SOLUTION APPROACHES FOR

MULTI OBJECTIVE PARALLEL MACHINE

SCHEDULING PROBLEMS

PhD Dissertation

Aseel N.H. SABTI

Eskişehir 2017

SOLUTION APPROACHES FOR MULTI OBJECTIVE PARALLEL MACHINE SCHEDULING PROBLEMS

Aseel N.H. SABTI

PhD Dissertation

Statistics Program

Assist. Prof. Dr. Zehra KAMIŞLI ÖZTÜRK

Eskişehir
Anadolu University
Graduate School of Sciences

December 2017

FINAL APPROVAL FOR THESIS

This thesis titled "Solution Approaches for Multi Objective Parallel Machine Scheduling Problems" has been prepared and submitted by Aseel N.H. SABTI in partial fullfillment of the requirements in "Anadolu University Directive on Graduate Education and Examination" for the Degree of Doctor of Philosophy (PhD) in Statistics Department has been examined and approved on 29/12/2017

Committee Members		Title Name Surname	Signature
Member (Supervisor)	:	Assist. Prof. Dr. Zehra KAMIŞLI ÖZTÜRK	....
Member	:	Prof. Dr. Yeliz MERT KANTAR	....
Member	:	Assist. Prof. Dr. Nergiz KASIMBEYLİ	....
Member	:	Prof. Dr. Hasan Kıvanç AKSOY	....
Member	:	Assist. Prof. Dr. Neslihan IYIT	....

Prof.Dr. Ersin YÜCEL
Director of Graduate School of Sciences

ABSTRACT

SOLUTION APPROACHES FOR MULTI OBJECTIVE PARALLEL MACHINE SCHEDULING PROBLEMS

Aseel N.H. SABTI

Statistics Program

Anadolu University, Graduate School of Sciences, December, 2017

Supervisor: Assist. Prof. Dr. Zehra KAMIŞLI ÖZTÜRK

This study considers the multi-objective parallel machine scheduling. A novel algorithm with name Sequence Job Minimum Completion Time (SJMCT) is proposed for unrelated parallel machines and non-identical jobs to minimize the two objectives. These objectives are minimization of maximum job completion time and total tardiness when each job is assigned only to one machine at time. The proposed algorithm's performance is compared with some common dispatching rules based on a small size problem (four machines and nine jobs).

Because of the complexity in multi-objective parallel machine scheduling problems, for large size problems, two novel metaheuristic algorithms SJMCT-NSGA-II based on Non-dominated sorting genetic algorithm (NSGA-II) and SJMCT-SPEA-II based on Strength Pareto evolutionary algorithm (SPEA-II) are proposed to obtain Pareto optimal solutions. The simulation results for 272 tests are reported to show the efficiency of these two algorithms. Two test problems of simulation experiences are done to study effects of the different parameters. In the simulations, the effects of generation numbers and job numbers are investigated. The results demonstrate that the proposed SJMCT-SPEA-II has better performed than the SJMCT-NSGA-II. Besides choosing the appropriate performance measures, Spacing and Spread Diversity Metrics are also ensured this result. Finally, the conclusions and some directions for future research are reported.

Keywords: Operations research**;** Scheduling; Unrelated parallel machine; Multi-objective evolutionary algorithms; SJMCT-NSGA-II and SJMCT-SPEA-II algorithms.

ÖZET

ÇOK AMAÇLI PARALEL MAKİNE ÇİZELGELEME PROBLEMLERİ İÇİN ÇÖZÜM YAKLAŞIMLARI

Aseel N.H. SABTI

İstatistik Anabilim Dalı

Anadolu Üniversitesi, Fen Bilimleri Enstitüsü, Aralık, 2017

Danışman: Yard. Doç. Dr. Zehra KAMIŞLI ÖZTÜRK

Bu çalışmada çok amaçlı paralel makine çizelgeleme problemi ele alınmıştır. Bağımsız paralel makineler ve özdeş olmayan iş dizileri için Ardışık İş Enküçük Tamamlanma Zaman (SJMCT) isimli yeni bir algoritma önerilerek iki amaç eniyilenmiştir. Bu amaçlar; her bir işin sadece tek bir zaman ve makineye atandığı durumdaki enbüyük tamamlanma zamanı ve toplam gecikmenin en küçüklenmesidir. Geliştirilen algoritmanın performansı, küçük boyutlu bir problem (dört makine ve dokuz iş) üzerinden çok kullanılan genel sevk etme kuralları ile karşılaştırılmıştır.

Büyük boyutlu problemler için çok amaçlı makine çizelgeleme problemlerindeki karmaşıklıklardan dolayı, Baskın Olmayan Sıralama Genetik Algoritma (NSGA-II) tabanlı ile Güçlü Pareto Evrimsel Algoritma (SPEA-II) tabanlı SJMCT-NSGA-II ve SJMCT-SPEA-II isimli iki yeni melez metasezgisel algoritma Pareto optimal çözümleri elde etmek için önerilmiştir. 272 simülasyon sonucu, geliştirilen algoritmaların etkinliğini göstermektedir. Değişik parametrelerin etkilerini göstermek için iki farklı problem üzerinden simülasyonlar yapılmıştır. Simülasyonlarda iterasyon sayısı ve iş sayısı etkileri araştırılmıştır. Sonuçlar, önerilen SJMCT-SPEA-II algortimasının SJMCT-NSGA-II'den daha iyi performansa sahip olduğunu göstermektedir. Uygun performans ölçülerini seçmeden önce, elde edilen Pareto çözümlerin etkiliğini göstermek için Yayılma ve Mesafe metrikleri de kullanılmıştır. Son olarak, sonuçlar ve gelecek çalışmalar için bazı öneriler de sunulmuştur.

