, initial values of some parameters are set as follows: The initial computation-only time allocated to our CIP implementation is set to 200 seconds. In some of the partial solution freezing policies, we define some initial parameters in order to make the fixing decision or to decide how many operations are fixed. For Policy 2 and Policy 3, we perform tests with θ being equal to either 0.01 or 0.02. We choose small values for θ because bigger θ's imply fixing larger number of operations, and leads to sharp decrease in the neighborhood size. For Policy 4, we also choose small values for the parameter q since in our preliminary experiments we observed that Policy 4 performs better with small q's. In the experiments, q is either 0.1 or 0.2.

Before the details of the computational experiments, we will explain how we generate the random instances we run the experiments on.

4.1. Random Instances for the Computational Experiments

The attributes of the instances affect the computation time requirements and the performance of both solution methods. While creating the random test instances, we cover as many attributes as possible so that we can develop a better understanding of the performance of our CIP implementation. We create the test instance groups based on

(i) different number of jobs,
(ii) different number of machines
(iii) average number of operations in each job,
(iv) number of machines to perform each operation,

(v) distribution of the processing times of the operations.

Number of jobs and number of machines of an instance determine the problem size, consequently the required computation time. Average number of operations in each job is another parameter that affects the problem size.

The flexibility of the machines is also one attribute we test on. The flexibility of a machine is determined by the number of operations it can process. In some of the data sets, operations can be processed by large number of machines and in the other data sets, each operations' machine cardinality is smaller. The set of machines that can process an operation are randomly determined according to the machine cardinality of that operation.

The processing times of operations are uniformly randomly distributed with different parameters. The processing times of operations are important in our problem setting because large and small processing time of operations may bring different results under CIP approach.

According to the properties of instances, we can classify them based on the flexibility, problem size and average processing time of operations. Flexibility of an instance is measured by average number of operations of a job and flexibility of the machines. Flexibility of the machines is measured by average machine cardinality of operations. Number of operations of a job affects the instance flexibility too. When a job has fewer operations, there are fewer precedence constraints. Instance flexibility affects running time because neighborhoods are larger in flexible instances. Better initial solutions can be found when there is less precedence relationship,; it becomes harder to improve its quality. We measure problem size by the number of jobs, number of machines and the number of operations. The last attribute that we use to classify instances is average processing time of operations. Average processing time of operations affects the quality of CIP implementation since it determines the time allocated to computation time after partial-solution freezing.

The instance sets are created by taking a relatively small and a higher value of each attribute. The values of each test attribute are taken as follows:

- The number of jobs, m, is equal to 30 in small instances and 40 in large instances.
- Number of machines, k is equal to 20 or 30.
- Average number of operations in each job, number of machines to perform each operation and processing times of the operations are uniformly distributed.
- Average number of operations in each job is uniformly randomly distributed between 10 and 20 in small instances and 20 and 30 in large instances.
- Number of machines to perform is also uniformly distributed between 5 and 15 in less flexible instances and uniformly distributed between 10 and 20 in more flexible instances.
- Processing times of operations are uniformly distributed between 100 and 700 in one group and between 600 and 1200 seconds in the other group.

There are 5 different attributes and we take 2 different values or distributions for each attribute. Therefore, we generate 2^5 different instance sets in our experiments. For our random experiment, we create 5 instances for each instance set. The instance sets and their attributes are given in the Table 2 whereas all the attributes of 160 instances, including which set they belong to, are presented in Table 15 in the Appendix C. The uniform distribution is abbreviated by $U(a, b)$ in Table 2 and Table 15. Moreover, number of operations of a job is denoted by *Num. of Operations Dist* and number of available machines of an operation is denoted by *Available Mac. Dist.* in these tables. The initial makespan and maximum tardiness of each instance are provided in the Table 13 in the Appendix A in order to provide information about instances. These initial values are computed by the two-step greedy algorithm defined in the Chapter 2.

Table 2 – Properties of Data Sets

Set	#Job	#Mac.	Num. of Operations Dist.	Available #Mac. Dist.	Processing Times
Set1	30	20	$U(10,20)$	$U(5,15)$	$U(100,700)$
Set2	30	20	$U(10,20)$	$U(5,15)$	$U(600,1200)$
Set3	30	20	$U(10,20)$	$U(10,20)$	$U(100,700)$
Set4	30	20	$U(10,20)$	$U(10,20)$	$U(600,1200)$
Set5	30	20	$U(10,30)$	$U(5,15)$	$U(100,700)$
Set6	30	20	$U(10,30)$	$U(5,15)$	$U(600,1200)$
Set7	30	20	$U(10,30)$	$U(10,20)$	$U(100,700)$
Set8	30	20	$U(10,30)$	$U(10,20)$	$U(600,1200)$
Set9	30	30	$U(10,20)$	$U(5,15)$	$U(100,700)$
Set10	30	30	$U(10,20)$	$U(5,15)$	$U(600,1200)$
Set11	30	30	$U(10,20)$	$U(10,20)$	$U(100,700)$
Set12	30	30	$U(10,20)$	$U(10,20)$	$U(600,1200)$
Set13	30	30	$U(10,30)$	$U(5,15)$	$U(100,700)$
Set14	30	30	$U(10,30)$	$U(5,15)$	$U(600,1200)$
Set15	30	30	$U(10,30)$	$U(10,20)$	$U(100,700)$
Set16	30	30	$U(10,30)$	$U(10,20)$	$U(600,1200)$
Set17	40	20	$U(10,20)$	$U(5,15)$	$U(100,700)$
Set18	40	20	$U(10,20)$	$U(5,15)$	$U(600,1200)$
Set19	40	20	$U(10,20)$	$U(10,20)$	$U(100,700)$
Set20	40	20	$U(10,20)$	$U(10,20)$	$U(600,1200)$
Set21	40	20	$U(10,30)$	$U(5,15)$	$U(100,700)$
Set22	40	20	$U(10,30)$	$U(5,15)$	$U(600,1200)$
Set23	40	20	$U(10,30)$	$U(10,20)$	$U(100,700)$
Set24	40	20	$U(10,30)$	$U(10,20)$	$U(600,1200)$
Set25	40	30	$U(10,20)$	$U(5,15)$	$U(100,700)$

Table 2 Continued

Set26	40	30	$U(10{,}20)$	$U(5{,}15)$	$U(600{,}1200)$
Set27	40	30	$U(10{,}20)$	$U(10{,}20)$	$U(100{,}700)$
Set28	40	30	$U(10{,}20)$	$U(10{,}20)$	$U(600{,}1200)$
Set29	40	30	$U(10{,}30)$	$U(5{,}15)$	$U(100{,}700)$
Set30	40	30	$U(10{,}30)$	$U(5{,}15)$	$U(600{,}1200)$
Set31	40	30	$U(10{,}30)$	$U(10{,}20)$	$U(100{,}700)$
Set32	40	30	$U(10{,}30)$	$U(10{,}20)$	$U(600{,}1200)$

4.2. Computational Results

In this section, we present the computational results on each instance set for the base solution method with the conventional implementation and the CIP implementation of it with different partial solution-freezing policies. We compare the performances of both solution approaches on the objective values and discuss our findings.

In our experiments, we initialize the random α value, the coefficient of makespan in the linear combination of objectives to 0.8 and all experiments use the same random seed to change α. We start with an higher α value because minimizing total completion time may have higher importance in many cases than minimizing total tardiness, however, α can be chosen according to the importance of the objectives determined by the problem setter.

We analyze our results in two sections. In the first part, we will discuss whether we can improve the solution quality using the CIP approach although we do not increase the computation-only time. The analyses are based on different scenarios. In the second part, we discuss whether we can still achieve similar quality of a solution by decreasing the computation-only time. For this purpose, we compare the base solution method which is run for 600, 1200 or 2400 seconds with the CIP implementation of it in which we allocate 200 seconds of computation-only time.

Our CIP implementation and the base tabu search algorithm are coded in C++ programming language and experiments are done on a Intel Core i7, 2.60 GHz, 12GB RAM computer. The tabu search algorithm and two-step greedy algorithm to create initial solution of Billaut and Vilcot (2011) is recoded from the start.

4.2.1. Experiments on Allocating Same Computation Time for Both of the Methods

In this section, we present the results of the base solution method with CIP and compute-first implement-later implementation. We present the percentage improvement of total completion time and total maximum tardiness by letting both methods use the same amount of computation-only time. By the results provided in this section, we can observe how different attributes affect the solution quality of our CIP implementation, and in which instances CIP implementation outperforms the base tabu search algorithm and vice versa. In these experiments, both implementation of the tabu search algorithm is provided 200 seconds of computation-only time.

Allocated computation-only time is denoted by t_{BV} for the base solution method and t_{CIP} for the CIP implementation. In both implementations, we create an initial solution and then run the base solution method as long as the computation-only time. In the traditional approach, the base solution method stops after allocated time passes and the best solution found is implemented immediately. On the other hand, after the computation only time passes, our CIP implementation freezes a part of the best solution found, implements that frozen part on machines and continues the computation in parallel to the implementation.

For the same computation-only time, we present the percentage improvements in total completion time and the total maximum tardiness using the CIP approach in Table 16 in Appendix D. If there is no improvement with our proposed CIP implementation over the tabu search algorithm, this information is illustrated with NI in the table.

For all 32 sets, we present the average percentage improvements in objectives obtained by each solution freezing policy in Table 3 whereas the average values of percentage of improvement in total completion time and total maximum tardiness for each set is given in Table 4.

Table 3 - Average percentage improvement in total completion time and total maximum tardiness according to the each fixing policy where $t_{BV} = t_{CIP}$

$t_{BV} = t_{CIP}$ = 200 seconds	Improvement Percentage in TC_{max}	Improvement Percentage in TL_{max}
Policy1	7.35	22.49
Policy2 with θ = 0.01	7.31	21.89
Policy2 with θ = 0.02	6.62	18.29
Policy3 with θ = 0.01	7.58	23.44
Policy3 with θ = 0.02	6.84	20.03
Policy4 with q = 0.1	7.39	23.77
Policy4 with q = 0.2	5.84	19.65

Based on the results in Table 3, we see that when θ = 0.02 or q = 0.2 the policies perform worse compared to θ = 0.01 or q = 0.1; however we improve the solution quality of conventional implementation of base solution method with our CIP implementation even with the worst performing fixing policies. Excluding these policies, it can be observed that Policy 3 with θ = 0.01 and Policy 4 with q = 0.1 outperforms than Policy 1 and Policy 2 with θ = 0.01 on overage. However, the average percentage of improvements are close and Policy 1 and Policy 2 with θ = 0.01 can find the best results in some of the instances.

The main conclusion regarding to these results is that fixing more operations results less percentage of improvement in the solution quality because when we fix more operations, we narrow the neighborhood of base algorithm and we limit it to find better results. According to this fact, when we set θ = 0.02 in Policy 2 and Policy 3 or q = 0.2 in Policy 4, the number of fixed operations gets larger and we get worse results compared to the $\theta = 0.01$ in Policy 2 and Policy 3 or $q = 0.1$ in Policy 4. There is a significant difference in objectives between these policies.

Policy 1 and Policy 4 fix operations in parallel in the schedule. However, in Policy 2 and Policy 3, the operations fixed in the same iteration can be on the same machine and this may create a larger idle time for other machines. For example, if the policy fixes two operations from the same machine and no operations from other machines, all the operations assigned to other machines are shifted to the maximum completion time of that two operations at the current solution. This may worsen the total completion time of the schedule because it creates an idle time at other machines.

Even though Policy 3 may fix operations on the same machine in the same iteration, it performs better than other policies because fixing the less flexible operations limits the further computations less. It narrows the neighborhood less than other policies so that base algorithm has more opportunities to improve the solution quality.

In Policy 4, we check the remaining time of the unfixed operations to make the fixing decision. We aim to create less idle time with Policy 4 and according to the results in Table 4, Policy 4 provides considerable amount of improvement percentage in both of the objectives. Policy 4 outperforms Policy 1 and Policy 2, and provides the best total maximum tardiness on average. Policy 1 also aims to create less idle time; however, when we check the fixed number of operations in each fixing step, we observe that Policy 1 fixes more operations than Policy 4. This decreases the size of the neighborhood and consequently leaves fewer improvement opportunities as it is stated before and Policy 1 performs worse than Policy 4.

In Table 4, we can see the performance of the CIP approach with different fixing policies on different instance scenarios. From this table, we can analyze the effects of each instance attribute.

Average processing time of the operations is the first parameter we will look at. When we compare the instances with the same parameters except the processing time, it can be seen that the instances with longer average operation processing time have better results as expected. For example, Set13 and Set14 have same number of jobs and machines and same number of operations of a job distribution and available machine distribution whereas the average processing time of operations is equal to 400 seconds in Set13 and 900 seconds in Set14. The CIP approach performs better with all the fixing policies for Set14 than Set13. Moreover, the effect of average processing time on the solution quality of the CIP implementation can be observed from the average improvement percentage obtained from all the fixing policies. Table 5 shows the effect according to the processing times of operations. Average processing time of the operations affects solution quality of CIP implementation because the allocated time for the base solution algorithm is determined by the processing time of fixed operations. We need to allocate a considerable amount of time for the base algorithm since each partial-solution fixing iteration may worsen the solution quality due to the shifts. When the average processing time of operations is short, larger number of operations needs to be fixed in order to generate enough computation time for the base algorithm. However, fixing larger number of operations decreases the size of neighborhood and it causes more improvement opportunities to be eliminated. When processing times of operations are longer, the CIP implementation can allocate more time to the base algorithm with fixing less number of operations and correspondingly with less interfere to the neighborhood.

Table 4 – Average percentage improvement in total completion time and total maximum tardiness in each set where $t_{BV} = t_{CIP} = 200$

	TC_{max} (Policy1 with t_{CIP} = 200)	TL_{max} (Policy1 with t_{CIP} = 200)	TC_{max} (Policy2 with t_{CIP} = 200 & θ = 0.01)	TL_{max} (Policy2 with t_{CIP} = 200 & θ = 0.01)	TC_{max} (Policy2 with t_{CIP} = 200 & θ = 0.02)	TL_{max} (Policy2 with t_{CIP} = 200 & θ = 0.02)	TC_{max} (Policy3 with t_{CIP} = 200 & θ = 0.01)	TL_{max} (Policy3 with t_{CIP} = 200 & θ = 0.01)	TC_{max} (Policy3 with t_{CIP} = 200 & θ = 0.02)	TL_{max} (Policy3 with t_{CIP} = 200 & θ = 0.02)	TC_{max} (Policy4 with t_{CIP} = 200 & q = 0.1)	TL_{max} (Policy4 with t_{CIP} = 200 & q = 0.1)	TC_{max} (Policy4 with t_{CIP} = 200 & q = 0.02)	TL_{max} (Policy4 with t_{CIP} = 200 & q = 0.2)
Set1	10.67	28.30	10.34	31.91	11.00	25.06	10.21	24.65	10.03	28.50	10.67	27.81	10.81	32.63
Set2	12.17	49.80	9.94	43.95	8.53	37.66	10.93	45.45	9.67	38.80	12.18	52.11	8.60	33.93
Set3	7.10	19.59	7.30	22.65	7.03	23.37	6.38	21.09	6.43	16.20	7.41	22.49	5.85	18.95
Set4	12.19	44.14	12.40	39.77	8.63	26.68	13.69	45.80	9.57	29.99	12.07	45.81	8.68	32.58
Set5	6.95	17.06	8.18	20.38	7.10	15.54	7.48	19.34	6.26	13.97	6.93	18.43	6.54	19.61
Set6	10.98	42.22	13.82	51.71	13.41	46.76	13.63	52.24	12.56	46.04	13.03	48.24	8.60	33.75
Set7	3.73	8.06	4.99	5.48	4.66	6.52	4.78	5.65	4.23	5.26	5.04	10.56	4.54	9.32
Set8	7.74	32.88	8.08	35.02	9.17	37.54	9.49	39.50	7.10	28.94	8.28	36.02	5.50	25.21
Set9	7.02	24.13	6.49	17.13	6.52	14.79	5.43	24.39	6.45	23.09	7.71	30.07	6.56	21.87
Set10	12.84	54.69	10.11	32.51	7.86	22.46	13.38	49.20	11.23	34.93	12.62	44.07	10.19	39.81
Set11	4.71	9.58	3.63	9.93	2.70	5.33	4.23	11.20	3.38	13.73	4.97	14.86	3.31	15.86
Set12	12.82	36.48	11.15	31.62	12.48	29.73	11.73	39.90	13.16	31.90	10.65	38.60	10.73	38.20
Set13	9.57	27.58	8.89	22.74	7.25	19.07	7.72	22.09	6.64	19.73	6.86	26.41	6.36	18.55
Set14	11.21	49.17	10.03	47.27	10.80	39.99	11.43	50.05	10.49	41.42	11.84	50.43	9.77	49.87
Set15	4.71	5.39	3.92	5.49	4.27	6.34	4.59	2.71	3.92	6.63	4.05	5.21	3.14	1.51
Set16	10.90	40.93	10.50	39.13	9.84	34.44	10.99	37.33	10.38	35.39	10.68	41.35	8.16	39.27
Set17	4.74	10.09	5.37	11.43	3.50	7.13	5.20	12.51	3.67	7.14	6.10	17.14	4.76	12.77
Set18	7.87	22.48	9.70	31.33	7.21	21.25	9.03	26.26	8.75	25.33	9.40	28.97	5.83	19.75
Set19	5.35	7.67	4.77	8.29	3.09	4.79	4.29	6.15	4.00	5.80	5.71	10.37	6.71	11.22

Table 4 Continued

Set20	4.46	11.19	7.10	19.70	2.66	8.26	7.56	23.12	6.23	18.27	6.84	19.91	4.99	18.03
Set21	3.30	9.63	2.71	6.65	2.23	5.80	2.19	5.85	0.36	0.20	2.81	7.93	2.16	4.58
Set22	5.14	16.15	5.44	17.98	4.87	11.68	6.67	22.53	7.19	24.35	4.81	14.83	2.70	8.73
Set23	2.16	2.31	2.17	2.26	2.45	2.64	1.52	1.27	1.08	0.00	1.90	3.72	1.78	3.83
Set24	3.95	12.03	3.75	11.68	1.99	5.62	4.54	13.64	3.23	10.66	4.50	14.01	2.00	7.98
Set25	7.11	14.69	6.98	13.96	7.70	13.65	7.37	14.81	7.43	11.57	7.48	16.34	6.86	14.73
Set26	6.97	28.48	7.89	31.90	8.27	34.08	8.63	45.65	9.21	35.90	6.48	27.04	6.36	25.59
Set27	2.84	0.72	3.02	0.00	3.15	0.00	3.00	0.42	3.30	1.95	3.91	1.70	3.71	1.93
Set28	11.79	37.96	11.72	34.90	11.13	30.79	12.68	34.21	9.96	30.03	10.09	31.25	7.08	23.44
Set29	4.55	7.38	3.38	7.56	4.40	5.97	2.96	6.81	3.19	7.52	3.89	8.79	1.63	4.90
Set30	11.78	35.75	12.66	29.82	10.47	28.89	13.01	33.13	12.30	38.28	10.95	32.50	9.13	27.02
Set31	3.75	1.81	2.63	0.87	3.03	0.12	3.32	1.24	3.76	0.00	3.77	2.42	1.28	1.81
Set32	4.24	11.20	4.73	15.44	4.51	13.41	4.63	12.00	3.57	9.39	3.02	11.24	2.68	11.61

Table 5 - Average percentage improvement in total completion time and total maximum tardiness according to the processing times where $t_{BV} = t_{CIP} = 200$

	TC_{max} where processing time is $U(100{,}700)$	TL_{max} where processing time is $U(100{,}700)$	TC_{max} where processing time is $U(600{,}1200)$	TL_{max} where processing time is $U(600{,}1200)$
Policy 1	5.52	12.12	9.19	32.85
Policy 2 (θ = 0.01)	5.30	11.67	9.32	32.11
Policy 2 (θ = 0.02)	5.00	9.76	8.24	26.83
Policy 3 (θ = 0.01)	5.04	11.26	10.13	35.62
Policy 3 (θ = 0.02)	4.63	10.08	9.04	29.98
Policy 4 (q = 0.1)	5.58	14.02	9.21	33.52
Policy 4 (q = 0.2)	4.75	12.13	6.94	27.17

The size of instances is another attribute that affects the quality of both implementations. The size of instances is determined by number of jobs, number of machines and average number of operations in each job. The number of jobs is the main attribute that determines the size and it affects solution quality directly. We illustrate the results according to the number of jobs in Table 6. The improvement percentage is significantly higher for the sets having 30 jobs than sets having 40 jobs. The average percentage improvement of these sets is different due to the characteristics of CIP and tabu search algorithm. When number of jobs is equal to 40, the tabu search algorithm takes longer to find a good solution. The computation time created by fixing operations may not be used efficiently since finding an improvement move takes more time. According to the results in Table 6, Policy 3 with θ = 0.01 or Policy 4 with q = 0.1 gives better results for instances having 30 jobs whereas Policy 2 with θ = 0.01 or Policy 3 with θ = 0.01 can be suggested for the instances having 40 jobs to get better results.