Anahtar sözcükler: Yöneylem araştırması; Çizelgeleme; Özdeş olmayan paralel makine; Çok amaçlı evrimsel algoritmalar; SJMCT-NSGA-II; SJMCT-SPEA-II.

ACKNOWLEDGEMENTS

Allah the Almighty said: "And those who strive for Us – We will surely guide them to Our ways. And indeed, Allah is with the doers of good" [Quran] (29:69)

First of all, I would like to thank Türkiye Bursları (Scholarships program) for the great experience and support throughout the study period.

I would like to thank the Iraqi government and Ministry of Higher Education and Scientific Research for giving me this opportunity that contributes to my future scientific development. Also, I would like to thank the Iraq Culture Office in Ankara and its staff for their help and support.

Big thanks to AL-Irqia University and its staff for the great help they have provided to me.

I would like to thank my adviser Assist. Prof. Dr. Zehra Kamışlı Öztürk for her guidance and suggestions greatly helped me to focus on the topic of the thesis. Without her academic knowledge and dedications this thesis would not reach this point.

I would like to express my appreciation to my research committee members Prof. Dr. Yeliz Mert Kantar and Assist. Prof. Dr. Nergiz Kasimbeyli. Their willingness to provide assistance and guidance during the writing process was essential to the completion of this study.

I would like to thank my research committee members Assoc. Prof. Dr. Hakan Kıvanç Aksoy and Assist. Prof. Dr. Neslihan Iyit for their time and support.

I am deeply grateful to Prof. Dr. Aladdin Shamilov for his support and valuable information through the course of Entropy Optimization Methods and Applications.

I would like to thank Prof. Dr. Memmedağa Memmedli for his support and science guidance through the courses of Artificial Neural Network I and II.

I would also like to thank Prof. Dr. Refail Kasımbeyli for the knowledge he conveyed to me through the courses in Operations Research, Constrained and Non-constrained Optimization Methods.

It is pleasure to me to thank my friends Res. Asst. Nihal Ince and Res. Asst. Gökçen Uysal for giving me the special help whenever I needed.

I thank my family, especially my mother and my father for being such a great support and for pushing me forward my dreams.

Aseel N.H. SABTI

29/12/2017

STATEMENT OF COMPLIANCE WITH ETHICAL PRINCIPLES AND RULES

I hereby truthfully declare that this thesis is an original work prepared by me; that I have behaved in accordance with the scientific ethical principles and rules throughout the stages of preparation, data collection, analysis and presentation of my work; that I have cited the sources of all the data and information that could be obtained within the scope of this study, and included these sources in the references section; and that this study has been scanned for plagiarism with “scientific plagiarism detection program” used by Anadolu University, and that “it does not have any plagiarism” whatsoever. I also declare that, if a case contrary to my declaration is detected in my work at any time, I hereby express my consent to all the ethical and legal consequences that are involved.

Aseel N.H. SABTI

CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

LIST OF ABBREVIATIONS

Abbreviation	**Explanation**
SJMCT	: Sequence Job Minimum Completion Time
NSGA	: Non-Dominated Sorting Genetic Algorithm
SPEA	: Strength Pareto Evolutionary Algorithm
SJMCT-NSGA-II	: Sequence Job Minimum Completion Time Based on NSGA-II
SJMCT-SPEA-II	: Sequence Job Minimum Completion Time Based on SPEA-II
ERD	: Earliest Release Date
EDD	: Earliest Due Date
MS	: Minimum Slack First
LPT	: Longest Processing Time
WSPT	: Weighted Shortest Processing Time
SPT	: Shortest Processing Time
CP	: Critical Path
LNS	: Largest Number Of Successors
SIRO	: Service In Random Order
SST	: Shortest Setup Time
LFJ	: Least Flexible Job
SQNQ	: Shortest Queue At The Next Operation
TSA	: Tabu Search Algorithm
MA	: Memetic Algorithm
APS	: Advanced Planning And Scheduling Systems
LS	: List Scheduling Rule
PMS	: Parallel Machine Scheduling
PMSP-E/T	: Parallel Machine Scheduling Problem With Earless And Tardiness Penalties
ML	: Maximum Likelihood
MCTE	: Maximum Completion Time Estimation
MLPT	: Modified Longest Processing Time