Table 6 - Average percentage improvement in total completion time and total maximum tardiness related to the number of jobs where $t_{BV} = t_{CIP} = 200$

	TC_{max} where $m = 30$	TL_{max} where $m = 30$	TC_{max} where $m = 40$	TL_{max} where $m = 40$
Policy 1	9.08	30.62	5.62	14.35
Policy 2 (θ = 0.01)	8.74	28.54	5.88	15.24

Table 6 Continued

Policy 2 (θ = 0.02)	8.20	24.46	5.04	12.13
Policy 3 (θ = 0.01)	9.13	30.66	6.04	16.22
Policy 3 (θ = 0.02)	8.22	25.91	5.45	14.15
Policy 4 (q = 0.1)	9.06	32.03	5.73	15.51
Policy 4 (q = 0.2)	7.33	26.93	4.35	12.37

When the average number of operations in each job gets higher, the solution quality of CIP approach gets worse with all the policies. The performance of policies related to number of operations in each job is provided in Table 7. According to the results in Table 7, Policy 4 with q = 0.1 provides the best results for the instances having number of operations in each job uniformly distributed between 10 and 20. For the instances with number of operations in each job uniformly distributed between 10 and 30, Policy 3 with θ = 0.01 and Policy 4 with q = 0.1 can be used to get better improvement.

Table 7 - Average percentage improvement in total completion time and total maximum tardiness according to number of operations in each job where $t_{BV} = t_{CIP} = 200$

	TC_{max} where number of operations in each job is $U(10,20)$	TL_{max} where number of operations in each job is $U(10,20)$	TC_{max} where number of operations in each job is $U(10,30)$	TL_{max} where number of operations in each job is $U(10,30)$
Policy 1	8.17	25.00	6.54	19.97
Policy 2 (θ = 0.01)	8.00	23.81	6.62	19.97
Policy 2 (θ = 0.02)	6.97	19.06	6.28	17.52
Policy 3 (θ = 0.01)	8.36	26.55	6.81	20.34
Policy 3 (θ = 0.02)	7.65	22.07	6.02	17.99
Policy 4 (q = 0.1)	8.39	26.78	6.40	20.76
Policy 4 (q = 0.2)	6.94	22.58	7.38	20.53

In contrast to number of operations and average number of operations in each job, we get better results with the CIP approach when the number of machines is larger. Almost all fixing policies provide better improvement in both of the objectives when the number of machines is equal to 30; however, the difference is not significant. There is a difference between these two scenarios because when the number of machines is smaller, the idle time created by fixing may be smaller and there may be less improvement opportunities. The improvement percentage related to this attribute is shown in Table 8. Policy 3 with θ = 0.01 and Policy 4 with q = 0.1 gives better results for instances having 20 machines. On the other hand, when there are 30 machines, CIP approach with Policy 1 performs better than the other policies.

Table 8 - Average percentage improvement in total completion time and total maximum tardiness according to the number of machines where $t_{BV} = t_{CIP} = 200$

	TC_{max} where $k = 20$	TL_{max} where $k = 20$	TC_{max} where $k = 30$	TL_{max} where $k = 30$
Policy 1	6.78	20.85	7.93	24.12
Policy 2 (θ = 0.01)	7.26	22.51	7.36	21.27
Policy 2 (θ = 0.02)	6.10	17.89	7.15	18.69
Policy 3 (θ = 0.01)	7.35	22.81	7.82	24.07
Policy 3 (θ = 0.02)	6.27	18.72	7.40	21.34
Policy 4 (q = 0.1)	7.35	23.65	7.44	23.89
Policy 4 (q = 0.2)	5.63	18.30	6.06	21.00

The performance of the CIP implementation of the tabu search algorithm is also affected by the flexibility of instances. On Table 6, we see that our CIP approach performs better with all the policies when the number of operations of a job is uniformly distributed between 10 and 20 instead of 10 and 30. The effect of this parameter on the instance size is discussed before. In addition to the instance size, this attribute affects the flexibility. When there are more operations belonging to the same job, there are more operations limited by the precedence relations. So, an instance with smaller number of operations of a job is more flexible and has smaller instance size and consequently, the CIP approach performs better for these instances.

The last attribute that we discuss is the average machine cardinality of operations. This attribute affects the flexibility of an instance and the computation time to find a move which improves the objective values. The average results related to the number of machines to perform each operation obtained from our experiments are provided in Table 9. According to the results in Table 9, the CIP approach performs considerably better for the instances with smaller average machine cardinality of operations. Even though the instances with smaller machine cardinality are less flexible than those with larger average machine cardinality, the first group takes less time to find an improvement move. From the results in Table 9, the amount of time to find an improvement move by the base algorithm has a higher impact on getting better results with our CIP approach than the machine flexibility of an instance.

Table 9 - Average percentage improvement in total completion time and total maximum tardiness according to the number of machines to perform each operation where $t_{BV} = t_{CIP}$

	TC_{max} where number of machines to perform each operation is $U(5,15)$	TL_{max} where number of machines to perform each operation is $U(5,15)$	TC_{max} where number of machines to perform each operation is $U(10,20)$	TL_{max} where number of machines to perform each operation is $U(10,20)$
Policy 1	8.30	27.35	6.40	17.62
Policy 2 (θ = 0.01)	8.25	26.14	6.37	17.64
Policy 2 (θ = 0.02)	7.57	21.86	5.67	14.72
Policy 3 (θ = 0.01)	8.46	28.43	6.71	18.45
Policy 3 (θ = 0.02)	7.84	24.80	5.83	15.26
Policy 4 (q = 0.1)	8.36	28.19	6.43	19.35
Policy 4 (q = 0.2)	6.68	23.01	5.01	16.30

After analyzing the effects of each instance attribute, we want to observe the effects of our CIP approach on a multi-objective problem. According to results in Table 3, it can be seen that both objectives are affected similarly by our fixing policies. Whenever a policy performs better on total completion time than the other policy, it also provides better improvement percentage for the total maximum tardiness in general. Furthermore, we observe that both objectives are also affected similarly by the shifts due to the fixing decisions. This impact is shown in Instance36 below an illustrative example of the proposed CIP implementation subtitle. Even though the percentage improvements in the objectives are correlated, we should note that fixing policies use different rules and these rules affect different objectives. For example, shifts occurring due to Policy 2 affect total maximum tardiness less than the other policies, but we cannot get the best total maximum tardiness value with this policy. This happens because fixing some operations may limit the neighborhood more and consequently, there may be less improvement opportunities found by the base algorithm.

In the following section, we compare two implementations of the base solution method with different computation-only times to observe whether we can improve the solution quality even if we decrease the computation-only time.

4.2.2. Experiments on Decreased Computation-only Time for the CIP Implementation

The second aim that we want to reach is to shorten the computation-only time of the base algorithm without worsening the solution quality. We test three different cases by allocation different amounts of computation-only time for tabu search algorithm. In each case, we increase this time amount by doubling it. In this section, we compare the traditional implementation of tabu search algorithm and the CIP implementation of it by allowing tabu search algorithm to make computation for 600, 1200 and 2400 seconds in the conventional method and CIP implementation to spend 200 seconds of computation-only time. In these experiments, the base makespan and total tardiness values can be improved when we increase the computation-only time of base solution method, however in some cases total completion time and total maximum tardiness values may not be improved because these objectives include the computation time. If the gain in objectives is smaller than the computation time, the objectives may get worse. Therefore, we allocate different amounts of computation-only time for the traditional implementation of the base solution algorithm to see where it provides better improvements in these objectives and compare the CIP implementation with these three different cases.

In Table 10, Table 11 and Table 12 the experimental results are provided and the average percentage of improvements in objectives are illustrated based on the fixing policies. Detailed experimental results are also given in Appendix D in Table 17, Table 18 and Table 19. The percentage improvements are calculated by comparing the objective values of the conventional approach of the base solution method and its CIP implementation. Negative values show the percentage decrease in the quality of the objectives.

Table 10 - Average percentage improvement in total completion time and total maximum tardiness based on fixing policies where $t_{BV} = 600$ and $t_{CIP} = 200$

	Improvement Percentage in TC_{max}	Improvement Percentage in TL_{max}
Policy1	3.58	10.22
Policy2 with θ = 0.01	3.48	8.59
Policy2 with θ = 0.02	2.55	2.64
Policy3 with θ = 0.01	3.79	10.87
Policy3 with θ = 0.02	2.84	5.49
Policy4 with q = 0.1	3.63	11.79
Policy4 with q = 0.2	1.94	6.07

When we allow 600 seconds of computation-only time to the tabu search algorithm and 200 seconds of computation-only time to our CIP approach, we can on average improve both objectives with all policies. Even though we shorten the computation-only time by 400 seconds, there is a significant improvement in both objectives. The average percentage of improvement with each policy is shown in Table 10. By using CIP approach with fixing Policy 3 setting θ = 0.01, total completion time is improved by 3.79% and total maximum tardiness is improved by 10.87%. Additionally, according to the results in Table 17, by using Policy 3 with θ = 0.01, we either improve or maintain the same objective values in 23 instance groups and we improve one of the objectives or keep one objective value same in 8 sets although we shorten computation time by 400 seconds.

Table 11 - Average percentage improvement in total completion time and total maximum tardiness based on fixing policies where $t_{BV} = 1200$ and $t_{CIP} = 200$

	Improvement Percentage in TC_{max}	Improvement Percentage in TL_{max}
Policy1	0.89	2.09
Policy2 with θ = 0.01	0.78	0.25
Policy2 with θ = 0.02	-0.17	-6.71
Policy3 with θ = 0.01	1.12	3.10
Policy3 with θ = 0.02	0.14	-3.16
Policy4 with q = 0.1	0.94	3.66
Policy4 with q = 0.2	-0.83	-3.32

According to the experimental results in Table 11, where we shorten computation-only time by 1000 seconds, the average percentage improvement values decrease compared to the results in Table 10. This means allowing 1200 seconds of computation-only time to the base algorithm provides better solution quality for the instances that we use. In Table 11, it can be seen that we improve the objective values by CIP approach just with fixing Policy 1, Policy2 with θ = 0.01, Policy3 with θ = 0.01 and Policy4 with q = 0.1. When we compare the average percentage of improvement by using Policy3 with θ = 0.01 in Table 10 and Table 11, the improvement in total completion time decreases from 3.79% to 1.12% and this difference shows that allowing 1200 seconds to the tabu search algorithm leads to better objective values as it is stated above. Additionally, when we look at the Table 18, both objectives are improved or kept same in 16 sets and one of objectives is improved or kept same in 6 sets by CIP approach with Policy3 with θ = 0.01 whereas the computation-only time is shortened by 1000 seconds.

Table 12 - Average percentage improvement in total completion time and total maximum tardiness based on fixing policies where $t_{BV} = 2400$ and $t_{CIP} = 200$

	Improvement Percentage in TC_{max}	Improvement Percentage in TL_{max}
Policy1	0.83	6.04
Policy2 with θ = 0.01	0.74	5.10
Policy2 with θ = 0.02	-0.22	-1.55
Policy3 with θ = 0.01	1.09	7.81
Policy3 with θ = 0.02	0.09	1.67
Policy4 with q = 0.1	0.88	7.69
Policy4 with q = 0.2	-0.91	0.53

We increase the computation-only time of tabu search algorithm to 2400 seconds and we keep the computation-only time of our CIP approach as 200 seconds. We illustrate the experimental results based on policies in Table 12. When we compare the results in Table 11 with Table 12, there is a small difference between the average percentages of improvement in the objectives. This shows us that allowing 1200 seconds of computation-only time to the base algorithm provides similar objective values when we allow 2400 seconds of computation-only time. Thus, increasing the computation time does not improve the objectives in all instances. In some of the instances, even though we allow more computation time to the base algorithm, the time-sensitive objectives get worse because the improvements in the makespan and the lateness are not justified by the increase in the computation-only time. This situation happens especially in the instance groups with smaller sizes because there may be less improvement opportunities in the neighborhood after a certain amount of time. Increasing computation-time for these groups may lead to worse objective values or it provides a minor improvement. In addition to analysis on Table 12, we provide the results based on the instance groups in Table 19. According to these results, it can be seen that out of 32 sets, we can improve or keep the objective values same at 19 instances by using CIP approach with Policy3 with θ = 0.01 and we improve or keep same one of the objectives at 4 instances while the computation time is shortened by 2200 seconds.

According to both three tables, we shorten the computation time in the experiments and we observe improvement in both objectives with our CIP approach over the tabu search algorithm in four cases, which are using CIP approach with Policy1, Policy2 with θ = 0.01, Policy3 with θ = 0.01 and Policy4 with q = 0.1. The percentage improvements provided above in all four cases are significant in time-sensitive systems. We have shown that the using a CIP approach can provide benefits in different dimensions on time-sensitive FJSSP.

CHAPTER 5

CONCLUSION

In this thesis, we focus on a multi-objective flexible job shop scheduling problem in a time-sensitive environment. In time-sensitive problems, the computation time becomes more important since the main aim is to decrease the total time spent between receiving the instance and completing the implementation of the solution. As a part of the classical objectives makespan and maximum tardiness, we also consider the computation-only time as well. We redefine these two objectives for time-sensitive problems to find solutions which are more suited for the cases where there is a very limited time for computation.

Using a CIP approach, we parallelize computation time with the implementation. Using this approach, the computation time can be embedded into the implementation of the solution and consequently, we can decrease the computation-only time. In our CIP implementation, we use the tabu search algorithm of Billaut and Vilcot's (2011) study as the base algorithm to improve solution quality. We choose this tabu search algorithm since it performs well in terms of both solution quality and computation time, however CIP approach can be used on any solution method.

To perform parallelization, we proposed and tested four partial solution freezing policies that determine which part of the solution to start implementing, and continue computation. Policy 1 and Policy 4 focus on creating less idle time that occurs due to the fixing mechanism. Policy 2 focuses on fixing the operations belonging to the tardiest job with the aim of minimizing total maximum total tardiness whereas Policy 3 aims to avoid narrowing the neighborhood.

We tested our solution freezing policies through extensive computational experiments on randomly generated medium and large size instances. We compared the conventional implementation of the base tabu search with the CIP implementation with two different aims. The first aim is to shorten computation-only time of the tabu search algorithm while keeping the solution quality same and the second aim is to improve the solution quality by allocating same computation-only time to tabu search algorithm and CIP implementation. In our experiments, we observe that we reach both of our aims with the proposed CIP implementation for many of the instance sets especially by using the fixing Policy 3 with θ = 0.01 and Policy 4 with q = 0.01. We also provide information on determining which partial solution freezing policy to use depending on the instance properties.

We should emphasize that the quality of fixing policies directly affects the solution quality of our CIP implementation. We have two main findings related to the fixing policies. The first finding is about the number of fixed operations. Whenever a policy fixes a relatively large number of operations in an iteration, our CIP implementation provides smaller improvement in objective values. The second finding is that fixing the operations that has fewer moves provides better results since we narrow the neighborhood of the solution less.

To the best of our knowledge, this study is the first study in the literature to solve flexible job shop scheduling problem with the aim of minimizing the total completion time and total maximum tardiness that considers the computation-only time as well. This is also the first study to show how a CIP approach might perform in a problem where there are precedence relationships between operations. Due to precedence relationship, partial freezing rules had to create some idle time on the machines due to necessary forward shifts of successor operations. However, our CIP implementation of the base solution method still outperformed the conventional implementation on certain scenarios.

This study provides an insight to use CIP implementation for the problems having precedence constraints. Future studies can also use our CIP implementation and use the ideas of partial solution freezing policies on the problems which have similar problem structure to FJSSP. The base algorithm can be chosen depending on the problem properties. Our results provide guidance for when embedding computation time into implementation would improve makespan and maximum lateness that includes the computation time or vice versa.

Our proposed CIP implementation can be used for FJSSP with more objectives. When there are two or more objectives in a time-sensitive scenario, our CIP implementation may provide better results since the problem may require a great amount of computation time, and parallelizing the computation with the implementation may be useful.

REFERENCES

Brandimarte, Paolo. "Routing and scheduling in a flexible job shop by tabu search." Annals of Operations research 41.3 (1993): 157-183.

Brucker, Peter, and Rainer Schlie. "Job-shop scheduling with multi-purpose machines." Computing 45.4 (1990): 369-375.

Chaudhry, Imran Ali, and Abid Ali Khan. "A research survey: review of flexible job shop scheduling techniques." International Transactions in Operational Research 23.3 (2016): 551-591.

Chen, Haoxun, Jürgen Ihlow, and Carsten Lehmann. "A genetic algorithm for flexible job-shop scheduling." Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on. Vol. 2. IEEE, 1999.

Chen, James C., et al. "Flexible job shop scheduling with parallel machines using Genetic Algorithm and Grouping Genetic Algorithm." Expert Systems with Applications 39.11 (2012): 10016-10021.

Çavdar, Bahar, and Joel Sokol. "TSP Race: Minimizing completion time in time-sensitive applications." European Journal of Operational Research 244.1 (2015): 47-54.

Dauzère-Pérès, Stéphane, and Jan Paulli. "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search." Annals of Operations Research 70 (1997): 281-306.

Fattahi, Parviz, Mohammad Saidi Mehrabad, and Fariborz Jolai. "Mathematical modeling and heuristic approaches to flexible job shop scheduling problems." Journal of intelligent manufacturing 18.3 (2007): 331.

Gambardella, L. M., and M. Mastrolilli. "Effective neighborhood functions for the flexible job shop problem." Journal of scheduling 3.3 (1996): 3-20.

Gao, Jie, Linyan Sun, and Mitsuo Gen. "A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems." Computers & Operations Research 35.9 (2008): 2892-2907.

Gao, K. Z., Suganthan, P. N., Pan, Q. K., Chua, T. J., Cai, T. X., & Chong, C. S. "Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives." Journal of Intelligent Manufacturing 27.2 (2016): 363-374.

Garey, Michael R., and David S. Johnson. "A Guide to the Theory of NP-Completeness." WH Freemann, New York 70 (1979).

Glover, Fred, Manuel Laguna, and Rafael Marti. "Principles of tabu search." Approximation algorithms and metaheuristics 23 (2007): 1-12.

Ho, Nhu Binh, and Joc Cing Tay. "GENACE: An efficient cultural algorithm for solving the flexible job-shop problem." Evolutionary Computation, 2004. CEC2004. Congress on. Vol. 2. IEEE, 2004.

Hurink, Johann, Bernd Jurisch, and Monika Thole. "Tabu search for the job-shop scheduling problem with multi-purpose machines." Operations-Research-Spektrum 15.4 (1994): 205-215.

Jia, H. Z., Nee, A. Y., Fuh, J. Y., & Zhang, Y. F. "A modified genetic algorithm for distributed scheduling problems." Journal of Intelligent Manufacturing 14.3 (2003): 351-362.

Kacem, Imed, Slim Hammadi, and Pierre Borne. "Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic." Mathematics and computers in simulation 60.3 (2002): 245-276.

Li, J. Q., Pan, Q. K., Suganthan, P. N., & Chua, T. J. "A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem." The International Journal of Advanced Manufacturing Technology 52.5 (2011): 683-697.

Mönch, L., Fowler, J. W., Dauzère-Pérès, S., Mason, S. J., & Rose, O. "A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations." Journal of Scheduling 14.6 (2011): 583-599.

Na, Hongbum, and Jinwoo Park. "Multi-level job scheduling in a flexible job shop environment." International Journal of Production Research 52.13 (2014): 3877-3887.