SA	: Simulated Annealing
GAs	: Genetic Algorithms
GA-DR-P	: Genetic Algorithm With Processing –Time Based Dispatching Rule
PSO	: Particle Swarm Optimization Algorithm
SSO	: Simplified Swarm Optimization Algorithm
MOSA	: Multi-Objective Simulated Annealing
PMBSP	: Bi-Criteria Scheduling Problem For Parallel Machine
ANN	: Artificial Neural Networks
MOPSO	: Multi-Objective Particle Swarm Optimization
CMOPSO	: Conventional Multi-Objective Particle Swarm Optimization
GLS	: Genetic Local Search
MOCO	: Multi-Objective Combinatorial Optimization
TSP	: Traveling Salesman Problem
PESA	: Pareto Envelope- Based Selection Algorithm
MPGA	: Multiple Population Genetic Algorithm
TWT	: Total Weighted Tardiness
TWC	: Total Weighted Completion Time
SPGA	: Sub-Population Genetic Algorithm
FLC-NSGA-II	: Fuzzy Logic Non-Dominated Sorting Genetic Algorithm
TPM	: Two Phase Method
VEGA	: Vector Evaluated Genetic Algorithm
AL	: Approach By Localization
CGA	: Controlled Genetic Algorithm
PVNS	: Parallel Variable Neighborhood Algorithm
FJSP	: Flexible Job Shop Scheduling
MOEA	: Multi-Objective Evolutionary Algorithm
MOP	: Multi-Objective Optimization Problem
ACO	: Ant Colony Optimization
PF_{Known}	: Pareto Optimal Front (P)

PF_{True}	: Pareto Obtained Solution (*S*)
MOO	: Multi-Objective Optimization
ER	: Error Ratio
GD	: Generational Distance
SP	: Spacing Metric
OS	: Overall Pareto Spread
IGD	: Inverted Generational Distance
MPFE	: Maximum Pareto Front Error

1. INTRODUCTION

Scheduling is a field of study concerned with optimal allocation or assignment of limited resources, over time, to a set of tasks or activities (Parker, 1996). Tasks and resources can stand for jobs and machines in a manufacturing system, patients and hospital equipment in health care problem, class and teachers in educational institution, ships and dockyards in a logistic system, programs and computers, or cities and traveling salesmen.

In each of these different systems, the decision makers try to optimize an objective function. For example, minimization of total tardiness, minimization of total course clashes of students and etc.

As Pinedo, (2008) mentioned, scheduling is a decision-making process, plays an important role in most manufacturing and production systems.

In general, the machine scheduling problems are first classified into two classes in terms of the nature of problem. The first class is the deterministic machine problem when the processing constraints and parameters can be ascertained with certainty. The second is the uncertain machine scheduling problem when some processing conditions or parameters cannot be determined in advance.

The deterministic machine scheduling problems are categorized into four types according to shop configuration. These types are classified as: single machine, parallel machines, flow shop, and job shop. In parallel-machine shop, a number of one operation jobs can be processed on any of machines. In flow shop, machines are arranged in a serial fashion, and each job has to pass through each machine. Job shop is a configuration in which each job has different processing routes.

The uncertain machine scheduling problems are grouped into two types in terms of the description method of uncertainty. The first type is fuzzy machine scheduling problem in which the processing conditions and parameters are modeled using fuzzy number. The second is stochastic machine scheduling problem in which stochastic variables are used to indicate the processing constraints and parameters.

Today's parallel machine scheduling has become one of the most attractive subjects because of the competition in production environments. Parallel machine scheduling is one of the machine scheduling classes. In addition, the unrelated parallel machine scheduling which means there is no relationship among these machines (Eroglu, Ozmutlu and Ozmutlu, 2014). Therefore, this study deals with this type of scheduling problem and the motivation behind this thesis is to solve large size of unrelated parallel machine scheduling with non-identical jobs to optimize two objectives represented by minimizing the maximum completion time and total tardiness.

The organization of this thesis is as follows: Chapter 2 presents a brief overview of the literature related with single machine scheduling, single and multi-objective parallel machine scheduling. Also, flow shop and job shop scheduling problems.

In Chapter 3, a new mathematical model is proposed for unrelated parallel machines with non-identical jobs when jobs have different processing times on each machine. Sequence Job Minimum Completion Time (SJMCT) algorithm is used to solve this model. The minimum random processing time is used to assignment problems. The aim is to minimize two objectives: the maximum completion time and total tardiness. The comparison between SJMCT and some dispatching rules is represented.

In Chapter 4, the most challenge of this study is proposed two novel heuristic algorithms Sequence Job Minimum Completion Time-based NSGA-II (SJMCT-NSGA-II) and Sequence Job Minimum Completion Time-based SPEA-II (SJMCT-SPEA-II). These algorithms are able to solve large and more complex multi-objective parallel machine scheduling problems.

In Chapter 5, firstly, 32 simulation test problems are made with 60 jobs and with different generation numbers (40, 100, 300 and 500). Secondly, 240 simulation test problems are reported with different number of jobs (20, 60 and 100) where the generation number is 500. All of these tests with different crossover and mutation probabilities and with the same size of population are used to compare between these two algorithms. In addition, the spacing and spread diversity metrics are used to find the best algorithms.

In Chapter 6, the conclusions, the contribution of this thesis and some suggesting future research directions are explained.

2. SOME DEFINITIONS AND LITERATURE REVIEW

2.1. Background of Machine Scheduling

In single machine scheduling models, there is only one machine and the routes consist of only one operation performed on this machine (Akyol, 2006). On the other hand, in parallel machine scheduling there are N jobs and M machines and each job can be processed on any one of available machines (Allahverdi, Gupta and Aldowaisan, 1999). Also, there are three types of parallel machines (Ma, Chu and Zuo, 2010) and (Strusevich and Rustogi, 2017):

- **Identical machines:** If each processing time of a job is independent of the machine when performing a job.
- **Uniform machines:** The machines operated at different speeds.
- **Unrelated parallel machines:** The processing time of a job depends on the machine assignment.