Özgüven, Cemal, Lale Özbakır, and Yasemin Yavuz. "Mathematical models for job-shop scheduling problems with routing and process plan flexibility." Applied Mathematical Modelling 34.6 (2010): 1539-1548.

Pezzella, F., G. Morganti, and G. Ciaschetti. "A genetic algorithm for the flexible job-shop scheduling problem." Computers & Operations Research 35.10 (2008): 3202-3212.

Pitts Jr, R. A., and J. A. Ventura. "Scheduling flexible manufacturing cells using Tabu Search." International Journal of Production Research 47.24 (2009): 6907-6928.

Roshanaei, V., Ahmed Azab, and H. ElMaraghy. "Mathematical modelling and a meta-heuristic for flexible job shop scheduling." International Journal of Production Research 51.20 (2013): 6247-6274.

Scrich, Cintia Rigão, Vinícius Amaral Armentano, and Manuel Laguna. "Tardiness minimization in a flexible job shop: A tabu search approach." Journal of Intelligent Manufacturing 15.1 (2004): 103-115.

Vilcot, Geoffrey, and Jean-Charles Billaut. "A tabu search algorithm for solving a multicriteria flexible job shop scheduling problem." International Journal of Production Research 49.23 (2011): 6963-6980.

Vilcot, Geoffrey, and Jean-Charles Billaut. "A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem." European Journal of Operational Research 190.2 (2008): 398-411.

Zhang, Guohui, Liang Gao, and Yang Shi. "An effective genetic algorithm for the flexible job-shop scheduling problem." Expert Systems with Applications 38.4 (2011): 3563-3573.

APPENDIX A

Table 13 – Total completion time and total maximum tardiness of initial solutions created by two-step greedy algorithm

	TC_{max}	TL_{max}		TC_{max}	TL_{max}
Instance1001	16209	3814	Instance1081	19338	5435
Instance1002	16062	4350	Instance1082	20043	4980
Instance1003	17718	3522	Instance1083	20119	6122
Instance1004	17567	5368	Instance1084	19406	5807
Instance1005	16163	3826	Instance1085	19909	5836
Instance1006	33678	7536	Instance1086	37644	10722
Instance1007	31680	8829	Instance1087	41371	10918
Instance1008	36507	9316	Instance1088	45005	13046
Instance1009	37412	8329	Instance1089	39362	8580
Instance1010	33662	9062	Instance1090	42291	11714
Instance1011	16186	4922	Instance1091	20685	6363
Instance1012	15865	3350	Instance1092	19700	5665
Instance1013	18959	4026	Instance1093	20624	5538
Instance1014	15781	3296	Instance1094	17248	4903
Instance1015	15731	4972	Instance1095	20435	5805
Instance1016	41680	9089	Instance1096	42688	12083
Instance1017	34096	7484	Instance1097	44715	10073
Instance1018	35110	8101	Instance1098	39434	11780
Instance1019	31281	6041	Instance1099	43228	9630
Instance1020	36118	9130	Instance1100	39767	11220
Instance1021	23694	5183	Instance1101	30815	7905
Instance1022	24806	5530	Instance1102	27676	7899
Instance1023	22765	5827	Instance1103	25383	6580
Instance1024	23333	4459	Instance1104	27806	6227
Instance1025	23378	4486	Instance1105	28932	6424
Instance1026	54203	14995	Instance1106	56813	13179
Instance1027	51487	11544	Instance1107	62398	12881
Instance1028	49306	10185	Instance1108	62547	17052
Instance1029	49101	11346	Instance1109	57743	13102
Instance1030	44458	10646	Instance1110	62625	16239
Instance1031	21581	5244	Instance1111	26061	7343
Instance1032	22390	4990	Instance1112	25459	7414
Instance1033	23443	5875	Instance1113	27531	6925
Instance1034	25110	5455	Instance1114	28240	7909
Instance1035	21551	3703	Instance1115	27270	6842
Instance1036	50688	9818	Instance1116	61740	14138
Instance1037	51632	7287	Instance1117	62783	14144

Table 13 Continued

Instance1038	53987	11627	Instance1118	61279	15622
Instance1039	44597	8189	Instance1119	57837	13630
Instance1040	48312	11232	Instance1120	57317	15432
Instance1041	12113	3326	Instance1121	15097	3405
Instance1042	12817	3883	Instance1122	15542	3915
Instance1043	13379	3110	Instance1123	14460	3325
Instance1044	12331	2743	Instance1124	18516	6918
Instance1045	13332	2302	Instance1125	14946	3206
Instance1046	30683	7118	Instance1126	34134	6943
Instance1047	27074	6340	Instance1127	34358	9632
Instance1048	29143	5877	Instance1128	32907	8918
Instance1049	27862	5907	Instance1129	33596	7081
Instance1050	27106	5975	Instance1130	33414	8275
Instance1051	12596	2796	Instance1131	16315	3437
Instance1052	12092	3229	Instance1132	15311	4608
Instance1053	13919	2964	Instance1133	16507	4166
Instance1054	12926	2541	Instance1134	14578	3544
Instance1055	14505	2907	Instance1135	16444	3617
Instance1056	29869	6485	Instance1136	36539	8621
Instance1057	28420	5844	Instance1137	33134	7023
Instance1058	29723	6196	Instance1138	32541	7920
Instance1059	27390	5008	Instance1139	34661	7457
Instance1060	29786	6117	Instance1140	36535	7900
Instance1061	19276	3638	Instance1141	23403	4698
Instance1062	18412	3459	Instance1142	23056	5032
Instance1063	19854	4541	Instance1143	21351	5448
Instance1064	21186	4477	Instance1144	24167	4062
Instance1065	17383	3235	Instance1145	22335	5547
Instance1066	41757	5099	Instance1146	48717	9534
Instance1067	40812	5725	Instance1147	46220	7085
Instance1068	44595	7373	Instance1148	43859	7104
Instance1069	45564	7395	Instance1149	60480	17616
Instance1070	41636	6618	Instance1150	50788	11212
Instance1071	18093	3801	Instance1151	22032	5461
Instance1072	19207	3518	Instance1152	21244	5023
Instance1073	20440	3184	Instance1153	22987	3921
Instance1074	19194	4710	Instance1154	20736	5721
Instance1075	17435	3423	Instance1155	24579	5938
Instance1076	40532	9080	Instance1156	47129	9774
Instance1077	39367	7034	Instance1157	46067	8631
Instance1078	39845	5999	Instance1158	46353	6515
Instance1079	43691	9007	Instance1159	41885	9121
Instance1080	43989	8010	Instance1160	47747	10463

APPENDIX B

Table 14 – Properties of the Instance36

Job	Due Date	Op.	Card of Op.	Available Machines	Proc. Time
1	13693	Op1	5	M10, M14, M20, M23, M24	1198
		Op2	8	M5, M10, M11, M17, M22, M23, M28, M30	1356
		Op3	15	M1, M3, M4, M5, M7, M9, M12, M17, M20, M22, M24, M25, M26, M27, M29	652
		Op4	8	M11, M14, M16, M20, M21 M22, M26, M30	627
		Op5	15	M2, M5, M7, M8, M9, M10, M12, M15, M16, M18, M19, M21, M23, M25, M27	1047
		Op6	8	M9, M10, M15, M18, M21, M23, M26, M28	699
		Op7	14	M2, M3, M4, M5, M7, M10, M12, M14, M16, M17, M18, M20, M22, M24	1142
		Op8	5	M1, M5, M7, M8, M28	887
		Op9	14	M2, M5, M6, M9, M10, M11, M12, M14, M15, M20, M22, M24, M28, M29	768
		Op10	11	M2, M3, M7, M10, M11, M18, M22, M23, M24, M26, M27	1257
2	15054	Op11	5	M7, M20, M26, M28, M29	1058
		Op12	6	M1, M5, M8, M11, M12, M30	1149
		Op13	15	M3, M5, M6, M9, M11, M14, M15, M16, M18, M20, M21, M23, M25, M28, M29	958
		Op14	8	M2, M7, M11, M12, M14, M21, M22, M29	1394
		Op15	6	M1, M4, M7, M12, M13, M17	1394
		Op16	14	M2, M6, M7, M9, M13, M14, M15, M16, M17, M23, M24, M28, M29, M30	1244
		Op17	6	M4, M5, M8, M9, M10, M26	524
		Op18	7	M2, M9, M16, M20, M22, M29, M30	565
		Op19	6	M5, M8, M17, M27, M29, M30	1168
		Op20	9	M7, M8, M18, M21, M24, M25, M27, M28, M29	499
3	16120	Op21	12	M1, M7, M8, M9, M10, M12, M20, M23, M24, M25, M27, M29	1220
		Op22	5	M6, M10, M11, M18, M28	1318

Table 14 Continued

3	16120	Op23	9	M7, M8, M14, M15, M18, M22, M25, M26, M29	1153
		Op24	8	M1, M4, M6, M8, M10, M15, M17, M25	761
		Op25	9	M4, M15, M16, M17, M23, M24, M26, M28, M29	497
		Op26	15	M1, M2, M5, M9, M10, M14, M15, M16, M18, M20, M22, M23, M26, M29, M30	1207
		Op27	6	M1, M10, M20, M22, M28, M30	1024
		Op28	9	M1, M2, M5, M6, M9, M12, M17, M21, M23	1249
		Op29	7	M5, M8, M14, M16, M18, M20, M25	710
		Op30	13	M1, M2, M4, M11, M12, M13, M14, M17, M21, M22, M25, M28, M30	1314
4	26854	Op31	15	M3, M4, M8, M11, M12, M13, M16, M19, M22, M23, M25, M26, M27, M29, M30	485
		Op32	15	M1, M2, M5, M8, M9, M10, M11, M13, M14, M15, M19, M25, M27, M28, M29	1202
		Op33	9	M1, M3, M9, M12, M17, M20, M21, M25, M28	892
		Op34	10	M2, M5, M7, M9, M17, M18, M19, M20, M24, M25	471
		Op35	12	M1, M4, M5, M6, M8, M11, M17, M19, M21, M26, M27, M30	1086
		Op36	6	M7, M9, M10, M19, M22, M27	705
		Op37	8	M1, M4, M6, M19, M22, M23, M27, M28	612
		Op38	5	M7, M8, M15, M23, M30	875
		Op39	8	M8, M9, M10, M11, M13, M15, M19, M26	1232
		Op40	11	M2, M7, M9, M10, M13, M14, M17, M19, M24, M25, M26	844
		Op41	14	M2, M7, M8, M11, M14, M16, M17, M20, M25, M26, M27, M28, M29, M30	446
		Op42	9	M3, M6, M9, M16, M17, M23, M26, M28, M30	763
		Op43	14	M1, M5, M10, M11, M12, M13, M17, M18, M19, M22, M25, M27, M28, M29	1164
		Op44	13	M3, M9, M10, M12, M13, M14, M16, M18, M21, M22, M24, M28, M30	1051
		Op45	6	M1, M3, M8, M19, M25, M30	717

Table 14 Continued

4	26854	Op46	11	M1, M3, M8, M10, M11, M12, M13, M15, M17, M27, M29	727
		Op47	10	M2, M5, M7, M8, M9, M12, M13, M17, M25, M29	1140
		Op48	12	M3, M4, M5, M9, M10, M13, M19, M23, M25, M26, M29, M30	555
		Op49	11	M3, M5, M6, M7, M9, M10, M15, M17, M20, M23, M25	1248
		Op50	8	M1, M2, M8, M19, M20, M21, M24, M28	1243
5	26432	Op51	7	M10, M14, M15, M16, M17, M21, M22	791
		Op52	15	M1, M2, M5, M6, M7, M9, M10, M14, M15, M16, M18, M21, M23, M28, M30	923
		Op53	9	M1, M4, M6, M10, M17, M18, M24, M25, M27	603
		Op54	7	M1, M8, M9, M13, M18, M29, M30	481
		Op55	5	M7, M9, M21, M22, M26	452
		Op56	13	M7, M8, M10, M13, M14, M17, M21, M22, M24, M26, M27, M28, M29	549
		Op57	15	M1, M5, M7, M9, M10, M11, M13, M14, M15, M16, M20, M21, M22, M24, M26	868
		Op58	7	M1, M2, M4, M10, M14, M18, M25	1090
		Op59	8	M6, M7, M14, M16, M18, M19, M23, M28	1393
		Op60	13	M1, M3, M5, M12, M16, M17, M19, M20, M21, M24, M25, M27, M28	932
		Op61	6	M4, M7, M18, M21, M27, M28	1128
		Op62	11	M1, M4, M5, M7, M10, M13, M15, M20, M22, M28, M29	1034
		Op63	15	M2, M3, M5, M6, M8, M11, M14, M17, M19, M22, M24, M25, M28, M29, M30	710
		Op64	10	M3, M10, M11, M15, M16, M19, M22, M25, M28, M30	462
		Op65	5	M5, M8, M18, M21, M23	475
		Op66	9	M2, M3, M10, M11, M15, M20, M24, M26, M29	1321
		Op67	14	M6, M8, M9, M13, M15, M17, M18, M22, M23, M24, M26, M27, M28, M30	937
		Op68	9	M3, M4, M8, M12, M15, M21, M24, M25, M27	899

Table 14 Continued

5	26432	Op69	9	M1, M4, M5, M8, M9, M14, M22, M23, M29	783
		Op70	15	M9, M11, M12, M14, M15, M16, M17, M21, M22, M23, M24, M25, M26, M28, M29	1088
		Op71	15	M1, M2, M3, M6, M10, M11, M12, M16, M17, M18, M21, M23, M27, M28, M30	965
		Op72	8	M2, M3, M12, M18, M20, M22, M24, M25	911
		Op73	11	M5, M7, M11, M13, M16, M17, M21, M23, M24, M28, M29	1050
6	21930	Op74	7	M3, M7, M9, M10, M16, M28, M29	1288
		Op75	14	M3, M4, M6, M8, M9, M11, M14, M15, M17, M22, M24, M27, M28, M29	1190
		Op76	11	M1, M2, M3, M5, M6, M8, M9, M12, M16, M21, M23	1269
		Op77	14	M1, M3, M6, M7, M10, M14, M15, M16, M17, M21, M27, M28, M29, M30	467
		Op78	12	M5, M6, M10, M13, M15, M19, M21, M23, M24, M25, M27, M30	499
		Op79	10	M5, M6, M9, M11, M12, M13 M15, M18, M21, M25	1182
		Op80	14	M1, M3, M6, M7, M8, M10, M11, M12, M13, M14, M15, M17, M22, M24	836
		Op81	10	M2, M7, M10, M12, M17, M18, M19, M20, M21, M26	555
		Op82	5	M2, M5, M6, M24, M30	937
		Op83	11	M1, M2, M3, M6, M8, M12, M18, M21, M22, M24, M29	1349
		Op84	10	M2, M5, M11, M12, M13, M17, M18, M23, M26, M28	1349
		Op85	13	M3, M4, M6, M10, M11, M14, M15, M16, M17, M19, M26, M28, M29	1195
		Op86	13	M1, M6, M7, M8, M9, M11, M13, M14, M16, M21, M24, M25, M26	586
		Op87	12	M1, M2, M3, M7, M9, M20, M21, M22, M24, M25, M28, M30	1023
		Op88	9	M8, M11, M15, M17, M23, M24, M27, M28, M29	1144
		Op89	9	M7, M9, M12, M13, M15, M18, M20, M22, M27	933

Table 14 Continued

6	21930	Op90	13	M2, M3, M7, M8, M9, M11, M12, M13, M19, M20, M24, M25, M30	1120
		Op91	8	M11, M12, M16, M17, M24, M25, M29, M30	1317
		Op92	15	M4, M5, M11, M12, M14, M16, M17, M19, M18, M21, M23, M24, M25, M26, M27	701
7	11121	Op93	8	M6, M11, M12, M13, M16, M19, M20, M24	1123
		Op94	11	M4, M5, M6, M8, M9, M14, M15, M21, M22, M25, M30	444
		Op95	13	M1, M5, M6, M8, M9, M11, M15, M18, M20, M22, M26, M27, M28	916
		Op96	7	M7, M15, M23, M25, M27, M29, M30	1151
		Op97	7	M6, M7, M10, M12, M25, M28, M30	931
		Op98	14	M2, M4, M6, M9, M11, M12, M15, M16, M18, M22, M25, M26, M27, M29	581
		Op99	6	M6, M11, M25, M26, M27, M28	930
		Op100	7	M5, M7, M9, M11, M15, M16, M18	933
		Op101	12	M1, M3, M4, M7, M9, M13, M14, M16, M20, M22, M24, M28	799
		Op102	11	M3, M6, M8, M10, M11, M17, M19, M20, M27, M28, M30	1295
8	44577	Op103	14	M4, M6, M8, M9, M12, M13, M15, M16, M19, M20, M23, M24, M26, M28	718
		Op104	9	M5, M7, M8, M9, M12, M18, M21, M25, M30	1364
		Op105	11	M2, M8, M14, M16, M19, M20, M21, M23, M24, M25, M29	1217
		Op106	8	M3, M5, M6, M15, M21, M23, M27, M29	1369
		Op107	5	M4, M8, M19, M20, M26	879
		Op108	8	M3, M10, M11, M13, M20, M22, M23, M26	755
		Op109	13	M3, M6, M7, M8, M12, M13, M18, M19, M21, M23, M24, M26, M30	743
		Op110	14	M1, M2, M6, M7, M10, M17, M19, M20, M21, M24, M25, M27, M29, M30	637
		Op111	9	M3, M6, M10, M15, M16, M20, M25, M28, M29	877
		Op112	14	M2, M3, M4, M8, M9, M12, M14, M15, M16, M18, M20, M23, M24, M27	833

Table 14 Continued

8	44577	Op113	14	M1, M2, M5, M8, M9, M11, M15, M16, M20, M22, M24, M26, M29, M30	1286
		Op114	5	M12, M14, M18, M22, 29	899
		Op115	15	M3, M5, M6, M8, M9, M12, M15, M16, M19, M20, M22, M23, M25, M27, M29	622
		Op116	7	M3, M8, M10, M14, M15, M19, M26	860
		Op117	8	M1, M7, M9, M10, M15, M16, M18, M23	1368
		Op118	8	M1, M2, M5, M7, M9, M12, M20, M25	1056
		Op119	8	M3, M5, M6, M8, M10, M17, M18, M23	590
		Op120	10	M7, M9, M10, M12, M14, M15, M16, M26, M27, M30	973
		Op121	13	M2, M4, M6, M7, M8, M9, M11, M16, M19, M20, M24, M26, M30	590
		Op122	8	M3, M4, M10, M13, M14, M19, M22, M30	1147
		Op123	7	M1, M2, M6, M7, M16, M20, M27	1238
		Op124	12	M3, M4, M8, M9, M18, M19, M20, M22, M23, M24, M26, M27	817
		Op125	11	M1, M2, M3, M7, M10, M17, M18, M19, M23, M28, M30	1232
		Op126	6	M6, M16, M17, M20, M29, M30	890
		Op127	5	M3, M6, M13, M15, M17	574
		Op128	11	M1, M7, M9, M10, M15, M16, M21, M22, M24, M25, M30	553
		Op129	10	M2, M8, M9, M10, M13, M16, M20, M23, M25, M27	979
		Op130	10	M7, M11, M17, M20, M22, M24, M25, M26, M27, M30	402
		Op131	9	M4, M5, M8, M11, M20, M22, M23, M24, M26	1294
9	35863	Op132	10	M8, M9, M10, M11, M16, M17, M20, M22, M23, M26	681
		Op133	10	M1, M7, M8, M10, M16, M18, M21, M25, M29, M30	512
		Op134	14	M6, M7, M9, M10, M11, M12, M16, M18, M20, M21, M24, M25, M27, M28	1018
		Op135	15	M1, M2, M3, M5, M8, M13, M14, M15, M16, M18, M22, M23, M25, M27, M28	1204
		Op136	5	M2, M13, M18, M19, M25	1226
		Op137	9	M4, M12, M14, M19, M22, M24, M25, M26, M27	1241