The basic parameters in machine scheduling are given bellow (Pinado, 2005):

Processing time (P_{ij}): It is the required time of job j to complete its processing on machine i.

Release date (r_j): It is the time at which job $j \in N$ becomes available for processing.

Deadline: It is the time by which a job $j \in N$ must be completed; unlike the due date, a deadline is a hard constraint.

Due date (d_i): It is the time at which job $j \in N$ is expected to complete.

For any scheduling problem, the following primary criteria are used as a function.

Completion time (C_i): It is the popular quality measure, represents the times by which jobs are completed on machine i.

Lateness (L_i): Lateness is expresses the deviation of the completion time of a job j from a due date, it can be positive, negative or zero $L_j=C_j-d_j$.

Tardiness (T_i): Tardiness is the non-negative quantity that can be calculated to show how much time a job is completed after its due date $T_j = max\{0, C_j - d_j\} = max\{L_j, 0\}$.

According to Lawler et al., (1993) and Xing and Zhang, (2000) the three field classification of machine scheduling are $\alpha/\beta/\gamma$, where:

- α describes machine environment, $\alpha \in \{P, Q, R\}$

$\alpha = P$: Identical parallel machines, $p_{ij} = p_j$ for all M_i,

$\alpha = Q$: Uniform parallel machines, $p_{ij} = p_j/r_j$ for a given speed r_i of M_i,

$\alpha = R$: Unrelated parallel machines, $p_{ij} = p_j/r_{ij}$ for given job-dependent speeds r_{ij} of M_i.

- β describes job characteristics, $\beta \in \{o, pmtn\}$

$\beta = pmtn$: Preemption is allowed; the processing of any operation may be interrupted and resumed at a later time.

$\beta = o$: No preemption is allowed.

- γ describes optimality criteria. In general $\gamma \in \{C_{max}, L_{max}, \sum C_j, \sum T_j, \sum wC_j\}$.

2.2. Dispatching Rules

The term dispatching rule is used to determine the next job waiting in front of a machine when the machine becomes available (Pinedo, 2005). The main advantage of dispatching rules is that, they are easy to understand, easy to apply and require relatively little computer time. Their primary disadvantage is that, they can't guarantee an optimal solution (Akyol, 2006). Dispatching rules can be classified in different ways. Static rules are not time dependent and they are just a function of the job and/or machine data. Dynamic rules are time dependent. Another classification of dispatching rules is according to the information they are based upon. There are many basic dispatching rules but a sample of these rules is given as bellow (Pinedo, 2005) and (Massabò, Paletta and Ruiz-Torres, 2016).

- *The Earliest Release Date first (ERD) rule*: This rule tends to minimize the diversity in the waiting times of the jobs at a machine. The job which has the earliest release date is selected next to be processed.
- The *Earliest Due Date first (EDD) rule*: This rule refers to minimize the maximum lateness among the jobs waiting for processing. The job which has the earliest due date is selected next to be processed.
- The *Minimum Slack first (MS) rule*: When a machine is freed the minimum slack job will schedule next. Also, the remaining slack of each job at that time *t* is defined as $\max(d_j - p_j - t, 0)$.
- The *Longest Processing Time first (LPT) rule*: The LPT rule sorts jobs in decreasing order of processing times and iteratively assigns each job to the machine which would complete in the shortest processing time.

- The *Weighted Shortest Processing Time first (WSPT) rule*: This rule schedules the job with highest ratio of weight over processing time. Jobs are ordered in decreasing order of w_j/p_j. If all the weights are equal, the WSPT rule reduces to the Shortest Processing Time first (SPT) rule.
- The *Critical Path (CP) rule*: This rule is related with jobs subject to precedence constraints. The job which has the longest string of processing times in the precedence constraints graph (Prec) is selected next to be processed.
- The *Largest Number of Successors (LNS) rule*: This rule also is used when the jobs are subject to precedence constraints. The job which has the largest number of jobs following it is selected next to be processed.
- The *Service in Random Order (SIRO) rule*: In this rule, the next job is selected at random from those waiting for processing.
- The *Shortest Setup Time first (SST) rule*: In this rule, the job with the shortest setup time is firstly selected for processing.
- The *Least Flexible Job first (LFJ) rule*: This priority rule is used with the non-identical parallel machine and the jobs are subject to machine eligibility constraints. Job j can only be processed on a specific subset of the m machines, say M_j. It selects the job which is processed on the smallest number of remaining machines i.e., the job with the fewest processing alternatives.
- The *Shortest Queue at the Next Operation (SQNO) rule*: In job shops, this rule selects the job with the shortest queue at the next machine on its route for processing. At the next machine the length of the queue can be measured in different ways. It may be simply the number of jobs waiting in queue or it may be the total amount of work waiting in queue.

Pinedo, (2005) describes the basic dispatching rules mentioned above as given in Table 2.1.