Table 14 Continued

9	35863	Op138	6	M2, M6, M11, M13, M24, M27	1030
		Op139	15	M2, M3, M4, M7, M8, M10, M11, M12, M20, M21, M22, M23, M27, M28, M29	1006
		Op140	5	M15, M20, M21, M25, M28	689
		Op141	8	M1, M3, M9, M17, M18, M20, M21, M26	1047
		Op142	10	M4, M5, M6, M10, M12, M13, M16, M18, M25, M29	990
		Op143	5	M2, M4, M11, M14, M19	733
		Op144	14	M1, M2, M4, M5, M7, M8, M9, M16, M19, M21, M23, M24, M26, M27	704
		Op145	5	M3, M24, M25, M29, M30	571
		Op146	7	M1, M4, M9, M12, M20, M23, M28	1382
		Op147	7	M2, M7, M11, M12, M18, M25, M29	847
		Op148	13	M2, M3, M8, M9, M11, M12, M14, M16, M19, M20, M24, M25, M27	1371
		Op149	14	M2, M4, M5, M11, M13, M16, M17, M18, M20, M21, M22, M26, M29, M30	1260
		Op150	6	M15, M17, M23, M25, M28, M30	1352
		Op151	12	M1, M2, M4, M8, M19, M22, M24, M25, M26, M28, M29, M30	1139
		Op152	6	M3, M10, M14, M19, M20, M29	941
		Op153	14	M5, M6, M7, M8, M9, M10, M14, M16, M20, M21, M23, M25, M28, M30	809
		Op154	14	M2, M4, M5, M10, M12, M16, M18, M20, M22, M23, M24, M25, M27, M28	714
		Op155	15	M2, M3, M7, M8, M9, M14, M15, M16, M17, M20, M21, M23, M25, M28, M29	870
		Op156	5	M4, M5, M10, M12, M15	587
		Op157	14	M1, M2, M3, M4, M5, M8, M13, M17, M18, M23, M24, M25, M26, M27	882
10	10001	Op158	12	M3, M7, M9, M13, M15, M16, M17, M22, M25, M26, M27, M29	500
		Op159	7	M1, M6, M7, M12, M15, M24, M26	1311
		Op160	14	M1, M4, M9, M11, M13, M15, M17, M19, M21, M22, M23, M24, M25, M28	886
		Op161	12	M1, M4, M9, M11, M12, M13, M14, M19, M21, M23, M25, M28	873
		Op162	7	M3, M8, M10, M15, M17, M24, M26	1061
		Op163	5	M10, M11, M14, M19, M20	925
		Op164	5	M4, M11, M14, M16, M23	744
		Op165	7	M5, M14, M16, M22, M23, M29, M30	588

Table 14 Continued

10	10001	Op166	8	M6, M13, M17, M18, M21, M23, M29, M30	438
		Op167	11	M1, M8, M9, M11, M16, M18, M19, M21, M22, M25, M29	524
11	37571	Op168	10	M3, M5, M11, M12, M15, M20, M21, M24, M25, M28	1362
		Op169	11	M6, M7, M10, M18, M20, M22, M25, M27, M28, M29, M30	689
		Op170	6	M3, M9, M10, M21, M23, M25	714
		Op171	14	M1, M2, M4, M5, M9, M11, M13, M14, M18, M19, M22, M24, M25, M30	1380
		Op172	9	M1, M2, M5, M7, M8, M17, M24, M27, M28	1123
		Op173	8	M1, M3, M9, M14, M16, M20, M21, M27	858
		Op174	10	M5, M7, M9, M11, M12, M13, M14, M24, M28, M29	1214
		Op175	12	M5, M6, M7, M9, M13, M17, M18, M19, M23, M24, M27, M30	1304
		Op176	15	M2, M3, M6, M7, M9, M14, M15, M16, M17, M19, M20, M23, M27, M28, M29	428
		Op177	15	M2, M7, M12, M15, M16, M18, M19, M20, M21, M22, M24, M27, M28, M29, M30	1307
		Op178	13	M3, M4, M8, M9, M12, M13, M16, M18, M19, M23, M25, M28, M30	879
		Op179	9	M2, M9, M10, M13, M18, M21, M23, M25, M29	403
		Op180	5	M5, M11, M13, M18, M29	683
		Op181	12	M4, M7, M8, M10, M13, M17, M19, M20, M21, M23, M24, M28	910
		Op182	15	M1, M6, M7, M11, M12, M17, M18, M19, M20, M22, M23, M25, M27, M28, M29	1165
		Op183	7	M7, M8, M9, M13, M14, M18, M21	1274
		Op184	8	M2, M3, M5, M8, M22, M25, M27, M30	467
		Op185	10	M2, M3, M4, M9, M15, M20, M21, M23, M25, M30	966
		Op186	5	M6, M11, M21, M24, M25	776
		Op187	6	M1, M5, M14, M15, M19, M26	958
		Op188	7	M4, M8, M9, M12, M24, M29, M30	1226

Table 14 Continued

11	37571	Op189	10	M1, M2, M4, M5, M12, M20, M23, M25, M27, M29	931
		Op190	7	M10, M12, M21, M22, M23, M24, M26	1337
12	20651	Op191	14	M1, M5, M6, M10, M15, M16, M17, M18, M20, M22, M24, M25, M27, M29	519
		Op192	12	M6, M7, M12, M17, M19, M21, M22, M24, M26, M27, M29, M30	1047
		Op193	15	M1, M2, M3, M4, M5, M6, M8, M12, M14, M22 M24, M25, M26, M29, M30	1168
		Op194	12	M2, M5, M8, M10, M14, M17, M19, M21, M25, M28, M29, M30	943
		Op195	8	M1, M14, M20, M24, M25, M26, M28, M29	1340
		Op196	15	M2, M3, M5, M6, M7, M9, M10, M11, M15, M19, M22, M23, M25, M27, M28	1344
		Op197	10	M3, M6, M7, M8, M9, M17, M19, M21, M25, M27	1018
		Op198	8	M6, M9, M12, M15, M18, M19, M24, M29	835
		Op199	11	M6, M13, M14, M17, M18, M19, M22, M24, M25, M28, M29	902
		Op200	12	M1, M7, M10, M12, M16, M17, M19, M20, M22, M24, M25, M26	1065
		Op201	15	M1, M2, M3, M6, M10, M11, M13, M14, M16, M17, M18, M19, M23, M27, M29	749
		Op202	8	M3, M4, M5, M6, M7, M11, M26, M27	1081
		Op203	11	M1, M5, M8, M9, M11, M13, M20, M21, M23, M24, M26	1003
		Op204	13	M1, M2, M5, M6, M8, M9, M10, M15, M16, M17, M24, M27, M29	1017
		Op205	6	M3, M18, M19, M21, M23, M30	700
		Op206	14	M1, M3, M11, M12, M15, M18, M19, M20, M21, M23, M25, M28, M29, M30	746
		Op207	14	M1, M2, M4, M7, M8, M13, M15, M19, M20, M21, M22, M23, M25, M27	1268
		Op208	15	M2, M6, M10, M11, M12, M13, M14, M17, M18, M19, M25, M26, M27, M29, M30	1201
		Op209	8	M2, M3, M5, M11, M14, M21, M23, M29	545
		Op210	14	M2, M3, M4, M8, M10, M11, M13, M15, M19, M24, M25, M26, M27, M28	935

Table 14 Continued

13	22910	Op211	12	M1, M2, M8, M9, M13, M14, M19, M20, M22, M23, M24, M30	497
		Op212	11	M1, M3, M4, M5, M6, M10, M14, M15, M17, M18, M27	466
		Op213	10	M1, M6, M7, M10 M11, M13, M16, M20, M21, M29	1288
		Op214	13	M1, M2, M4, M5, M12, M15, M16, M19, M22, M25, M26, M29, M30	594
		Op215	6	M3, M7, M8, M16, M17, M20	966
		Op216	13	M1, M4, M5, M7, M8, M9, M10, M15, M16, M17, M19, M21, M22	742
		Op217	13	M1, M2, M4, M6, M11, M13, M14, M15, M18, M21, M22, M26, M30	792
		Op218	14	M1, M6, M9, M10, M11, M12, M15, M19, M21, M23, M25, M27, M28, M29	704
		Op219	12	M1, M2, M4, M6, M7, M9, M10, M16, M18, M22, M25, M30	1301
		Op220	7	M11, M14, M22, M24, M26, M29, M30	650
		Op221	12	M6, M7, M9, M11, M13, M14, M18, M20, M21, M22, M23, M25	1078
		Op222	14	M4, M8, M11, M15, M16, M19, M20, M21, M22, M23, M24, M25, M27, M30	463
		Op223	10	M1, M10, M14, M19, M21, M22, M24, M25, M26, M28	813
		Op224	8	M5, M7, M10, M12, M16, M27, M28, M30	1007
		Op225	11	M1, M4, M6, M7, M11, M13, M16, M23, M25, M26, M29	550
		Op226	12	M2, M6, M7, M10, M11, M14, M15, M17, M20, M21, M23, M25	750
		Op227	11	M6, M11, M14, M15, M17, M18, M19, M24, M25, M26, M27	629
		Op228	15	M1, M3, M4, M8, M9, M11, M13, M14, M16, M19, M22, M23, M26, M27, M30	649
		Op229	13	M1, M4, M6, M11, M13, M15, M16, M20, M21, M23, M24, M26, M29	1132
		Op230	10	M1, M5, M6, M8, M19, M20, M21, M24, M26, M30	1023
14	17208	Op231	11	M1, M5, M9, M10, M12, M17, M18, M20, M24, M27, M28	863
		Op232	5	M4, M8, M11, M12, M16	1220
		Op233	7	M1, M11, M12, M13, M16, M18, M29	570

Table 14 Continued

14	17208	Op234	6	M2, M6, M14, M16, M20, M21	1202
		Op235	9	M1, M2, M5, M8, M9, M10, M14, M24, M27	1264
		Op236	13	M3, M6, M7, M8, M10, M18, M19, M20, M21, M22, M23, M29, M30	670
		Op237	12	M2, M7, M8, M10, M19, M20, M21, M22, M25, M26, M29, M30	654
		Op238	12	M1, M4, M5, M6, M8, M12, M13, M15, M20, M23, M26, M30	1120
		Op239	10	M7, M9, M10, M11, M12, M14, M17, M25, M28, M29	680
		Op240	15	M1, M2, M3, M4, M6, M7, M9, M13, M14, M17, M19, M24, M26, M27, M30	530
		Op241	6	M5, M10, M11, M16, M23, M25	1319
		Op242	5	M5, M10, M24, M28, M29	1133
15	30822	Op243	15	M3, M8, M10, M12, M13, M14, M15, M16, M17, M21, M24, M27, M28, M29, M30	1020
		Op244	11	M6, M9, M10, M14, M18, M19, M20, M25, M27, M29, M30	1069
		Op245	5	M7, M10, M21, M23, M25	1169
		Op246	13	M1, M3, M5, M6, M8, M11, M14, M17, M19, M22, M25, M27, M28	1299
		Op247	8	M1, M4, M11, M13, M14, M15, M17, M21	1294
		Op248	12	M2, M3, M5, M13, M16, M18, M19, M20, M23, M24, M25, M29	1133
		Op249	9	M1, M8, M10, M18, M19, M20, M21, M26, M29	602
		Op250	11	M1, M2, M3, M4, M8, M9, M14, M15, M20, M22, M30	533
		Op251	15	M1, M2, M4, M5, M8, M9, M12, M13, M15, M20, M24, M26, M28, M29, M30	1089
		Op252	12	M1, M5, M7, M9, M12, M21, M22, M23, M24, M25, M27, M29	819
		Op253	10	M11, M13, M14, M15, M16, M17, M19, M23, M29, M30	1200
		Op254	10	M1, M2, M4, M7, M9, M10, M16, M17, M20, M22	684
		Op255	7	M10, M16, M19, M23, M24, M25, M28	1365
		Op256	9	M1, M7, M12, M14, M15, M19, M24, M26, M30	1176

Table 14 Continued

15	30822	Op257	15	M1, M2, M4, M8, M9, M12, M13, M15, M17, M18, M21, M22, M23, M24, M25	691
		Op258	14	M1, M2, M3, M4, M7, M, M11, M14, M16, M21, M26, M28, M29, M30	937
		Op259	13	M1, M2, M9, M11, M14, M21, M22, M24, M25, M27, M28, M29, M30	1197
		Op260	5	M4, M12, M18, M24, M30	989
		Op261	15	M1, M2, M3, M4, M5, M6, M7, M9, M10, M15, M20, M21, M22, M24, M29	767
		Op262	6	M7, M16, M18, M19, M28, M29	1252
		Op263	11	M1, M5, M10, M13, M17, M19, M24, M25, M26, M27, M29	870
		Op264	6	M4, M7, M15, M22, M27, M28	1345
16	20092	Op265	11	M2, M4, M7, M8, M11, M13, M14, M15, M19, M21, M24	441
		Op266	7	M5, M9, M12, M13, M15, M19, M24	1137
		Op267	13	M4, M5, M12, M18, M20, M21, M22, M23, M24, M26, M27, M28, M29	551
		Op268	6	M1, M3, M5, M14, M24, M26	742
		Op269	14	M1, M3, M5, M6, M7, M9, M16, M17, M18, M22, M23, M26, M28, M30	1308
		Op270	8	M2, M9, M12, M16, M19, M22, M25, M26	513
		Op271	11	M3, M6, M12, M15, M16, M17, M19, M23, M26, M28, M29	1060
		Op272	7	M3, M9, M16, M19, M23, M27, M28	979
		Op273	10	M4, M9, M10, M14, M15, M19, M22, M23, M26, M29	1277
		Op274	11	M1, M2, M3, M4, M6, M10, M14, M17, M18, M20, M30	976
		Op275	13	M1, M3, M5, M7, M9, M13, M15, M19, M21, M22, M26, M28, M30	646
		Op276	15	M2, M3, M7, M8, M9, M10, M11, M12, M13, M14, M18, M19, M24, M26, M28	1294
		Op277	10	M1, M5, M7, M9, M15, M17, M18, M22, M25, M27	677
		Op278	15	M1, M3, M5, M6, M8, M10, M15, M16, M18, M19, M20, M21, M23, M24, M29	793
		Op279	15	M5, M6, M9, M10, M11, M13, M15, M17, M22, M23, M24, M25, M26, M29, M30	1043

Table 14 Continued

16	20092	Op280	13	M2, M8, M9, M12, M13, M14, M18, M21, M25, M26, M27, M29, M30	401
		Op281	9	M3, M4, M6, M8, M15, M16, M17, M19, M26	434
		Op282	12	M1, M2, M4, M6, M9, M11, M14, M15, M18, M24, M26, M29	983
		Op283	11	M2, M5, M10, M13, M14, M15, M16, M17, M18, M25, M30	866
		Op284	6	M9, M10, M11, M18, M24, M30	1255
		Op285	10	M3, M5, M7, M11, M12, M16, M20, M23, M24, M29	902
17	21544	Op286	15	M1, M3, M4, M7, M10, M11, M13, M14, M15, M18, M19, M21, M25, M27, M30	925
		Op287	14	M5, M6, M8, M11, M12, M13, M16, M18, M24, M25, M26, M27, M28, M29	859
		Op288	10	M3, M6, M8, M10, M11, M15, M19, M27, M28, M29	913
		Op289	9	M7, M14, M16, M22, M25, M26, M27, M29, M30	738
		Op290	13	M3, M4, M9, M11, M12, M15, M22, M23, M24, M26, M27, M28, M30	1036
		Op291	9	M3, M5, M6, M11, M17, M21, M26, M28, M30	545
		Op292	13	M2, M3, M6, M7, M9, M12, M17, M18, M19, M23, M25, M26, M30	558
		Op293	11	M2, M3, M7, M8, M10, M12, M13, M15, M17, M19, M28	1213
		Op294	14	M1, M5, M6, M8, M9, M11, M13, M14, M15, M21, M25, M26, M29, M30	1035
		Op295	15	M3, M4, M6, M8, M11, M13, M15, M16, M18, M19, M20, M21, M23, M24, M26	505
		Op296	5	M1, M2, M9, M16, M19	1279
		Op297	12	M2, M5, M8, M10, M11, M12, M13, M19, M20, M22, M27, M30	1007
		Op298	6	M1, M15, M21, M25, M28, M30	984
		Op299	15	M1, M4, M7, M8, M10, M11, M14, M16, M18, M20, M23, M24, M25, M28, M30	1087
		Op300	13	M1, M2, M5, M6, M7, M9, M12, M14, M16, M17, M18, M20, M24	568

Table 14 Continued

17	21544	Op301	10	M1, M3, M4, M13, M18, M20, M22, M23, M25, M30	843
		Op302	7	M9, M11, M12, M14, M16, M23, M24	1102
		Op303	5	M7, M11, M15, M18, M21	1221
		Op304	15	M3, M4, M6, M8, M10, M15, M17, M19, M21, M23, M24, M25, M26, M28, M30	532
		Op305	11	M4, M7, M9, M16, M17, M19, M24, M25, M26, M28, M29	644
18	42143	Op306	8	M7, M11, M16, M19, M21, M24, M25, M30	814
		Op307	6	M3, M16, M20, M22, M23, M27	829
		Op308	7	M8, M11, M15, M17, M22, M26, M30	1013
		Op309	11	M3, M5, M10, M11, M12, M15, M16, M17, M25, M26, M27	1127
		Op310	11	M8, M9, M14, M17, M18, M21, M22, M23, M25, M26, M27	469
		Op311	13	M1, M2, M3, M5, M7, M8, M9, M14, M15, M16, M21, M22, M24	1187
		Op312	15	M1, M2, M5, M7, M9, M10, M14, M15, M20, M21, M23, M24, M26, M28, M30	621
		Op313	6	M5, M10, M13, M15, M22, M27	1152
		Op314	12	M2, M3, M7, M8, M9, M10, M12, M15, M16, M17, M18, M20	1049
		Op315	8	M6, M11, M22, M23, M24, M27, M28, M30	970
		Op316	7	M2, M6, M7, M9, M14, M16, M19	1004
		Op317	11	M2, M3, M4, M7, M10, M14, M18, M19, M24, M29, M30	1124
		Op318	12	M5, M8, M10, M11, M12, M13, M16, M17, M18, M19, M25, M27	652
		Op319	10	M2, M9, M12, M14, M18, M19, M22, M24, M25, M29	697
		Op320	5	M3, M15, M25, M27, M29	508
		Op321	12	M7, M8, M11, M16, M17, M18, M21, M22, M23, M25, M28, M30	762
		Op322	10	M6, M7, M9, M12, M17, M18, M19, M24, M29, M30	1327
		Op323	6	M4, M13, M16, M26, M28, M30	1219
		Op324	15	M2, M3, M7, M8, M12, M16, M17, M18, M19, M21, M22, M23, M26, M29, M30	757

Table 14 Continued

18	42143	Op325	12	M1, M4, M5, M8, M13, M17, M20, M21, M22, M24, M28, M30	1008
		Op326	5	M1, M13, M14, M21, M23	1191
		Op327	11	M1, M5, M11, M13, M16, M17, M20, M25, M27, M29, M30	1362
		Op328	10	M1, M5, M9, M10, M17, M19, M20, M22, M25, M27	1057
		Op329	9	M1, M3, M5, M6, M7, M9, M12, M15, M25	1048
		Op330	6	M4, M5, M8, M18, M21, M28	914
		Op331	15	M1, M2, M3, M10, M17, M18, M19, M20, M21, M22, M24, M27, M28, M29, M30	1035
19	37150	Op332	14	M1, M3, M4, M7, M8, M9, M13, M14, M19, M21, M24, M25, M27, M29	1397
		Op333	5	M3, M5, M16, M23, M25	406
		Op334	8	M2, M3, M5, M7, M18, M24, M26, M27	1273
		Op335	8	M2, M4, M5, M10, M16, M21, M23, M27	742
		Op336	15	M2, M6, M8, M10, M11, M12, M14, M15, M17, M19, M22, M23, M24, M27, M29	441
		Op337	11	M6, M8, M9, M11, M16, M18, M22, M23, M24, M25, M30	929
		Op338	14	M5, M8, M9, M10, M14, M15, M16, M18, M19, M20, M21, M22, M29, M30	976
		Op339	13	M2, M6, M9, M12, M13, M14, M17, M18, M19, M21, M23, M26, M30	997
		Op340	6	M1, M9, M10, M20, M26, M30	919
		Op341	5	M2, M3, M13, M21, M28	740
		Op342	5	M13, M18, M20, M26, M28	1146
		Op343	13	M1, M3, M4, M5, M6, M12, M13, M18, M19, M20, M23, M28, M30	522
		Op344	11	M3, M8, M9, M12, M15, M18, M23, M24, M26, M27, M29	1313
		Op345	6	M6, M10, M12, M19, M24, M30	720
		Op346	8	M1, M2, M6, M7, M17, M19, M26, M28	768
		Op347	7	M2, M3, M17, M18, M24, M26, M28	1212
		Op348	9	M1, M2, M3, M12, M20, M22, M24, M25, M28	1333
		Op349	9	M3, M4, M5, M8, M14, M15, M18, M19, M20	1121