Table 2. 1. *Summary of dispatching rules (Pinedo, 2005)*

	RULE	DATA	OBJECTIVES
Rules Dependent on Release Dates and Due Dates	ERD EDD MS	r_j d_j d_j	Variance in Throughput Times Maximum Lateness Maximum Lateness
Rules Dependent on Processing Times	LPT SPT WSPT CP LNS	p_j p_j p_j, w_j $p_j, prec$ $p_j, prec$	Load Balancing Over Parallel Machines Sum of Completion Times, WIP Weighted Sum of Completion Times, WIP Makespan Makespan
Miscellaneous	SIRO STT LFJ SQNQ	— s_{jk} M_j —	Ease of Implementation Makespan and Throughput Makespan and Throughput Machine Idleness

2.3. Literature Review

In this section, many relevant works about single machine scheduling problems, single objective parallel machine solved by exact and heuristic solution approaches, multi-objective parallel machine scheduling problems solved by different and evolutionary solution approaches are indicated. Other relevant works in shop scheduling problems are also viewed.

2.4. Relevant Works in Single Machine Scheduling Problems

Dyer and Wolsey (1990) considered the formulation of the single machine sequencing problem with release dates as a mixed integer programming problem to minimize the weighted sum of start (or completion) times for the *n*-jobs *1*-machine problem. They showed that; a first hierarchy of relaxations (obtained by combining enumeration of initial sequences with Smith's rule) and the second hierarchy of relaxations (obtained by studying various relaxations and alternative formulations) can be formulated as a linear programming problem.

Laguna, Barnes and Glover, (1991) used three local searches strategies within tabu search algorithm (TSA) to minimize the sum of the set up costs and linear delay penalties. Firstly, they used TS approach of making a succession of pairwise job exchange or swaps to move from one trail solution to another. Next, they used the insert moves to define the local neighborhood of each trail solution. Finally, a hybrid TSA employed to swap and insert moves. The experiment results for benchmark problem of up to 60 jobs illustrate that, there is an advantage in using more than one strategy to move from one trail solution to another with in a TSA method.

Crauwels, Potts and Van Wassenhove, (1998) presented several local search heuristics to minimize total weighted tardiness. A new binary representation and the additional diversifying element in the tabu search methods are introduced to represent solutions. The extensive computational tests ensure that, binary encoding scheme produces very robust results for the total weighted tardiness problem.

França, Mendes and Moscato, (2001) proposed a new Memetic Algorithm (MA) with due dates and sequence dependent setup time to minimize total tardiness. The Genetic algorithms GAs and MA are compared with three other heuristics. Several neighborhood reduction schemes are improved starting with a set of random generated parameters. The computational results using a non-structured population and less elaborated neighborhoods led to a considerable loss of performance especially for large instances.

2.5. Relevant Works in Single Objective Parallel Machine Scheduling Problems

2.5.1. Exact solution approaches for single objective parallel machine scheduling problems

The most associated studies in parallel machine scheduling for single objective can be summarized as follows:

Balakrishnan, Kanet and Sridharan, (1999) considered the problem of scheduling *n* jobs on *m* parallel machines that operating at different speeds (known as uniform parallel machines), to minimize the sum of earliness and tardiness costs. They presented a mixed integer mathematical model to solve small sized problems (10 jobs and 5 machines).

Uma, Wein and Williamson, (2006) investigated from a theoretical perspective, the relationship between combinatorial relaxation and several linear programming relaxation -based lower bounds for three scheduling problems to minimize the average weighted completion time of the jobs scheduled. As a result, they obtained the first worst-case analysis of the quality of the lower bounds delivered by these combinatorial relaxations.

Senthiil, Selladurai and Rajesh, (2007) proposed a new algorithm, the extension of the traveling salesman problem in a parallel machine environment to minimize the makespan. The proposed algorithm extends the optimization of a single machine problem to a parallel machine problem using the traveling salesman problem for scheduling. Moreover, they used the ant colonies optimization algorithm to find a solution for this new proposed problem. The simulation results show that, the algorithm is able to optimize the different scheduling problems.

Lu, Zhang and Yuan, (2008) considered the unbounded parallel batch machine scheduling with release dates and rejection. A job is either rejected with a certain penalty having to be paid, or accepted and processed in batches. The aim is to minimize the sum of the makespan of the accepted jobs and the total rejection penalty of the rejected jobs. They showed that, the problem is binary NP-hard and it can be solved in polynomial time when the jobs have the same rejection penalty.

Lin and Liao, (2008) proposed an optimal algorithm for solving the uniform parallel machine problem to minimize the makespan. Two important theorems are developed for the problem. The first theorem provides an improved lower bound as the starting point for the search, and the second theorem further accelerates the search speed in the algorithm.

Unlu and Mason, (2010) represented different Mixed Integer Programming formulations based on different types of decision variables for non-preemptive parallel machine scheduling problems. Different performance measures such as, total weighted completion time, makespan, maximum lateness, total weighted tardiness and total number of tardy jobs are used to evaluate the formulation efficiency.

Ruiz and Andrés-Romano, (2011) considered a novel complex scheduling problem with unrelated parallel machine problem and job sequence dependent setup times. A combination of total assigned resources and total completion time is used as a criterion. The good performance of the mixed integer programming model with large number of constraints and variables and other heuristics algorithm are obtained.

Zhang and Luo, (2013) studied the rejection and a fixed non-availability interval on two identical parallel machines when the processing time of a job is a simple linear increasing function of its starting time. The objective is to minimize the makespan of the accepted jobs plus the total penalty of the rejected jobs. In addition, for two identical machines a "fully polynomial-time approximation scheme" presented to show that the problem is NP-hard in the ordinary sense only.

Öztürk and Ornek, (2014) improved a mixed integer programming formulations for advanced planning and scheduling systems (APS). The objective function includes the cost of idle times of the machines and penalties on tardiness and earliness. They developed a basic model with sequence dependent setups time and transfer times between machines. They also showed that the presented model can be used to provide delivery times for customer orders in case due dates are not specified.

2.5.2. Heuristic and metaheuristic solution approaches for single objective parallel machine scheduling problems

Frenk and Rinnooy Kan, (1987) studied the behavior of list scheduling rules (LS) to minimize makespan for parallel machines of different speed. The jobs are assigned successively to the first available machine in the order. The processing requirements of the jobs are independent, identically non-negative random variable. They obtained strong asymptotic optimality results for the LPT (longest processing time) rule, when the jobs are assigned to the machines in order of non- increasing processing requirements.

Cheng and Gen, (1997) used Memetic Algorithm (hybrid genetic algorithm) to minimize the maximum weighted absolute lateness. The computational experiments demonstrate that the hybrid genetic algorithm outperforms the genetic algorithms and the conventional heuristics.

Sivrikaya-Şerifoğlu and Ulusoy, (1999) considered the parallel machine problem scheduling with earliness and tardiness penalties (PMSP-E/T). The problem consisted of scheduling a set of independent jobs with sequence-dependent setup times to minimize the sum of the weighted earliness and tardiness values. Also, they employed two

Genetic Algorithms (GAs) approaches. Firstly, they used a crossover operator to solve multi-component combinatorial optimization problems. Secondly, they didn't use a crossover operator. The computational results showed that, GAs with crossover operator is more attractive in large sized and more difficult problems.

Xing and Zhang, (2000) studied the parallel machine scheduling (PMS) problem with a hypothesis: a job cannot be processed on two machines simultaneously if preemption is allowed, and under a hypothesis: any part of a job can be processed on two different machines at the same time, they called it PMS with splitting jobs. They presented some simple cases which are polynomial solvable. Furthermore, a heuristic maximum likelihood (ML) used to convert the original problem to a new problem by using the maximum completion time estimation (MCTE) and its worst-case analysis were shown for $P/split/C_{max}$ with independent job setup times. The objective was to minimize the total cost.

Weng, Lu and Ren, (2001) proposed seven heuristic algorithms tested by simulation to scheduling a set of independent jobs on unrelated parallel machines with job sequence dependent setup times to minimize the total weighted completion time.

Gupta and Ho, (2001) developed an optimization algorithm and polynomially bounded heuristic solution procedures for the scheduling jobs on two identical parallel machines to hierarchically minimize the makespan subject to the optimality of the total flow time.

Lin and Liao, (2008) developed the algorithm which has an exponential time complexity in addition to the optimal algorithm mentioned before. They also examined the effectiveness of the popular LPT heuristic for solving the uniform parallel machine problem with the objective of minimizing the makespan.

Koulamas and Kyparisis, (2009) proposed a modified longest processing time (MLPT) heuristic algorithm for the two uniform machine makespan minimization problems. They showed that the performance of the LPT heuristic for the ($Q_2//C_{max}$) problem can be improved by sequencing the longest three jobs optimally. The results demonstrate the applicability of this approach (already implemented for identical parallel machine scheduling problems) to a uniform parallel machine environment.

Yeh et al., (2014) proposed two meta-heuristics, the Simulated Annealing (SA) and the Genetic Algorithm (GA) for parallel machine scheduling with fuzzy processing times and learning effects with aim to minimize the makespan. The results show that, SA is better than GA for this problem.

Ou, Zhong and Wang, (2015) found new properties and improved an *O (n log n+ n/ε)* heuristic for parallel machine scheduling with rejection. When the jobs are accepted and processed or rejected and paid a rejection penalty to minimize the completion time of the last accepted job plus the total penalty of all rejected jobs.

Joo and Kim, (2015) proposed hybrid Genetic Algorithms with three dispatching rules for unrelated parallel machine scheduling to minimize the total completion time. MIP Mixed Integer Programming model derived to find the optimal solution. The results show that, GA using chromosomes with processing-time-based dispatching rule (GA_DR_P) could offer a better solution in both effectiveness and efficiency.

Yeh, Chuang and Lee, (2015) proposed a scheduling problem on uniform parallel machines where the objective is to minimize the makespan. Three algorithms, Genetic Algorithm (GA), Particle Swarm Optimization Algorithm (PSO), and Simplified Swarm Optimization Algorithm (SSO) are proposed to solve the problem. In results, SSO has better solutions in a small number of jobs, and the GA approach has better solutions for large job-sized problems.

Massabò, Paletta and Ruiz-Torres, (2016) developed a posterior worst-case performance ratio of the LPT heuristic for scheduling independent jobs on two uniform parallel machines to minimize the makespan. The posterior worst-case performance ratio depends on the index of the latest job inserted in the machine where the makespan takes place. They show that the posterior worst-case performance ratio is tight.

Similar to the previous work, other review of the scheduling problems with multiple objectives is given in the next subsection.