Table 14 Continued

19	37150	Op350	15	M2, M3, M5, M6, M10, M11, M15, M16, M17, M19, M20, M22, M24, M26, M30	975
		Op351	14	M1, M2, M3, M4, M6, M7, M15, M16, M17, M18, M24, M26, M28, M30	828
		Op352	12	M1, M5, M6, M8, M9, M10, M15, M19, M21, M22, M23, M30	936
		Op353	14	M2, M3, M4, M6, M9, M14, M18, M21, M24, M25, M27, M28, M29, M30	1182
		Op354	9	M1, M2, M4, M14, M22, M23, M25, M26, M29	418
		Op355	11	M6, M12, M13, M15, M17, M18, M20, M26, M28, M29, M30	1220
		Op356	11	M4, M5, M7, M8, M9, M14, M16, M17, M22, M26, M29	1011
20	18305	Op357	11	M3, M5, M14, M16, M19, M22, M23, M25, M27, M28, M29	1150
		Op358	12	M1, M2, M4, M7, M9, M15, M18, M22, M23, M24, M25, M30	780
		Op359	12	M4, M5, M10, M11, M14, M15, M20, M24, M25, M26, M27, M30	802
		Op360	11	M1, M4, M8, M10, M12, M16, M17, M18, M20, M25, M30	1185
		Op361	13	M1, M5, M6, M8, M9, M12, M14, M16, M18, M20, M23, M24, M25	750
		Op362	14	M2, M3, M6, M7, M10, M14, M15, M18, M19, M21, M23, M25, M27, M28	673
		Op363	12	M3, M4, M6, M9, M15, M16, M18, M19, M20, M22, M24, M27	945
		Op364	8	M4, M7, M14, M20, M21, M22, M26, M29	770
		Op365	11	M1, M2, M8, M9, M12, M14, M16, M20, M24, M27, M28	687
		Op366	9	M1, M2, M3, M6, M7, M21, M23, M27, M28	435
		Op367	11	M1, M3, M7, M10, M14, M18, M19, M20, M21, M22, M30	444
		Op368	12	M8, M12, M13, M14, M16, M17, M19, M20, M21, M23, M26, M27	1299
		Op369	15	M2, M3, M6, M8, M9, M12, M13, M16, M17, M18, M20, M21, M22, M24, M29	1231

Table 14 Continued

20	18305	Op370	13	M1, M2, M5, M9, M13, M14, M15, M22, M24, M25, M26, M27, M28	626
		Op371	5	M10, M14, M25, M26, M27	905
		Op372	14	M1, M5, M8, M10, M11, M13, M14, M15, M16, M24, M25, M28, M29, M30	970
		Op373	11	M2, M6, M9, M10, M11, M13, M17, M18, M19, M22, M30	1396
		Op374	13	M1, M3, M5, M8, M9, M12, M14, M16, M17, M19, M21, M28, M30	1107
		Op375	14	M2, M3, M6, M9, M11, M12, M14, M16, M19, M20, M27, M28, M29, M30	1383
21	19613	Op376	12	M3, M7, M8, M9, M13, M14, M18, M19, M22, M23, M28, M29	614
		Op377	11	M4, M7, M10, M13, M15, M16, M21, M23, M28, M29, M30	752
		Op378	14	M1, M2, M3, M5, M8, M10, M11, M15, M18, M20, M26, M27, M29, M30	950
		Op379	14	M2, M3, M5, M6, M9, M10, M11, M13, M15, M18, M20, M24, M25, M26	1268
		Op380	9	M1, M5, M10, M14, M15, M16, M22, M24, M27	1181
		Op381	6	M1, M10, M11, M12, M18, M19	631
		Op382	8	M4, M6, M7, M8, M9, M13, M25, M29	1384
		Op383	11	M1, M2, M4, M5, M16, M20, M21, M23, M26, M28, M30	1309
		Op384	14	M2, M4, M6, M7, M8, M11, M12, M16, M21, M22, M23, M25, M28, M29	1292
		Op385	13	M4, M5, M6, M9, M10, M13, M14, M16, M22, M25, M27, M28, M29	1307
		Op386	14	M1, M2, M4, M5, M8, M13, M14, M15, M18, M20, M21, M25, M27, M28	1328
22	17912	Op387	12	M1, M2, M4, M6, M10, M11, M12, M15, M18, M21, M23, M26	829
		Op388	13	M2, M3, M4, M5, M6, M7, M18, M19, M23, M25, M26, M29, M30	821
		Op389	5	M3, M10, M16, M17, M25	1346
		Op390	7	M2, M4, M7, M14, M17, M19, M25	882
		Op391	11	M1, M2, M13, M15, M17, M18, M19, M22, M23, M29, M30	828
		Op392	12	M2, M5, M6, M7, M9, M14, M16, M17, M18, M19, M25, M27	752
		Op393	6	M4, M13, M16, M19, M21, M23	1324

Table 14 Continued

22	17912	Op394	11	M1, M2, M4, M7, M8, M10, M18, M20, M21, M23, M28	1074
		Op395	9	M2, M3, M11, M14, M15, M16, M19, M26, M28	1344
		Op396	10	M5, M9, M10, M11, M17, M19, M20, M23, M25, M26	803
		Op397	14	M1, M2, M3, M5, M7, M9, M11, M12, M13, M15, M23, M27, M28, M29	1036
		Op398	11	M3, M5, M8, M13, M14, M16, M18, M21, M25, M28, M30	847
		Op399	12	M2, M6, M7, M9, M10, M12, M13, M14, M25, M27, M28, M30	475
		Op400	9	M2, M6, M8, M13, M16, M18, M25, M28, M30	1005
		Op401	14	M2, M3, M5, M8, M11, M12, M18, M19, M23, M25, M26, M28, M29, M30	941
		Op402	13	M8, M10, M12, M13, M14, M16, M17, M18, M20, M22, M25, M26, M29	456
		Op403	5	M3, M13, M15, M19, M24	553
		Op404	7	M1, M4, M17, M21, M23, M25, M27	936
23	17028	Op405	11	M5, M7, M8, M12, M13, M14, M18, M20, M21, M25, M27	910
		Op406	5	M1, M5, M20, M26, M30	611
		Op407	12	M1, M2, M7, M8, M14, M15, M16, M19, M21, M23, M24, M28	1315
		Op408	14	M4, M5, M6, M7, M10, M11, M12, M16, M20, M21, M25, M27, M28, M30	602
		Op409	9	M2, M3, M4, M7, M14, M17, M18, M26, M29	768
		Op410	6	M2, M4, M7, M24, M26, M30	1334
		Op411	5	M4, M10, M20, M25, M29	915
		Op412	10	M1, M4, M5, M6, M8, M18, M20, M21, M24, M25	1290
		Op413	13	M2, M3, M5, M7, M8, M9, M10, M14, M16, M17, M18, M20, M27	930
		Op414	14	M2, M4, M5, M6, M7, M9, M10, M12, M16, M18, M19, M24, M25, M30	892
		Op415	13	M2, M4, M6, M7, M8, M13, M15, M18, M19, M24, M25, M28, M30	601
		Op416	12	M1, M2, M3, M5, M6, M8, M9, M13, M15, M21, M25, M28	899

Table 14 Continued

23	17028	Op417	15	M5, M7, M8, M9, M11, M12, M13, M15, M18, M19, M20, M21, M22, M26, M30	844
		Op418	13	M2, M3, M4, M5, M7, M9, M10, M11, M14, M16, M24, M28, M30	928
		Op419	15	M1, M3, M4, M7, M8, M11, M14, M17, M18, M19, M20, M23, M26, M29, M30	445
		Op420	9	M5, M8, M10, M15, M18, M20, M21, M24, M30	487
24	20666	Op421	9	M4, M5, M6, M10, M14, M15, M22, M27, M28	914
		Op422	6	M13, M18, M21, M22, M24, M29	1075
		Op423	7	M5, M14, M18, M20, M22, M25, M27	487
		Op424	5	M1, M7, M13, M27, M28	1357
		Op425	13	M4, M5, M6, M8, M9, M10, M13, M14, M19, M20, M22, M26, M28	831
		Op426	15	M1, M3, M5, M7, M10, M12, M13, M15, M17, M19, M23, M24, M25, M27, M30	1039
		Op427	8	M4, M9, M10, M11, M12, M13, M16, M25	876
		Op428	14	M4, M6, M7, M9, M10, M11, M13, M17, M19, M20, M21, M22, M24, M26	1274
		Op429	10	M4, M5, M13, M14, M18, M21, M23, M26, M27, M29	1160
		Op430	6	M3, M10, M12, M14, M19, M30	973
		Op431	6	M2, M4, M9, M10, M13, M17	696
		Op432	8	M2, M8, M13, M16, M17, M18, M22, M23	759
		Op433	7	M3, M6, M13, M23, M26, M27, M30	503
		Op434	5	M13, M16, M22, M25, M28	542
25	13274	Op435	6	M6, M7, M11, M14, M21, M26	1094
		Op436	7	M1, M8, M9, M11, M12, M17, M30	1191
		Op437	5	M17, M18, M22, M23, M27	977
		Op438	14	M1, M4, M6, M9, M11, M13, M15, M18, M21, M23, M25, M26, M29, M30	424
		Op439	13	M3, M4, M7, M10, M14, M15, M16, M19, M20, M21, M23, M24, M30	642
		Op440	15	M7, M8, M9, M10, M11, M13, M15, M16, M17, M21, M22, M24, M25, M27, M30	998

Table 14 Continued

25	13274	Op441	14	M4, M6, M7, M8, M9, M10, M13, M14, M16, M17, M18, M19, M23, M25	1140
		Op442	12	M1, M2, M3, M7, M8, M12, M18, M19, M23, M25, M27, M30	943
		Op443	11	M1, M3, M4, M7, M12, M13, M15, M17, M20, M26, M30	843
		Op444	8	M3, M8, M9, M12, M13, M14, M16, M27	1179
26	45608	Op445	10	M3, M4, M5, M8, M10, M14, M16, M17, M21, M29	1271
		Op446	13	M1, M2, M3, M4, M6, M9, M16, M17, M18, M20, M21, M23, M30	750
		Op447	11	M3, M4, M5, M9, M14, M15, M17, M21, M22, M29, M30	980
		Op448	7	M1, M5, M7, M17, M24, M25, M30	431
		Op449	7	M5, M10, M11, M14, M20, M24, M27	977
		Op450	10	M2, M4, M5, M11, M13, M17, M21, M27, M28, M29	1213
		Op451	10	M3, M8, M9, M12, M15, M17, M18, M19, M22, M26	819
		Op452	5	M6, M7, M19, M20, M22	1039
		Op453	5	M10, M11, M13, M20, M26	1363
		Op454	13	M1, M5, M7, M8, M10, M11, M12, M13, M14, M21, M26, M29, M30	1085
		Op455	12	M3, M5, M10, M11, M12, M15, M16, M17, M20, M21, M27, M28	1113
		Op456	7	M2, M8, M10, M14, M19, M24, M26	1342
		Op457	6	M1, M14, M24, M26, M27, M29	981
		Op458	15	M1, M5, M7, M8, M9, M10, M12, M13, M14, M15, M17, M18, M19, M20, M27	977
		Op459	15	M2, M5, M6, M7, M8, M11, M12, M13, M14, M19, M20, M23, M24, M25, M28	1377
		Op460	10	M2, M3, M4, M5, M8, M13, M14, M19, M21, M27	422
		Op461	7	M4, M6, M11, M17, M21, M22, M26	1221
		Op462	11	M3, M8, M10, M11, M17, M19, M20, M21, M23, M24, M28	525
		Op463	7	M2, M14, M18, M19, M20, M23, M26	1151
		Op464	8	M1, M2, M3, M5, M6, M14, M19, M24	875
		Op465	5	M6, M8, M12, M15, M17	743
		Op466	6	M1, M3, M7, M9, M12, M21	627

Table 14 Continued

26	45608	Op467	14	M3, M6, M10, M13, M14, M15, M16, M17, M18, M19, M20, M24, M27, M29	783
		Op468	9	M3, M7, M12, M17, M20, M21, M23, M25, M27	1163
		Op469	7	M14, M15, M21, M26, M28, M29, M30	816
		Op470	12	M4, M6, M7, M11, M13, M14, M15, M16, M19, M24, M27, M29	1220
		Op471	5	M8, M12, M13, M23, M26	629
		Op472	7	M9, M10, M15, M17, M21, M27, M30	1359
		Op473	8	M8, M11, M21, M22, M23, M25, M26, M29	1235
27	17794	Op474	8	M4, M14, M18, M20, M22, M25, M26, M30	1104
		Op475	14	M1, M3, M5, M9, M15, M16, M17, M18, M21, M23, M24, M25, M27, M30	511
		Op476	6	M2, M12, M16, M19, M24, M29	652
		Op477	12	M3, M9, M10, M11, M13, M14, M21, M25, M26, M28, M29, M30	1120
		Op478	10	M6, M7, M9, M12, M13, M16, M18, M26, M27, M29	765
		Op479	7	M3, M8, M9, M13, M24, M27, M28	1041
		Op480	7	M3, M11, M13, M21, M23, M25, M30	460
		Op481	15	M1, M4, M5, M7, M12, M13, M15, M16, M17, M19, M23, M26, M28, M29, M30	449
		Op482	13	M2, M4, M5, M7, M14, M16, M18, M19, M23, M26, M27, M28, M29	1138
		Op483	8	M2, M9, M12, M14, M17, M22, M26, M28	1111
		Op484	5	M6, M17, M21, M22, M25	884
		Op485	11	M1, M4, M9, M12, M13, M18, M19, M26, M28, M29, M30	1062
		Op486	15	M1, M2, M3, M6, M11, M12, M13, M16, M18, M21, M24, M25, M26, M27, M29	428
		Op487	5	M2, M8, M21, M22, M28	565
28	12113	Op488	5	M15, M16, M17, M20, M26	942
		Op489	12	M1, M2, M5, M7, M8, M9, M10, M12, M16, M22, M25, M28	802
		Op490	8	M1, M8, M10, M11, M13, M16, M23, M30	780

Table 14 Continued

28	12113	Op491	12	M5, M6, M8, M10, M12, M13, M14, M15, M21, M22, M23, M26	921
		Op492	5	M10, M16, M21, M28, M30	1190
		Op493	11	M1, M2, M3, M9, M11, M20, M23, M25, M26, M28, M29	814
		Op494	10	M4, M12, M14, M17, M19, M21, M22, M25, M27, M29	814
		Op495	9	M14, M15, M16, M19, M20, M23, M27, M28, M29	1389
		Op496	15	M2, M3, M6, M9, M10, M12, M15, M16, M19, M21, M25, M26, M27, M29, M30	1146
		Op497	12	M2, M5, M6, M7, M8, M14, M17, M18, M20, M23, M25, M28	815
		Op498	11	M2, M3, M6, M9, M10, M11, M14, M15, M16, M17, M19	643
		Op499	9	M2, M3, M11, M18, M19, M23, M26, M28, M29	1085
		Op500	13	M1, M4, M6, M7, M8, M9, M12, M13, M14, M19, M20, M21, M23	513
29	15202	Op501	5	M2, M4, M7, M21, M29	1321
		Op502	6	M1, M6, M10, M18, M21, M24	431
		Op503	9	M6, M8, M11, M12, M15, M18, M19, M22, M26	1105
		Op504	14	M3, M5, M7, M9, M10, M11, M16, M20, M24, M25, M26, M27, M28, M29	739
		Op505	15	M3, M9, M10, M11, M12, M13, M15, M17, M19, M20, M21, M24, M25, M26, M29	813
		Op506	10	M2, M3, M4, M6, M7, M10, M15, M16, M17, M23	1044
		Op507	13	M1, M2, M3, M7, M10, M14, M15, M17, M18, M19, M21, M23, M29	572
		Op508	8	M5, M7, M9, M11, M13, M21, M25, M28	740
		Op509	11	M2, M3, M4, M7, M9, M10, M16, M20, M21, M23, M27	612
		Op510	5	M3, M5, M6, M7, M29	840
		Op511	13	M2, M4, M5, M6, M7, M13, M14, M18, M19, M20, M22, M27, M29	1349
30	21639	Op512	10	M1, M2, M3, M5, M6, M14, M16, M21, M23, M26	598

Table 14 Continued

30	21639	Op513	9	M6, M8, M11, M12, M13, M17, M24, M25, M28	955
		Op514	13	M3, M7, M8, M9, M11, M12, M13, M16, M19, M21, M22, M23, M30	560
		Op515	14	M1, M2, M3, M5, M8, M11, M14, M15, M16, M19, M22, M26, M29, M30	1318
		Op516	14	M2, M3, M4, M7, M8, M10, M12, M14, M15, M17, M20, M22, M23, M30	850
		Op517	8	M2, M4, M9, M10, M19, M21, M27, M30	619
		Op518	5	M7, M8, M10, M13, M28	1268
		Op519	6	M19, M22, M24, M26, M28, M29	798
		Op520	11	M1, M3, M4, M5, M10, M13, M18, M19, M20, M21, M25	1264
		Op521	9	M2, M4, M7, M8, M14, M20, M23, M25, M30	1252
		Op522	13	M1, M4, M7, M8, M9, M10, M15, M16, M18, M20, M21, M22, M24	1271
		Op523	5	M6, M11, M13, M16, M28	1285
		Op524	10	M2, M3, M5, M9, M10, M13, M16, M18, M28, M29	988
		Op525	10	M1, M2, M6, M12, M14, M21, M22, M28, M29, M30	1138
31	28378	Op526	13	M3, M4, M7, M10, M14, M16, M17, M18, M21, M22, M27, M28, M29	688
		Op527	11	M3, M4, M6, M7, M9, M17, M20, M21, M23, M28, M30	1392
		Op528	14	M1, M4, M5, M9, M11, M14, M15, M21, M22, M24, M25, M26, M27, M30	1252
		Op529	5	M12, M18, M21, M26, M29	1177
		Op530	12	M3, M6, M7, M12, M17, M19, M20, M22, M23, M24, M26, M28	784
		Op531	8	M3, M5, M9, M10, M14, M17, M18, M28	693
		Op532	8	M5, M8, M10, M13, M17, M20, M23, M25	1017
		Op533	8	M1, M4, M5, M9, M10, M12, M21, M28	406
		Op534	8	M3, M10, M18, M19, M26, M27, M28, M29	1067
		Op535	14	M2, M3, M4, M6, M9, M11, M16, M17, M20, M21, M22, M27, M28, M29	915

Table 14 Continued

31	28378	Op536	11	M4, M9, M12, M20, M21, M25, M26, M27, M28, M29, M30	762
		Op537	7	M5, M9, M17, M20, M23, M24, M28	500
		Op538	12	M4, M5, M8, M10, M13, M16, M17, M21, M22, M23, M24, M29	1165
		Op539	13	M1, M4, M7, M10, M12, M13, M15, M17, M21, M23, M25, M27, M28	969
		Op540	14	M3, M4, M6, M7, M8, M9, M13, M14, M15, M18, M22, M25, M27, M28	1281
		Op541	5	M1, M5, M9, M12, M21	1354
		Op542	9	M4, M5, M8, M10, M11, M18, M24, M27, M30	1096
		Op543	8	M3, M7, M8, M13, M17, M20, M23, M30	1195
		Op544	10	M6, M7, M10, M11, M14, M15, M16, M20, M22, M27	1261
		Op545	11	M1, M2, M5, M8, M13, M15, M16, M18, M21, M26, M28	527
		Op546	15	M3, M4, M5, M6, M8, M10, M13, M14, M19, M21, M23, M24, M26, M28, M30	996
		Op547	10	M3, M9, M10, M14, M16, M22, M23, M24, M26, M30	1170
		Op548	6	M2, M6, M7, M15, M22, M24	952
		Op549	12	M2, M3, M4, M7, M8, M9, M12, M13, M20, M27, M28, M29	722
		Op550	8	M1, M3, M7, M9, M10, M17, M18, M27	954
		Op551	10	M4, M7, M10, M11, M12, M14, M22, M26, M27, M29	967
32	26298	Op552	7	M1, M2, M7, M14, M19, M22, M23	1013
		Op553	9	M3, M8, M9, M10, M11, M13, M16, M17, M27	635
		Op554	7	M1, M2, M7, M8, M24, M26, M29	1082
		Op555	15	M1, M3, M7, M9, M10, M12, M18, M19, M20, M22, M24, M25, M26, M27, M28	492
		Op556	9	M2, M4, M5, M9, M10, M13, M15, M18, M24	796
		Op557	13	M4, M6, M8, M11, M12, M13, M14, M15, M22, M23, M24, M26, M29	894
		Op558	15	M4, M6, M7, M8, M9, M12, M17, M18, M20, M21, M24, M25, M26, M27, M28	667