2.6. Relevant Works in Multi-Objectives Parallel Machine Scheduling Problems

2.6.1. Solution approaches for multi-objective parallel machine scheduling problems

Suresh and Chaudhuri, (1996) proposed an algorithm based on Tabu Search to minimize the makespan and maximum tardiness when each job has required a single stage of processing for unrelated parallel machine scheduling. Also, they compared their solutions with other heuristic algorithms. The extensive experiments show that, the proposed algorithm outperforms in the quality of solution and execution time.

Loukil, Teghem and Tuyttens, (2005) considered a Multi-objective Simulated Annealing (MOSA) to find the efficient schedules for a large set of scheduling models. They analyzed the solution correspond to one machine, parallel machines and permutation flow shops. Thereafter, they designed a Multi-objective Tabu Search Algorithm (MOTSA) and tested it numerically to compare with MOSA algorithm.

Tavakkoli-Moghaddam, Taheri and Bazzazi, (2008) proposed a new model to minimize the number of tardy jobs and total completion time for unrelated parallel machines scheduling problem with different machine speeds. They used a two-level mixed-integer programming and goal programming approach to solve the scheduling problem with precedence constraints and non-independent jobs. The good performance of proposed model is obtained in small and medium-sized problems. They solved the problem with (6, 8 and 10) jobs, (2, 3 and 4) machines and (3, 4 and 5) number of precedence constraints.

Mazdeh et al., (2010) studied the bi-criteria scheduling problem (PMBSP) for parallel machines with machine effects and job deterioration to minimize total tardiness and machine deteriorating cost. They proposed the LP-metric method and a metaheuristic algorithm based on Tabu Search. Numerical examples used to assess the effectiveness and efficiency of the model.

Cheng et al., (2012) considered the parallel batch processing machines with non-identical job sizes to minimize makespan and total completion time. They used a mixed integer programming method to find the optimal solution. Thereafter, they proposed a polynomial time algorithm and the worst case ratios to minimize the objective values. The reported results indicate to the efficiency of the algorithm.

Muralidhar and Alwarsamy, (2013) considered parallel machines scheduling problem to minimize the combined objective function of the makespan, total tardiness and total earliness. Artificial Neural Networks (ANN) was applied and compared with heuristic algorithms. The results show that, the adapted procedure is simpler and it can be used for scheduling large number of jobs without training the network again.

Torabi et al., (2013) considered a fuzzy multi-objective programming model for solving an unrelated parallel machine scheduling problem. A Multi-objective Particle Swarm Optimization (MOPSO) algorithm was proposed to find Pareto frontier. The aim is minimizing total weighted flow time, total weighted tardiness and total machine load variation. They compared the proposed algorithm with conventional multi-objective particle swarm optimization algorithm. Results of test problems observed that the proposed MOPSO is better performed than CMOPSO based on the linear statistical model for three hypotheses tests. Also, the ANOVA results have been summarized to study the effect of i^{th} method, j^{th} objective space and the effect of interaction between i^{th} method and j^{th} objective space.

Yang, (2013) presented unrelated parallel machine scheduling problems with deterioration effects and deteriorating multi-maintenance activities. Two models of scheduling have been examined: the job and position dependent on deterioration model and the time dependent deterioration model. The aim is minimizing total completion time to find jointly the optimal maintenance frequencies, the optimal maintenance positions and the optimal job sequences. A polynomial time solution was applied for variant and some special cases.

Lin and Lin, (2015) proposed a bicriteria heuristic and a Tabu Search Algorithm. The objective is to minimize the makespan and total weighted tardiness for unrelated parallel machine scheduling problems with release dates. The results indicate that, the proposed TSA is outperforms other heuristic algorithms.

2.6.2. Evolutionary solution approaches for multi-objective parallel machine scheduling problems

Zitzler, Laumanns and Thiele, (2001) improved Strength Pareto Evolutionary Algorithm (SPEA-II) for finding or approximating the Pareto-optimal set for multi-objective optimization problems and compare SPEA-II with SPEA and two other modern elitist methods, Pareto envelope- based selection algorithm (PESA) and NSGA-II, on different test problems.

Jaszkiewicz, (2002) proposed a novel Genetic Local Search algorithm (GLS) algorithm for multi-objective combinatorial optimization problems (MOCO) to find an efficient solution in both combinatorial optimization and non-convex continuous optimization problems. The results show that, the proposed algorithm has better performance than multi-objective methods based on GLS or based on traveling salesman problem TSP.

Cochran, Horng and Fowler, (2003) proposed a two-stage multiple population genetic algorithms (MPGA). The goal is to minimize makespan and total weighted tardiness (TWT). They also compared MPGA with benchmark method and multi-objective genetic algorithm MOGA. Moreover, The MPGA is extended to scheduling problems with three objectives: makespan, TWT, and total weighted completion times TWC. The experiment results in most of the test problems show that, MPGA has better performs than MOGA.

Chang, Chen and Hsieh, (2006) proposed a modified sub-population genetic algorithm SPGA and an adaptive SPGA for parallel machine scheduling problem to minimize total tardiness time and makespan. They show that, the results obtained by adaptive SPGA and modified SPGA are more efficient than other multi-objective optimization genetic algorithms NSGA-II and SPEA-II for large size problems.

Balasubramanian et al., (2009) proposed iterative SPT–LPT–SPT heuristic and a bicriteria genetic algorithm for interfering job sets. Where, the makespan minimized for one of the sets and the total completion time minimized for the other. Integer programming formulation solution was compared with the heurestic and GA algorithms to show the effeiciency of these algorithms. Results show that, the heuristic and the genetic algorithm provide high solution quality and are computationally efficient.