Table 14 Continued

32	26298	Op559	12	M1, M2, M4, M11, M12, M13, M14, M17, M19, M20, M22, M23	1386
		Op560	7	M2, M5, M15, M18, M22, M25, M27	506
		Op561	5	M4, M8, M13, M27, M30	1368
		Op562	12	M5, M6, M8, M10, M11, M14, M17, M18, M24, M25, M28, M29	696
		Op563	11	M3, M7, M15, M16, M18, M19, M20, M21, M23, M25, M26	1136
		Op564	15	M1, M2, M4, M6, M9, M10, M17, M19, M20, M21, M22, M25, M27, M29, M30	991
		Op565	7	M1, M5, M6, M12, M16, M24, M25	874
		Op566	9	M4, M6, M7, M10, M11, M13, M20, M22, M30	1209
		Op567	5	M12, M15, M17, M26, M30	478
		Op568	15	M4, M5, M7, M11, M12, M13, M14, M16, M22, M23, M25, M27, M28, M29, M30	947
		Op569	11	M1, M3, M9, M11, M13, M14, M19, M25, M27, M29, M30	969
		Op570	13	M2, M5, M6, M7, M8, M9, M15, M17, M19, M22, M25, M28, M30	1283
		Op571	13	M3, M5, M7, M8, M10, M12, M15, M16, M20, M22, M23, M24, M25	953
		Op572	6	M3, M5, M14, M15, M19, M24	1231
33	23076	Op573	9	M9, M10, M11, M13, M14, M19, M21, M23, M24	1315
		Op574	5	M2, M3, M6, M9, M11	481
		Op575	13	M3, M6, M10, M12, M15, M16, M18, M19, M20, M21, M24, M28, M30	1237
		Op576	11	M1, M4, M7, M10, M13, M18, M21, M22, M26, M28, M29	1126
		Op577	9	M1, M7, M9, M18, M19, M25, M27, M29, M30	716
		Op578	6	M4, M21, M22, M26, M27, M28	675
		Op579	11	M2, M3, M4, M5, M10, M12, M13, M17, M20, M25, M28	1084
		Op580	8	M7, M9, M16, M17, M19, M22, M24, M29	620
		Op581	13	M3, M10, M11, M12, M14, M15, M19, M20, M21, M23, M24, M25, M30	1205
		Op582	11	M1, M4, M7, M8, M12, M14, M16, M18, M19, M24, M27	492

Table 14 Continued

33	23076	Op583	15	M2, M3, M5, M8, M9, M12, M14, M17, M20, M22, M23, M26, M27, M28, M29	1066
		Op584	8	M6, M8, M10, M14, M18, M23, M25, M27	729
		Op585	13	M1, M4, M5, M10, M11, M12, M16, M17, M19, M22, M25, M26, M29	1253
		Op586	14	M1, M5, M8, M9, M10, M12, M14, M16, M18, M19, M21, M27, M29, M30	1206
		Op587	9	M2, M6, M9, M12, M14, M17, M23, M29, M30	623
		Op588	5	M4, M9, M11, M20, M23	800
		Op589	15	M1, M3, M5, M6, M9, M13, M14, M15, M18, M19, M21, M23, M25, M27, M30	446
		Op590	10	M1, M6, M7, M8, M10, M11, M18, M21, M25, M26	604
		Op591	13	M5, M6, M10, M13, M14, M18, M20, M23, M24, M25, M26, M28, M29	811
		Op592	12	M1, M2, M3, M8, M10, M12, M16, M18, M23, M28, M29, M30	1204
34	19809	Op593	9	M1, M3, M6, M13, M15, M16, M21, M23, M25	943
		Op594	9	M2, M3, M7, M12, M13, M17, M21, M22, M26	650
		Op595	9	M2, M5, M8, M16, M19, M20, M21, M24, M27	992
		Op596	7	M1, M3, M6, M15, M16, M25, M27	1212
		Op597	6	M12, M15, M16, M21, M26, M27	1243
		Op598	11	M1, M5, M6, M7, M8, M10, M20, M21, M22, M23, M28	833
		Op599	13	M2, M4, M5, M6, M8, M13, M16, M17, M21, M22, M24, M25, M27	556
		Op600	12	M5, M7, M8, M10, M11, M12, M13, M15, M16, M22, M24, M29	1296
		Op601	6	M4, M9, M10, M18, M29, M30	1214
		Op602	7	M3, M6, M7, M9, M13, M21, M22	494
		Op603	14	M1, M6, M7, M10, M12, M13, M16, M20, M21, M22, M23, M25, M29, M30	496
		Op604	10	M2, M4, M5, M7, M11, M13, M19, M25, M26, M27	851
		Op605	5	M2, M7, M14, M16, M21	642
		Op606	14	M1, M8, M12, M14, M15, M17, M18, M19, M20, M22, M25, M26, M29, M30	1271

Table 14 Continued

34	19809	Op607	15	M2, M9, M10, M12, M15, M17, M19, M23, M24, M25, M26, M27, M28, M29, M30	1230
		Op608	10	M1, M3, M5, M11, M14, M20, M23, M26, M27, M30	911
		Op609	10	M1, M2, M8, M9, M12, M19, M20, M21, M27, M30	506
		Op610	13	M6, M8, M10, M12, M13, M15, M16, M17, M21, M22, M25, M26, M30	613
35	29212	Op611	5	M6, M13, M17, M28, M29	1357
		Op612	10	M2, M4, M6, M12, M19, M22, M23, M26, M29, M30	1120
		Op613	12	M2, M3, M6, M8, M10, M14, M20, M22, M23, M25, M26, M28	1196
		Op614	13	M1, M2, M7, M8, M10, M13, M14, M18, M21, M24, M25, M27, M29	414
		Op615	8	M4, M10, M11, M18, M20, M23, M28, M30	1056
		Op616	15	M1, M4, M6, M8, M9, M10, M13, M14, M15, M16, M22, M24, M28, M29, M30	668
		Op617	6	M1, M14, M21, M25, M26, M28	597
		Op618	5	M2, M7, M17, M22, M28	1280
		Op619	6	M1, M6, M8, M18, M19, M22	985
		Op620	13	M1, M5, M7, M11, M12, M13, M14, M15, M16, M18, M20, M24, M27	1020
		Op621	7	M4, M8, M21, M23, M27, M28, M30	1076
		Op622	13	M3, M4, M8, M11, M12, M15, M18, M19, M20, M23, M25, M28, M30	664
		Op623	9	M1, M4, M7, M10, M13, M15, M17, M29, M30	1383
		Op624	5	M5, M13, M20, M23, M29	878
		Op625	12	M2, M4, M5, M6, M7, M14, M16, M18, M20, M23, M25, M29	998
		Op626	8	M1, M3, M5, M9, M12, M18, M27, M28	1284
		Op627	15	M3, M5, M7, M8, M10, M11, M12, M15, M16, M18, M19, M20, M23, M24, M29	794
		Op628	15	M1, M2, M9, M10, M13, M14, M16, M17, M18, M19, M21, M25, M28, M29, M30	1209
		Op629	5	M19, M22, M24, M28, M30	741
		Op630	8	M1, M3, M4, M9, M18, M20, M22, M26	727

Table 14 Continued

35	29212	Op631	15	M3, M4, M7, M9, M11, M12, M13, M15, M16, M18, M19, M20, M21, M25, M27	989
36	26502	Op632	8	M3, M4, M10, M13, M16, M17, M19, M20	1271
		Op633	6	M4, M22, M27, M28, M29, M30	927
		Op634	5	M3, M10, M15, M19, M30	801
		Op635	15	M4, M5, M7, M12, M13, M14, M16, M18, M20, M21, M22, M24, M25, M29, M30	573
		Op636	9	M1, M2, M11, M13, M17, M22, M23, M25, M26	1059
		Op637	10	M3, M5, M9, M10, M13, M14, M24, M26, M27, M29	939
		Op638	14	M3, M4, M8, M10, M12, M13, M15, M16, M17, M18, M22, M24, M26, M27	610
		Op639	7	M3, M12, M14, M17, M18, M23, M24	930
		Op640	13	M1, M7, M9, M12, M14, M19, M20, M21, M22, M23, M25, M28, M30	786
		Op641	14	M4, M5, M7, M8, M10, M14, M16, M18, M19, M20, M22, M23, M25, M28	1091
		Op642	9	M1, M4, M6, M11, M14, M22, M23, M24, M25	1340
		Op643	6	M7, M8, M12, M14, M28, M29	1400
		Op644	9	M4, M5, M12, M15, M18, M24, M25, M26, M27	922
		Op645	8	M2, M5, M14, M16, M22, M25, M26, M28	631
		Op646	8	M2, M3, M5, M19, M20, M22, M24, M28	503
		Op647	8	M1, M2, M11, M12, M14, M18, M20, M27	755
		Op648	14	M2, M3, M6, M7, M9, M11, M12, M13, M14, M17, M21, M25, M26, M29	813
		Op649	8	M3, M6, M11, M13, M16, M17, M18, M22	950
		Op650	6	M6, M8, M11, M19, M21, M26	1272

APPENDIX C

Table 15 – Properties of test instances

Instance	#Job	#Oper.	#Mac.	Num. of Operations Dist.	Available #Mac. Dist.	Processing Times	Set
Instance1001	30	456	20	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set1
Instance1002	30	448	20	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set1
Instance1003	30	440	20	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set1
Instance1004	30	479	20	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set1
Instance1005	30	451	20	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set1
Instance1006	30	463	20	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set2
Instance1007	30	451	20	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set2
Instance1008	30	472	20	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set2
Instance1009	30	441	20	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set2
Instance1010	30	442	20	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set2
Instance1011	30	454	20	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set3
Instance1012	30	436	20	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set3
Instance1013	30	475	20	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set3
Instance1014	30	440	20	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set3
Instance1015	30	434	20	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set3
Instance1016	30	486	20	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set4
Instance1017	30	419	20	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set4
Instance1018	30	455	20	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set4
Instance1019	30	446	20	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set4
Instance1020	30	474	20	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set4
Instance1021	30	652	20	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set5
Instance1022	30	607	20	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set5
Instance1023	30	669	20	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set5
Instance1024	30	602	20	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set5
Instance1025	30	576	20	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set5
Instance1026	30	609	20	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set6
Instance1027	30	622	20	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set6
Instance1028	30	557	20	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set6
Instance1029	30	575	20	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set6
Instance1030	30	577	20	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set6
Instance1031	30	609	20	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set7

Table 15 Continued

Instance1032	30	643	20	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set7
Instance1033	30	596	20	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set7
Instance1034	30	626	20	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set7
Instance1035	30	547	20	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set7
Instance1036	30	581	20	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set8
Instance1037	30	623	20	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set8
Instance1038	30	566	20	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set8
Instance1039	30	599	20	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set8
Instance1040	30	636	20	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set8
Instance1041	30	436	30	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set9
Instance1042	30	460	30	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set9
Instance1043	30	452	30	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set9
Instance1044	30	461	30	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set9
Instance1045	30	429	30	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set9
Instance1046	30	465	30	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set10
Instance1047	30	431	30	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set10
Instance1048	30	444	30	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set10
Instance1049	30	455	30	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set10
Instance1050	30	421	30	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set10
Instance1051	30	440	30	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set11
Instance1052	30	446	30	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set11
Instance1053	30	442	30	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set11
Instance1054	30	451	30	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set11
Instance1055	30	465	30	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set11
Instance1056	30	447	30	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set12
Instance1057	30	442	30	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set12
Instance1058	30	455	30	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set12
Instance1059	30	449	30	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set12
Instance1060	30	446	30	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set12
Instance1061	30	593	30	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set13
Instance1062	30	629	30	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set13
Instance1063	30	590	30	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set13
Instance1064	30	567	30	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set13
Instance1065	30	547	30	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set13
Instance1066	30	624	30	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set14
Instance1067	30	609	30	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set14
Instance1068	30	606	30	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set14
Instance1069	30	655	30	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set14
Instance1070	30	576	30	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set14

Table 15 Continued

Instance1071	30	583	30	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set15
Instance1072	30	618	30	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set15
Instance1073	30	553	30	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set15
Instance1074	30	640	30	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set15
Instance1075	30	585	30	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set15
Instance1076	30	627	30	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set16
Instance1077	30	598	30	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set16
Instance1078	30	565	30	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set16
Instance1079	30	608	30	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set16
Instance1080	30	640	30	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set16
Instance1081	40	605	20	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set17
Instance1082	40	604	20	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set17
Instance1083	40	600	20	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set17
Instance1084	40	606	20	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set17
Instance1085	40	598	20	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set17
Instance1086	40	597	20	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set18
Instance1087	40	589	20	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set18
Instance1088	40	621	20	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set18
Instance1089	40	570	20	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set18
Instance1090	40	618	20	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set18
Instance1091	40	588	20	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set19
Instance1092	40	594	20	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set19
Instance1093	40	620	20	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set19
Instance1094	40	555	20	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set19
Instance1095	40	631	20	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set19
Instance1096	40	611	20	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set20
Instance1097	40	597	20	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set20
Instance1098	40	584	20	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set20
Instance1099	40	587	20	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set20
Instance1100	40	578	20	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set20
Instance1101	40	864	20	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set21
Instance1102	40	855	20	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set21
Instance1103	40	785	20	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set21
Instance1104	40	833	20	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set21
Instance1105	40	802	20	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set21
Instance1106	40	762	20	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set22
Instance1107	40	847	20	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set22
Instance1108	40	831	20	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set22
Instance1109	40	769	20	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set22

Table 15 Continued

Instance1110	40	805	20	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set22
Instance1111	40	759	20	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set23
Instance1112	40	777	20	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set23
Instance1113	40	739	20	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set23
Instance1114	40	864	20	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set23
Instance1115	40	762	20	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set23
Instance1116	40	786	20	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set24
Instance1117	40	809	20	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set24
Instance1118	40	898	20	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set24
Instance1119	40	819	20	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set24
Instance1120	40	758	20	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set24
Instance1121	40	594	30	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set25
Instance1122	40	637	30	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set25
Instance1123	40	627	30	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set25
Instance1124	40	567	30	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set25
Instance1125	40	586	30	$U(10,20)$	$U(5,15)$	$U(100,700)$	Set25
Instance1126	40	586	30	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set26
Instance1127	40	618	30	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set26
Instance1128	40	631	30	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set26
Instance1129	40	606	30	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set26
Instance1130	40	585	30	$U(10,20)$	$U(5,15)$	$U(600,1200)$	Set26
Instance1131	40	610	30	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set27
Instance1132	40	621	30	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set27
Instance1133	40	611	30	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set27
Instance1134	40	602	30	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set27
Instance1135	40	615	30	$U(10,20)$	$U(10,20)$	$U(100,700)$	Set27
Instance1136	40	546	30	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set28
Instance1137	40	590	30	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set28
Instance1138	40	583	30	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set28
Instance1139	40	612	30	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set28
Instance1140	40	596	30	$U(10,20)$	$U(10,20)$	$U(600,1200)$	Set28
Instance1141	40	824	30	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set29
Instance1142	40	774	30	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set29
Instance1143	40	808	30	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set29
Instance1144	40	819	30	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set29
Instance1145	40	756	30	$U(10,30)$	$U(5,15)$	$U(100,700)$	Set29
Instance1146	40	801	30	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set30
Instance1147	40	790	30	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set30
Instance1148	40	748	30	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set30

Table 15 Continued

Instance1149	40	752	30	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set30
Instance1150	40	847	30	$U(10,30)$	$U(5,15)$	$U(600,1200)$	Set30
Instance1151	40	804	30	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set31
Instance1152	40	769	30	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set31
Instance1153	40	723	30	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set31
Instance1154	40	806	30	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set31
Instance1155	40	853	30	$U(10,30)$	$U(10,20)$	$U(100,700)$	Set31
Instance1156	40	751	30	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set32
Instance1157	40	809	30	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set32
Instance1158	40	744	30	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set32
Instance1159	40	725	30	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set32
Instance1160	40	805	30	$U(10,30)$	$U(10,20)$	$U(600,1200)$	Set32

APPENDIX D

Table 16 - Percentage improvement in total completion time and total maximum tardiness where $t_{BV} = t_{CIP} = 200$

Instance	TC_{max} (Policy1 with t_{CIP} = 200)	TL_{max} (Policy1 with t_{CIP} = 200)	TC_{max} (Policy2 with t_{CIP} = 200 & θ = 0.01)	TL_{max} (Policy2 with t_{CIP} = 200 & θ = 0.01)	TC_{max} (Policy2 with t_{CIP} = 200 & θ = 0.02)	TL_{max} (Policy2 with t_{CIP} = 200 & θ = 0.02)	TC_{max} (Policy3 with t_{CIP} = 200 & θ = 0.01)	TL_{max} (Policy3 with t_{CIP} = 200 & θ = 0.01)	TC_{max} (Policy3 with t_{CIP} = 200 & θ = 0.02)	TL_{max} (Policy3 with t_{CIP} = 200 & θ = 0.02)	TC_{max} (Policy4 with t_{CIP} = 200 & q = 0.1)	TL_{max} (Policy4 with t_{CIP} = 200 & q = 0.1)	TC_{max} (Policy4 with t_{CIP} = 200 & q = 0.02)	TL_{max} (Policy4 with t_{CIP} = 200 & q = 0.2)
Instance1001	9.12	30.33	8.73	35.13	7.71	19.26	6.05	17.32	9.01	27.25	5.67	17.17	10.1	37.03
Instance1002	8.18	23.74	7.74	27.74	8.85	21.45	7.47	18.88	5.13	17.23	9.86	28.03	8.3	28.35
Instance1003	13.17	25.73	12.1	30	14.76	26.74	13.81	27.16	10.29	28.07	12.27	30.23	8.64	32.2
Instance1004	8.89	32.65	7.33	38.03	7.71	27.23	11.57	39.53	9.89	38.32	10.03	34.15	10.42	34.98
Instance1005	13.97	29.03	15.79	28.67	15.97	30.6	12.15	20.36	15.85	31.61	15.54	29.49	16.57	30.6
Instance1006	10.6	39.68	7.49	23.6	6.94	14.38	10.92	41.99	9.6	30.52	10.35	44.37	6.1	16.86
Instance1007	9.82	48.53	7.98	39.61	7.05	35.07	7.7	36.41	8.07	27.28	10.02	47.52	6.25	32.97
Instance1008	12.02	53.75	11.11	57.97	10.63	51.78	10.33	50.9	9.47	51.11	12.46	57.83	6.45	33.72
Instance1009	14.22	58.97	12.33	50.33	11.59	52.16	14.42	57.22	11.96	54.6	14.84	62.97	10.79	40.74
Instance1010	14.19	48.07	10.8	48.22	6.42	34.93	11.3	40.71	9.23	30.47	13.21	47.86	13.4	45.37
Instance1011	8.21	28.89	9.04	33.98	8.88	32.25	8.17	26.76	6.27	15.37	9.47	32.35	5.94	22.27
Instance1012	2.28	4.51	4.56	13.06	2.84	9.57	4.27	10.71	1.84	NI	3.23	8.75	0.21	NI
Instance1013	10.46	22.41	11.7	28.1	8.58	21.43	9.76	19.72	10.03	19.06	10.24	24.12	10.44	25.44
Instance1014	4.46	13.51	4.33	17.94	3.55	18.69	3.54	23.37	3.33	12.56	5.54	12.97	5.91	20.85
Instance1015	10.1	28.62	6.86	20.15	11.3	34.9	6.17	24.87	10.67	34.03	8.55	34.26	6.73	26.21
Instance1016	12.61	48.66	14.58	51.63	8.99	41.67	17.28	49.97	12.57	53.68	8.65	38.11	8.65	38.11
Instance1017	14.28	55.6	6.26	1.65	NI	NI	12.93	43.45	12.19	27.97	15.14	54.76	12.97	41.43

Table 16 Continued

Instance1018	11.58	46.64	13.87	51.17	12.29	41.79	12.95	54	4.15	9.4	12.61	47.63	7.06	36.28
Instance1019	9.12	31.91	11.65	42.85	5.63	1.9	10.66	40.34	9.95	30.71	7.89	35.38	3.68	11.59
Instance1020	13.37	37.9	15.66	51.54	16.26	48.06	14.64	41.22	9.01	28.18	16.06	53.19	11.03	35.47
Instance1021	0.09	7.23	6.16	7.25	5.11	5.76	2.98	16.07	0.65	NI	3.17	7.55	3.17	10.81
Instance1022	9.87	24.38	9.46	15.2	10.43	27.1	10.47	22.63	8.56	21.34	6.85	20.6	7.44	19.36
Instance1023	7.57	25.78	7.49	28.31	7.34	27.01	7.08	22.78	6.45	22.23	6.42	21.76	3.55	13.54
Instance1024	5.26	15.32	4.62	29.15	1.76	2.76	3.53	19.95	4.55	17.25	5.15	18.62	6.15	30.07
Instance1025	11.97	12.58	13.18	22	10.88	15.09	13.35	15.26	11.11	9.05	13.07	23.6	12.39	24.25
Instance1026	14.15	39.93	15.9	45.58	19.1	61.62	16.01	47.72	16.84	54.12	16.67	51.96	13.47	44.97
Instance1027	9.67	30.78	13.63	51.25	12.61	43.74	13.86	44.26	12.06	39.99	14.57	47.31	8.08	26.76
Instance1028	7.83	48.58	12.8	57.89	10.24	30.05	11.08	63.85	13.61	54.31	7.92	47.62	7.32	42.9
Instance1029	10.79	44.67	14.19	58.09	13.11	56.56	14.21	55.21	11.28	50.36	13.28	52.87	6.82	28.11
Instance1030	12.46	47.14	12.58	45.76	11.97	41.84	12.99	50.14	9.02	31.44	12.7	41.46	7.33	26.01
Instance1031	1.63	8.55	1.51	NI	4.79	0.9	1.78	0.1	1.93	7.45	2.3	9.97	0.81	5.07
Instance1032	4.94	14.42	4.24	7.37	1.24	NI	4.31	6.72	3.75	3.77	4.46	11.85	4.56	10.56
Instance1033	4.3	7.46	6.82	11.88	4.56	13.72	4.67	8.98	5.21	8.27	5.69	9	5.69	9.64
Instance1034	4.24	9.86	5.54	8.13	8.01	16.17	5.58	6.18	4.36	6.8	4.78	10.44	3.98	10.14
Instance1035	3.55	NI	6.86	NI	4.69	1.81	7.56	6.26	5.9	NI	7.95	11.55	7.65	11.2
Instance1036	6.65	41.1	8.31	38.74	10.82	46.81	7.16	30.47	5.77	36.72	9.46	51.23	4.87	37.08
Instance1037	10.05	45.98	9.28	42.16	11.96	50.42	11.02	47.96	10.49	52.91	9.03	35.42	5.66	28.83
Instance1038	12.04	40.64	9.92	36.96	12.34	42.67	14.86	47.02	9.22	22.58	11.64	38.3	10.08	27.95
Instance1039	4.26	14.81	4.98	28.24	6.57	33.8	6.54	38.18	2	2.93	5.17	31.31	3.09	18.24
Instance1040	5.7	21.87	7.91	29.02	4.18	14	7.89	33.89	8	29.56	6.11	23.84	3.78	13.94
Instance1041	9.5	29.05	10.41	24.43	5.6	NI	4.52	19.27	10.42	25.16	9.21	32.7	7.18	24.18
Instance1042	9.47	37.51	6.86	25.38	6.72	20.66	6.04	25.2	6.78	29.16	9.02	35.49	7.9	28.26
Instance1043	7.88	24.87	3.27	NI	8.32	21.8	4.46	16.54	5.47	19.74	7.46	27.81	4.31	9.34

Table 16 Continued

Instance1044	2.67	15.88	5.19	24.68	4.16	13.87	4.22	27.91	1.76	9.84	3.91	30.38	5.75	27.68
Instance1045	5.57	13.34	6.74	11.15	7.81	17.61	7.93	33.02	7.82	31.55	8.93	23.95	7.65	19.9
Instance1046	14.77	57.08	12.75	57.52	4.73	25.36	10.26	59.53	9.84	52.65	13.72	53.69	NI	NI
Instance1047	11.3	56.76	7.01	19.52	3.63	16.18	12.72	54.81	13.13	44.26	8.63	30.88	10.58	49.99
Instance1048	12.74	44.09	7.88	15.3	11.69	31.81	15.59	37.21	8.47	12.36	12.93	38.54	13.15	54.34
Instance1049	12.45	52.27	8.73	32.34	4.75	5.45	17.3	55.31	14.54	38.21	13.14	44.02	12.57	41.49
Instance1050	12.96	63.23	14.2	37.87	14.51	33.52	11.05	39.16	10.15	27.16	14.67	53.21	14.67	53.21
Instance1051	4.78	12.76	4.31	7.97	4.14	4.84	4.21	12.63	0.35	NI	4.27	11.2	3.32	12.94
Instance1052	3.15	0.53	2.66	1.26	3.08	0.24	4.12	2.48	3.52	1.75	4.18	5.24	3.73	10.32
Instance1053	5.68	13.26	5.69	19.32	3.92	7.86	5.21	14.44	5.41	23.75	6.71	23.53	2.17	15.63
Instance1054	3.23	11.98	1.8	10.42	2.35	13.69	3.13	16.18	2.31	19.68	3.51	23.49	3.96	24.97
Instance1055	6.71	9.38	3.71	10.66	NI	NI	4.48	10.29	5.31	23.49	6.19	10.83	3.38	15.44
Instance1056	17.28	35.84	14.35	34.62	17.74	35.2	16.64	27.7	18.27	34.04	11.25	29.52	11.25	29.52
Instance1057	11.52	36.62	9.81	28.25	8.85	18.4	12.91	46.53	8.63	18.24	9.32	45.57	10.99	49.49
Instance1058	14.91	21.21	11.02	1.28	17.91	19.91	8.69	32.49	14.24	15.78	13.77	25.94	11.64	29.3
Instance1059	9.91	52.67	8.92	51.46	7.13	37.51	9.65	51.75	10.62	42.38	7.86	51.34	8.75	42.05
Instance1060	10.46	36.05	11.67	42.47	10.76	37.61	10.75	41.04	14.05	49.08	11.03	40.65	11.03	40.65
Instance1061	8.89	11.64	8.35	NI	6.92	8.38	7.93	10.28	1.94	NI	3.77	NI	7.71	11.32
Instance1062	9.45	26.28	6.95	25.03	4.35	2.03	6.45	28.2	8.2	23.31	6.24	24.52	4.69	14
Instance1063	10.7	32.65	9.41	20.04	8.79	25.66	10.98	27.55	8.45	33.38	6.64	38.01	6.39	18.3
Instance1064	13.16	27.46	12.4	28.69	10	28.43	11.58	35.77	13.41	30.51	9.36	22.65	6.95	13.75
Instance1065	5.65	39.87	7.36	39.96	6.17	30.86	1.68	8.65	1.21	11.46	8.3	46.85	6.04	35.39
Instance1066	9.85	55.7	9.88	51.49	10.01	29.8	9.02	53.55	8.66	28.51	11.84	41.69	10.29	53.5
Instance1067	12.05	59.56	10.03	50.01	10.08	27.32	12.59	51.9	11.59	50.01	11.52	63.17	11.04	57.14
Instance1068	10.37	41.03	11.05	50.43	12.16	46.17	12.29	49.27	11.76	44.47	11.83	50.44	10.3	51.6
Instance1069	11.43	35.36	9.38	32.2	9.55	36.3	12.03	37.09	10.95	37.14	11.33	35.06	10.02	39.42

Table 16 Continued

Instance1070	12.33	54.19	9.81	52.23	12.19	60.35	11.24	58.42	9.47	46.97	12.67	61.78	7.18	47.69
Instance1071	5.35	8.57	4.9	8.26	3.87	2.36	4.17	0.99	3.34	4.41	4.06	3.82	3.69	1.49
Instance1072	3.65	5.87	2.3	1.04	2.49	6.25	4.48	NI	3.33	4.01	2.68	7.32	3.41	0.16
Instance1073	7.65	3.97	6.71	NI	8.5	NI	8.24	NI	7.75	6.71	7.97	2.26	6.75	NI
Instance1074	2.84	1.2	1.43	9.84	2.87	15.75	2.84	4.86	2.18	9.25	1.87	5.61	0.54	NI
Instance1075	4.04	7.35	4.24	8.32	3.6	7.35	3.23	7.69	3	8.76	3.66	7.04	1.3	5.91
Instance1076	11.5	41.11	9.58	40.82	9.15	28	11	37.98	10.06	36.32	11.04	38.36	8.48	31.13
Instance1077	8.41	29.87	8.97	31.02	8.99	24.81	7.66	33.12	8.35	18.83	7.83	34.6	6.37	31.86
Instance1078	14.12	48.34	11.9	38.18	10.84	36.33	14.17	43.54	11.46	41.81	10.99	42.5	8.48	46.9
Instance1079	13.28	40.63	13.54	41.75	13.03	39.82	13.98	34.67	15.01	35.97	14.68	40.19	10.99	42.86
Instance1080	7.17	44.72	8.52	43.86	7.18	43.24	8.13	37.36	7.03	44.01	8.88	51.1	6.48	43.62
Instance1081	2.3	5.22	3.86	NI	2.14	6.94	2.84	7.69	3.88	4.26	5.14	14.31	3.26	12.49
Instance1082	6.28	8.06	6.57	11.04	6.36	13.02	5.51	11.14	6.23	11.23	7.26	16.6	7.66	18.87
Instance1083	6.99	10.18	6.47	11.42	2.69	NI	6.77	10.99	0.9	NI	7.75	17.79	6.96	14.99
Instance1084	5.34	17.36	5.13	14.53	2.85	6.11	5.2	16.6	3.46	7.6	5.93	18.34	3.31	6.62
Instance1085	2.81	9.64	4.84	20.16	3.45	9.58	5.7	16.15	3.89	12.62	4.44	18.67	2.59	10.89
Instance1086	7.33	14.08	9.62	28.69	3.32	6.04	8.15	25.45	8.01	18.86	9.09	26.44	5.52	15.98
Instance1087	7.32	19.45	8.21	25.37	6.41	19.95	8.42	17.26	7.65	22.95	9.73	27.29	6.62	23.2
Instance1088	11.85	35.94	12.04	39.84	11.69	34.61	11.54	40.19	12.66	43.03	12.33	43.75	5.63	19.34
Instance1089	6.07	22.63	8.63	32.94	6.29	21.42	10.5	31.01	9.51	26.36	5.65	18.39	6.76	25.88
Instance1090	6.76	20.28	10.02	29.82	8.36	24.24	6.56	17.4	5.94	15.44	10.19	28.97	4.64	14.35
Instance1091	8.72	14.15	8.42	12.72	6.6	6.88	8.21	11.7	8.45	13.09	8.93	13.05	11.54	15.14
Instance1092	3.4	7.7	1.59	7.83	NI	NI	NI	NI	2.04	2.63	2.91	8.56	3.35	9.11
Instance1093	8.25	1.83	4.37	NI	2.01	NI	4.88	2.54	2.46	NI	7.47	10.25	9.72	9.14
Instance1094	4.29	11.49	5.06	6.65	2.38	2.76	6.02	9.47	6.03	8.45	6.05	10.4	6.05	10.4
Instance1095	2.07	3.19	4.43	14.24	4.48	14.32	2.36	7.06	1.03	4.83	3.21	9.6	2.91	12.29

Table 16 Continued

Instance1096	3.5	NI	10.69	17.04	NI	NI	7.17	16.96	11.44	20.28	12.01	24.44	6.62	15.86
Instance1097	0.81	NI	8.11	21.97	4.95	11.46	6.91	16.95	NI	NI	5.75	15.33	5.12	15.3
Instance1098	4.56	13.4	6.59	21.39	NI	NI	9.09	28.86	6.61	20.76	8.67	29.32	6.46	24.3
Instance1099	6.97	18.48	7.14	28.36	NI	NI	7.31	25.99	7.05	27.7	4.9	18.2	3.03	21.27
Instance1100	6.44	24.08	2.96	9.73	8.36	29.86	7.3	26.82	6.03	22.62	2.85	12.26	3.74	13.43
Instance1101	3.58	7.28	1.95	2.12	1.15	NI	3.36	6.47	1.48	0.4	3.16	4.83	3.35	6
Instance1102	1.53	5.37	1.28	4.48	1.6	5.17	NI	NI	NI	NI	1.14	2.67	1.11	2.32
Instance1103	4.5	14.89	5.22	14.14	4.59	14.4	3.13	8.48	0.33	0.58	3.96	16.16	2.83	8.26
Instance1104	3.82	11.86	2.66	6.57	3.8	9.42	2.64	8.62	NI	NI	2.45	6.33	2.44	6.3
Instance1105	3.09	8.75	2.46	5.94	NI	NI	1.84	5.68	NI	NI	3.33	9.68	1.05	NI
Instance1106	1.7	3.91	6.22	22.6	1.91	5.96	4.65	18.8	6.11	24.62	1.04	2.25	3.62	13.49
Instance1107	4.99	21.16	0.32	NI	2.98	7.33	4.4	13.55	5.17	18.14	0.16	0.7	5.13	17.14
Instance1108	5.76	11.08	7.05	19.07	4.66	5.56	6.93	20.93	6.56	18.15	7.38	20.6	1.47	5.34
Instance1109	5.47	19.06	5.85	23.05	3.95	10.03	7.44	32.16	8.48	35.07	8.64	30.07	1.15	1.11
Instance1110	7.78	25.52	7.76	25.16	10.87	29.52	9.94	27.2	9.65	25.78	6.81	20.51	2.11	6.56
Instance1111	1.98	2.36	NI	NI	1	3.82	1.68	2.25	NI	NI	1.56	4.51	1.49	5.27
Instance1112	1.15	NI	2.9	3.58	3.15	2.68	1.2	NI	1.46	NI	2.7	4.14	2.83	2.99
Instance1113	6.09	3.81	6.45	5.77	5.13	5.98	2.24	4.09	3.03	NI	4.99	7.61	3.13	5.65
Instance1114	0.15	1.86	0.28	NI	NI	NI	0.19	NI	NI	NI	0.23	2.36	0.23	2.36
Instance1115	1.41	3.53	1.23	1.97	2.96	0.71	2.29	NI	0.93	NI	NI	NI	1.21	2.88
Instance1116	7.29	16.88	7.08	19.56	5.33	11.36	6.85	21.31	0.07	6.04	7.68	20.31	3.01	12.69
Instance1117	3.03	8.56	NI	NI	0.76	4.93	6.13	13.11	4.34	11.08	3.48	11.01	3.78	13.95
Instance1118	4.12	13.48	3.71	11.64	NI	NI	3.78	11.03	3.34	9.11	3.75	12.95	NI	0.87
Instance1119	0.23	5.07	3.77	14.28	3.84	11.81	1.77	10.9	2.89	11.1	1.98	8.38	3.21	12.37
Instance1120	5.06	16.17	4.21	12.94	NI	NI	4.15	11.84	5.51	15.99	5.59	17.38	NI	NI
Instance1121	5.11	8.95	4.29	2.49	7.04	3.63	4.98	NI	5.99	NI	5.92	7.22	2.98	6.08

Table 16 Continued

Instance1122	1.22	1.61	1.72	4.77	3.58	4.04	2.81	7.49	2.46	NI	1.47	7.77	1.47	7.77
Instance1123	2.88	7.3	1.74	2.75	1.55	0.65	3.01	12.15	2.65	6	2.48	12.04	2.3	6.65
Instance1124	23.24	49.27	22.55	50.4	22.38	48.2	22.14	46.25	22.94	49.06	22.85	49.1	23.67	49.28
Instance1125	3.12	6.3	4.59	9.4	3.94	11.74	3.91	8.14	3.11	2.77	4.66	5.57	3.88	3.89
Instance1126	4.51	26.52	5.69	22.13	9.12	35.69	10.91	49.53	10	47.37	6.06	29.33	6.06	29.33
Instance1127	8.52	34.81	8.49	29.97	9.83	30.21	10.16	91.51	9.66	36.01	10.29	37.94	9.49	31.95
Instance1128	6.87	26.62	7.83	28.91	7.33	41.28	5.97	18.68	7.12	30.05	2.62	12.13	2.62	12.13
Instance1129	6.11	14.7	6.45	34.39	4.26	17.43	4.57	21.6	7.45	20.36	4.59	18.29	4.59	18.29
Instance1130	8.83	39.74	10.98	44.12	10.82	45.81	11.55	46.93	11.83	45.69	8.86	37.5	9.04	36.25
Instance1131	6.74	NI	4.13	NI	6.96	NI	3.21	NI	6.02	NI	6.58	NI	7.38	NI
Instance1132	2.56	0.54	2.27	NI	1.28	NI	1.46	0.87	3.14	NI	2.68	NI	0.89	1.57
Instance1133	3.59	0.2	4.58	NI	3.57	NI	5.28	NI	5.83	NI	5.03	NI	5.35	2.55
Instance1134	0.21	2.2	NI	NI	NI	NI	1.65	NI	NI	NI	1.58	3.76	1.05	1.73
Instance1135	1.09	0.66	4.1	NI	3.96	NI	3.38	1.21	1.51	9.76	3.7	4.73	3.9	3.8
Instance1136	15.27	38.29	17.65	42.13	18.04	38.41	17.55	36.44	13.86	30.38	15.25	40.82	7.7	23.11
Instance1137	9.23	42.7	4.79	21.9	8.51	38.84	9.93	44.96	7.44	46.39	9.68	41.93	3.84	22.59
Instance1138	12.95	46.51	13.64	41.74	11.91	46.73	12.08	44.36	11.17	34.46	10.2	31.93	11.57	32.15
Instance1139	6.88	25.67	9.31	33.38	0.73	NI	9.46	22.38	3.38	5.64	5.25	21.12	5.25	21.12
Instance1140	14.64	36.64	13.21	35.34	16.47	29.95	14.39	22.89	13.96	33.27	10.09	20.44	7.04	18.25
Instance1141	3.21	7.91	1.56	5.56	2.6	3.15	0.95	4.37	2.09	5.43	2.93	7.63	1.39	5.13
Instance1142	5.17	2.67	4.51	8.78	6.09	6.78	3.4	2.16	4.74	11.72	6.03	7.4	3.39	5.12
Instance1143	2.86	18.04	2.06	10.52	2.09	8.53	1.63	14.35	1.07	9.53	1.56	14.85	0.9	9.31
Instance1144	7.32	NI	6.5	NI	6.98	NI	6.06	NI	4.72	NI	5.37	NI	0.48	NI
Instance1145	4.18	8.28	2.29	12.96	4.26	11.37	2.74	13.18	3.32	10.91	3.55	14.06	1.99	4.93
Instance1146	7.49	32.29	9.23	37.87	8.68	39.16	11.03	32.93	8.8	39.59	5.68	28.51	6.61	28.82
Instance1147	12.81	29.08	13.58	11.88	10.17	19.01	10.11	18.62	11.11	26.58	9.83	26.21	7.85	17.81

Table 16 Continued

Instance1148	6.08	21.37	7.22	17.84	NI	NI	6.98	29.04	7.24	36.33	6.38	27.35	4.46	10.47
Instance1149	22.87	67.59	24.03	60.95	23.68	59.05	26.99	69.22	24.96	68.88	25.02	67.06	23.1	67.32
Instance1150	9.66	28.44	9.25	20.54	9.8	27.22	9.94	15.82	9.37	20.01	7.85	13.36	3.62	10.66
Instance1151	1.69	NI	1.91	NI	2.05	NI	2.42	NI	3.14	NI	2.11	NI	1.94	NI
Instance1152	2.85	NI	2.96	NI	2.01	NI	2.58	NI	2.17	NI	2.54	2.18	0.87	0.33
Instance1153	10.6	NI	6.05	NI	10.23	NI	10.26	NI	12	NI	10.44	NI	0.53	4.46
Instance1154	1.91	9.07	0.78	4.35	NI	NI	0.31	5.37	NI	NI	1.61	7.85	1.14	3.18
Instance1155	1.72	NI	1.44	NI	0.86	0.58	1.01	0.84	1.49	NI	2.15	2.08	1.92	1.08
Instance1156	8.12	21.6	6.53	24.75	8.25	16.26	6.48	10.45	6.65	18.5	3.01	13.15	3.01	13.15
Instance1157	4.03	4.23	3.64	5.47	3.21	NI	2.98	NI	4.86	6.34	1.17	3.55	2.47	5.39
Instance1158	NI	NI	4.34	15.81	4.34	15.81	3.35	13.35	NI	NI	1.33	7.41	1.33	7.41
Instance1159	3.47	9.3	4.9	12.84	4.6	16.08	5.06	9.62	1.04	NI	4.04	13.43	2.91	14
Instance1160	5.59	20.87	4.26	18.33	2.15	18.92	5.27	26.56	5.28	22.11	5.53	18.65	3.68	18.08

Table 17 - Average percentage improvement in total completion time and total maximum tardiness in each set where $t_{BV} = 600$ and $t_{CIP} = 200$

	TC_{max} (Policy1 with t_{CIP} = 200)	TL_{max} (Policy1 with t_{CIP} = 200)	TC_{max} (Policy2 with t_{CIP} = 200 & θ = 0.01)	TL_{max} (Policy2 with t_{CIP} = 200 & θ = 0.01)	TC_{max} (Policy2 with t_{CIP} = 200 & θ = 0.02)	TL_{max} (Policy2 with t_{CIP} = 200 & θ = 0.02)	TC_{max} (Policy3 with t_{CIP} = 200 & θ = 0.01)	TL_{max} (Policy3 with t_{CIP} = 200 & θ = 0.01)	TC_{max} (Policy3 with t_{CIP} = 200 & θ = 0.02)	TL_{max} (Policy3 with t_{CIP} = 200 & θ = 0.02)	TC_{max} (Policy4 with t_{CIP} = 200 & q = 0.1)	TL_{max} (Policy4 with t_{CIP} = 200 & q = 0.1)	TC_{max} (Policy4 with t_{CIP} = 200 & q = 0.02)	TL_{max} (Policy4 with t_{CIP} = 200 & q = 0.2)
Set1	3.16	9.60	2.84	14.21	3.57	5.76	2.66	5.66	2.50	10.10	3.22	9.52	3.34	15.02
Set2	5.90	25.72	3.51	17.65	1.98	7.72	4.55	18.64	3.21	8.53	5.89	28.76	2.07	2.97
Set3	2.21	3.89	2.40	7.04	2.13	8.61	1.43	5.46	1.51	-2.15	2.51	7.52	0.87	1.84
Set4	4.62	19.89	4.80	10.86	0.58	-18.22	6.23	21.50	1.78	-1.38	4.51	22.04	0.83	3.14
Set5	2.49	1.42	3.74	5.40	2.61	-0.48	3.04	4.03	1.75	-4.43	2.44	2.98	2.03	4.44
Set6	5.54	24.12	8.54	37.07	8.11	29.48	8.35	37.68	7.20	29.53	7.72	32.12	3.01	13.08
Set7	1.19	-1.95	2.49	-4.69	2.11	-4.32	2.26	-3.54	1.70	-4.08	2.53	1.94	2.02	0.56
Set8	4.61	21.21	4.95	23.39	6.09	26.79	6.42	28.37	3.93	16.38	5.17	24.88	2.28	12.04
Set9	1.71	3.53	1.15	-5.26	1.17	-8.08	0.02	4.43	1.13	2.28	2.44	11.42	1.21	1.21
Set10	6.37	27.58	3.42	-7.93	1.02	-24.27	7.00	18.82	4.66	-3.63	6.13	10.15	1.30	-18.86
Set11	2.21	-3.50	1.11	-2.95	-0.68	-13.54	1.72	-1.53	0.86	0.91	2.48	2.95	0.78	3.93
Set12	7.53	19.75	5.78	13.39	7.16	11.52	6.40	24.25	7.91	14.08	5.23	22.56	5.33	22.37
Set13	4.07	6.79	3.36	1.04	1.60	-4.65	2.09	-3.46	0.97	-7.96	1.20	2.46	0.65	-4.79
Set14	7.16	34.58	5.93	32.24	6.74	23.33	7.40	35.85	6.41	25.09	7.82	36.98	5.66	35.38
Set15	2.88	0.00	2.08	-0.68	2.43	0.66	2.76	-3.82	2.08	1.38	2.21	-0.15	1.28	-4.70
Set16	7.52	29.66	7.11	27.23	6.41	21.83	7.62	25.23	6.99	23.10	7.30	30.11	4.67	27.72
Set17	0.37	0.90	1.02	2.19	-0.94	-3.09	0.84	3.59	-0.78	-2.68	1.79	8.67	0.39	3.85
Set18	3.25	7.31	5.18	17.91	2.56	5.85	4.48	11.95	4.19	10.77	4.85	14.91	1.11	3.99
Set19	1.83	2.51	1.21	2.48	-1.90	-7.97	-0.19	-3.10	0.42	-1.37	2.20	5.26	3.27	6.13
Set20	0.86	-2.07	3.59	9.24	-3.26	-16.75	4.06	13.32	2.42	6.24	3.32	9.58	1.40	7.47

Table 17 Continued

Set21	1.19	5.11	0.58	2.00	-0.24	-2.34	-0.14	0.06	-5.14	-19.87	0.68	3.36	0.01	-0.20
Set22	2.14	9.26	2.45	11.11	1.87	4.39	3.72	16.20	4.25	18.16	1.80	7.94	-0.40	1.20
Set23	1.32	-0.55	0.85	-1.72	1.23	-2.37	0.67	-4.88	-0.84	-8.57	1.06	0.96	0.93	1.06
Set24	2.68	5.31	1.87	2.61	-1.06	-9.01	3.27	7.14	1.95	3.66	3.23	7.49	0.69	1.01
Set25	4.05	9.59	3.91	8.81	4.64	8.45	4.30	8.60	4.37	4.71	4.42	11.31	3.79	9.65
Set26	3.98	16.54	4.94	20.26	5.34	23.21	5.73	36.68	6.31	25.50	3.50	14.89	3.37	13.22
Set27	0.70	-3.17	0.16	-10.55	-0.36	-12.90	0.84	-6.99	0.91	-5.34	1.79	-2.86	1.59	-1.33
Set28	7.11	25.92	7.07	22.11	6.43	15.87	8.05	21.43	5.19	16.59	5.31	17.79	2.14	8.26
Set29	2.75	2.12	1.57	2.77	2.61	0.10	1.13	1.10	1.37	3.46	2.09	3.56	-0.22	1.21
Set30	8.22	29.96	9.13	23.57	6.80	19.32	9.48	27.14	8.74	32.67	7.36	26.33	5.46	20.65
Set31	2.48	-4.80	1.33	-13.92	1.63	-6.01	2.03	-5.15	1.55	-11.03	2.49	-1.57	-0.05	-1.17
Set32	2.48	0.64	3.31	8.04	3.08	5.62	3.21	3.13	1.50	-4.95	1.57	3.44	1.22	3.82

Table 18 - Average percentage improvement in total completion time and total maximum tardiness in each set where $t_{BV} = 1200$ and $t_{CIP} = 200$

	TC_{max} (Policy1 with t_{CIP} = 200)	TL_{max} (Policy1 with t_{CIP} = 200)	TC_{max} (Policy2 with t_{CIP} = 200 & θ = 0.01)	TL_{max} (Policy2 with t_{CIP} = 200 & θ = 0.01)	TC_{max} (Policy2 with t_{CIP} = 200 & θ = 0.02)	TL_{max} (Policy2 with t_{CIP} = 200 & θ = 0.02)	TC_{max} (Policy3 with t_{CIP} = 200 & θ = 0.01)	TL_{max} (Policy3 with t_{CIP} = 200 & θ = 0.01)	TC_{max} (Policy3 with t_{CIP} = 200 & θ = 0.02)	TL_{max} (Policy3 with t_{CIP} = 200 & θ = 0.02)	TC_{max} (Policy4 with t_{CIP} = 200 & q = 0.1)	TL_{max} (Policy4 with t_{CIP} = 200 & q = 0.1)	TC_{max} (Policy4 with t_{CIP} = 200 & q = 0.02)	TL_{max} (Policy4 with t_{CIP} = 200 & q = 0.2)
Set1	-0.75	-0.24	-1.09	4.78	-0.34	-4.76	-1.26	-5.02	-1.42	0.29	-0.70	-0.76	-0.56	5.62
Set2	1.50	9.45	-1.00	1.03	-2.60	-11.23	0.10	1.35	-1.32	-10.08	1.50	13.56	-2.51	-18.81
Set3	0.40	-2.70	0.58	0.17	0.33	2.34	-0.41	-1.58	-0.31	-8.71	0.70	1.34	-0.97	-4.89
Set4	-0.68	-5.13	-0.51	-16.38	-4.95	-53.89	1.00	-2.97	-3.71	-32.48	-0.77	-2.55	-4.67	-26.94
Set5	-0.89	-3.28	0.40	0.87	-0.77	-5.10	-0.32	-0.62	-1.66	-9.48	-0.94	-1.61	-1.37	-0.30
Set6	0.78	-0.46	3.95	16.84	3.51	4.68	3.74	18.19	2.55	7.25	3.08	9.60	-1.85	-14.89
Set7	-0.19	-3.63	1.13	-6.36	0.77	-5.93	0.90	-5.19	0.33	-5.73	1.17	0.36	0.65	-1.04
Set8	1.71	5.05	2.05	7.38	3.22	11.85	3.58	13.12	1.01	-0.99	2.27	9.52	-0.69	-6.12
Set9	0.01	9.14	-0.57	-0.48	-0.54	-3.13	-1.72	8.21	-0.57	7.14	0.75	15.91	-0.52	5.60
Set10	1.41	10.78	-1.71	-33.29	-4.25	-51.49	2.06	1.29	-0.41	-26.07	1.15	-11.47	-4.00	-48.22
Set11	2.24	4.49	1.15	5.04	-0.67	-5.13	1.75	6.21	0.89	8.32	2.52	10.21	0.79	11.07
Set12	3.55	3.45	1.70	-3.11	3.20	-7.71	2.34	8.18	3.95	-3.81	1.13	6.45	1.22	5.09
Set13	1.96	5.80	1.21	-0.67	-0.59	-5.70	-0.03	-2.85	-1.18	-7.75	-1.02	0.64	-1.58	-6.10
Set14	2.76	13.86	1.47	10.28	2.32	-2.98	3.01	15.11	1.97	-0.34	3.46	17.01	1.18	14.07
Set15	0.84	1.92	0.02	1.30	0.39	2.62	0.73	-1.74	0.03	3.33	0.16	1.78	-0.78	-2.67
Set16	4.58	20.29	4.15	17.88	3.44	11.64	4.68	15.44	4.04	12.87	4.36	21.08	1.64	18.09
Set17	-1.75	-7.68	-1.09	-6.33	-3.09	-12.06	-1.27	-4.80	-2.92	-11.62	-0.30	0.70	-1.73	-4.57
Set18	-1.04	-9.24	0.96	3.12	-1.76	-10.89	0.22	-3.80	-0.07	-5.07	0.63	-0.42	-3.30	-13.53
Set19	0.40	1.95	-0.28	1.60	-3.44	-9.08	-1.66	-3.87	-1.05	-2.07	0.77	4.59	1.87	5.41
Set20	-2.53	-16.48	0.30	-3.60	-6.75	-33.54	0.80	0.89	-0.94	-6.94	0.01	-3.57	-1.95	-5.82

Table 18 Continued

Set21	-0.01	0.00	-0.62	-3.25	-1.45	-8.16	-1.35	-5.31	-6.41	-26.76	-0.52	-1.77	-1.20	-5.67
Set22	0.08	0.23	0.37	1.89	-0.18	-5.22	1.69	7.67	2.23	9.69	-0.26	-1.19	-2.54	-9.23
Set23	0.16	-0.61	-0.32	-1.74	0.07	-2.31	-0.52	-4.90	-2.02	-8.50	-0.11	0.92	-0.25	1.01
Set24	0.45	-0.17	-0.37	-2.93	-3.37	-15.67	1.05	1.67	-0.31	-2.00	1.01	2.12	-1.59	-4.96
Set25	2.72	7.11	2.57	6.37	3.33	5.99	2.97	6.19	3.05	2.20	3.10	8.87	2.45	7.16
Set26	-0.21	-4.44	0.79	-0.53	1.23	4.36	1.63	20.85	2.24	8.02	-0.70	-6.22	-0.83	-8.17
Set27	-0.19	0.21	-0.71	-6.91	-1.23	-9.53	-0.03	-3.33	0.03	-1.84	0.92	0.58	0.73	1.98
Set28	2.83	9.81	2.79	4.42	2.19	-1.71	3.81	4.67	0.84	-1.11	0.95	0.21	-2.43	-12.12
Set29	1.80	0.10	0.60	0.91	1.66	-1.78	0.16	-0.89	0.41	1.61	1.13	1.70	-1.20	-0.87
Set30	5.84	19.61	6.78	12.73	4.40	7.07	7.14	16.87	6.38	23.26	4.96	15.76	3.01	9.16
Set31	-0.12	0.24	-1.37	-8.19	-0.97	-0.93	-0.57	-0.02	-1.01	-5.56	-0.11	3.36	-2.87	3.72
Set32	0.73	-2.62	1.56	5.17	1.33	2.56	1.46	0.05	-0.27	-8.33	-0.21	0.26	-0.56	0.64

Table 19 - Average percentage improvement in total completion time and total maximum tardiness in each set where $t_{BV} = 2400$ and $t_{CIP} = 200$

	TC_{max} (Policy1 with t_{CIP} = 200)	TL_{max} (Policy1 with t_{CIP} = 200)	TC_{max} (Policy2 with t_{CIP} = 200 & θ = 0.01)	TL_{max} (Policy2 with t_{CIP} = 200 & θ = 0.01)	TC_{max} (Policy2 with t_{CIP} = 200 & θ = 0.02)	TL_{max} (Policy2 with t_{CIP} = 200 & θ = 0.02)	TC_{max} (Policy3 with t_{CIP} = 200 & θ = 0.01)	TL_{max} (Policy3 with t_{CIP} = 200 & θ = 0.01)	TC_{max} (Policy3 with t_{CIP} = 200 & θ = 0.02)	TL_{max} (Policy3 with t_{CIP} = 200 & θ = 0.02)	TC_{max} (Policy4 with t_{CIP} = 200 & q = 0.1)	TL_{max} (Policy4 with t_{CIP} = 200 & q = 0.1)	TC_{max} (Policy4 with t_{CIP} = 200 & q = 0.02)	TL_{max} (Policy4 with t_{CIP} = 200 & q = 0.2)
Set1	5.13	30.88	4.81	34.43	5.52	27.58	4.66	27.90	4.49	31.42	5.18	30.49	5.29	34.83
Set2	2.26	23.34	-0.21	15.44	-1.78	5.95	0.91	16.70	-0.53	7.22	2.28	27.05	-1.72	-1.21
Set3	2.58	19.45	2.74	21.71	2.50	23.32	1.75	19.90	1.89	14.46	2.85	22.72	1.22	17.48
Set4	-2.73	0.54	-2.48	-8.41	-6.99	-41.49	-0.96	2.77	-5.80	-24.46	-2.87	2.29	-6.82	-19.48
Set5	-1.80	6.64	-0.45	9.92	-1.64	5.19	-1.20	8.86	-2.55	0.78	-1.81	7.61	-2.24	8.31
Set6	-4.06	-31.68	-0.72	-9.14	-1.19	-20.83	-0.95	-8.03	-2.16	-20.71	-1.65	-17.05	-6.80	-49.55
Set7	-0.84	2.89	0.49	0.32	0.14	0.77	0.27	0.99	-0.31	0.68	0.54	6.04	0.02	4.77
Set8	-2.75	-17.83	-2.42	-14.74	-1.18	-9.57	-0.79	-7.25	-3.50	-24.46	-2.17	-12.84	-5.27	-31.95
Set9	7.36	39.96	6.80	33.11	6.84	31.49	5.75	38.57	6.78	38.08	8.03	44.02	6.87	37.12
Set10	1.13	31.62	-1.98	0.67	-4.53	-17.81	1.79	25.15	-0.72	4.57	0.90	16.00	-4.35	-21.64
Set11	7.32	28.59	6.27	28.89	4.49	21.37	6.84	29.81	6.03	30.89	7.57	32.60	5.91	33.37
Set12	1.39	7.62	-0.52	2.36	1.02	-2.91	0.18	11.74	1.78	1.19	-1.13	10.46	-1.03	9.41
Set13	5.44	31.51	4.69	26.79	2.94	23.74	3.53	27.25	2.44	22.83	2.51	26.50	1.98	22.86
Set14	0.95	14.19	-0.38	10.23	0.49	-2.02	1.20	15.39	0.13	-0.31	1.65	16.49	-0.68	14.01
Set15	3.44	14.94	2.64	14.83	3.02	16.14	3.34	12.18	2.66	16.45	2.79	15.04	1.85	11.21
Set16	1.89	6.60	1.45	4.12	0.72	-3.33	1.99	0.78	1.34	-1.89	1.66	7.71	-1.13	3.86
Set17	-0.35	-3.31	0.29	-1.95	-1.70	-7.66	0.11	-0.52	-1.54	-7.19	1.08	4.80	-0.33	-0.26
Set18	-2.52	-15.28	-0.50	-2.48	-3.24	-16.89	-1.24	-9.91	-1.53	-10.89	-0.83	-5.29	-4.83	-20.82
Set19	2.23	10.88	1.52	10.37	-1.61	0.77	0.17	5.75	0.77	7.09	2.57	13.30	3.68	14.03
Set20	-4.84	-31.30	-1.97	-16.84	-9.13	-50.27	-1.46	-11.88	-3.26	-20.78	-2.30	-17.16	-4.28	-19.50

Table 19 Continued

Set21	0.01	-0.01	-0.59	-3.26	-1.42	-8.18	-1.32	-5.31	-6.37	-26.67	-0.49	-1.77	-1.16	-5.67
Set22	-2.79	-13.18	-2.50	-11.28	-3.06	-19.48	-1.14	-4.64	-0.59	-2.37	-3.15	-14.55	-5.51	-24.05
Set23	-0.08	3.52	-0.56	2.46	-0.18	1.89	-0.77	-0.51	-2.27	-4.08	-0.36	5.04	-0.50	5.10
Set24	-1.51	-5.08	-2.35	-8.07	-5.39	-21.27	-0.90	-3.12	-2.30	-6.94	-0.93	-2.65	-3.59	-10.04
Set25	6.56	23.64	6.43	23.30	7.15	22.97	6.81	23.17	6.88	19.85	6.92	25.14	6.31	23.78
Set26	-3.33	-20.01	-2.30	-15.64	-1.85	-10.30	-1.44	10.39	-0.81	-6.13	-3.84	-21.97	-3.97	-24.41
Set27	2.65	17.87	2.22	12.23	1.70	10.10	2.85	14.99	2.88	16.29	3.76	18.08	3.57	19.35
Set28	-1.03	1.88	-1.07	-3.87	-1.67	-11.19	-0.02	-4.05	-3.09	-10.21	-3.04	-9.25	-6.53	-21.82
Set29	1.85	5.54	0.65	6.30	1.70	3.77	0.21	4.57	0.45	6.98	1.17	7.05	-1.17	4.60
Set30	3.11	9.79	4.07	2.17	1.62	-4.58	4.44	6.91	3.66	14.05	2.20	5.58	0.19	-1.90
Set31	1.11	7.58	-0.14	0.17	0.26	6.42	0.66	7.29	0.22	2.14	1.12	10.30	-1.65	10.62
Set32	-1.11	-8.66	-0.27	-0.79	-0.49	-3.43	-0.38	-5.83	-2.12	-14.58	-2.09	-5.83	-2.44	-5.46