Li et al., (2010) considered an identical parallel machines scheduling problem with release dates, due dates, and sequence-dependent setup times to minimize the makespan and the total tardiness. A new mathematical model and two metaheuristics NSGA-II (Non-dominated Sorting Genetic Algorithm–II) and SPEA-II (Strength Pareto Evolutionary Algorithm-II) were explained. A full enumeration method was applied to find the absolute Pareto optimal solutions. The results show that, the full enumeration method cannot solve the problems with more than 8 jobs.

Mirabedini and Mina, (2012) proposed multi-objective model including the problem of preventive maintenance and production scheduling by one objective. The weighted-sum objective function is considered with five parts; minimizing maintenance cost, makespan, total weighted completion time of jobs, total weighted tardiness, and maximizing machine availability. Multi-objective genetic algorithms solved the model and found a local optimum solution.

Li et al., (2012) presented an identical parallel machine scheduling problem with release dates, due dates and sequence dependent setup times to minimize the makespan and the total tardiness. They proposed a new mathematical model and developed two metahurestics as non-dominated sorting genetic algorithm (NSGA-II) and a fuzzy logic guided NSGA-II (FLC-NSGA-II). Also, two phase method TPM was used as an exact method to solve the problem. The FLC-NSGA-II was compared with the TPM method for the small size problems. Results indicate to the ability of FLC-NSGA-II to find the absolute optimal solutions and the TPM method can solve the problems with maximum 10 jobs.

Bandyopadhyay and Bhattacharya, (2013) represented a multi-objective parallel machine scheduling problem with minimization of three objectives: total cost due to tardiness, deterioration cost for the machines and makespan. They solved the mathematical model by multi-objective evolutionary algorithms modified NSGA-II, NSGA-II and SPEA-II. The processing, setup and deterioration costs were generated randomly to follow uniform distribution. Simulation experiments were performed to compare these algorithms. The comparison shows that, the modified NSGA-II has better performance than the NSGA-II and SPEA-II.

Wang and Liu, (2015) considered a multi-objective parallel machine scheduling problem with flexible preventive maintenance activities and with two kinds of resources (machines and moulds). The aim is to minimize the makespan for the production, the unavailability of the machine system and the unavailability of the mould system. They proposed a multi-objective integrated optimization method and NSGA-II adaption. The computationally results show that, the integrated optimization method of production scheduling and preventive maintenance outperforms the method with periodic preventive maintenance for this problem.

2.7. Relevant Works in Shop Scheduling Problems

Murata, Ishibuchi and Tanaka, (1996) proposed a multi-objective genetic algorithm for flow shop scheduling. They used crossover operation based on a weighted sum of multiple objective functions with variable weighted. The two objectives were determined as minimizing the makespan and total tardiness and three objectives were determined as minimizing the makespan, total tardiness and total flow time are examined. The simulation experience represents the ability of multi-objective genetic algorithm to find Pareto optimal solutions, and it has better performance than the VEGA (Vector Evaluated Genetic Algorithm) and the single-objective genetic algorithm.

Ishibuchi and Murata, (1998) proposed a multi-objective genetic local search algorithm on flow shop scheduling problems. A local search procedure was applied to each new solution generated by the genetic operations. They used a multi-objective weighted sum fitness function. The highest quality performance of the algorithm shows the ability of proposed algorithm to handle the non-convex feasible region in the objective space.

Bagchi, (2001) obtained Pareto optimal solutions by using metaheuristic methods. GAs and NSGA are used for sequencing jobs in a flow shop. Multi-objective production scheduling problems such as three-objective flow shops, three-objective job shops and two-objective open shop problems are explained. They demonstrated a statistical comparison between the NSGA and augmented NSGA.

Kacem, Hammadi and Borne, (2002) presented a novel approach by localization (AL) and controlled evolutionary approach CGA (generated by the first approach) to solve assignment and job shop scheduling problem. The considered objectives are minimization of the overall completion time (makespan) and the total workload of the machines.

Rajendran and Ziegler, (2003) proposed two heuristics in a static flow shop with sequence dependent setup time's jobs to minimize the sum of weighted flow time and weighted tardiness of jobs. A random search procedure and a greedy local search are used as benchmark problems to evaluate the proposed heuristic. Computationally, the proposed algorithm has better performance than benchmark procedures in both speed and effective.

Arroyo and Armentano, (2005) proposed a genetic local search algorithm for the flow shop scheduling problem. The algorithm was applied to the flow shop scheduling problem for the following two pairs of objectives: (i) makespan and maximum tardiness; (ii) makespan and total tardiness. The results show the efficiency of the proposed algorithm to find the Pareto optimal set.

Jungwattanakit et al., (2008) formulated a mathematical model to minimize the makespan and the tardy jobs for the flexible flow shop problem with unrelated parallel machines and considering setup times. Firstly, they studied several dispatching rules (constructive algorithms). Secondly, they studied GA-based algorithms as improvement algorithm. They compared the performance of the heuristics algorithms on a set of test problems up to 50 jobs and 20 stages. They found that, for population sizes, crossover types, and mutation types, there were statistically significant differences.

Yazdani, Amiri and Zandieh, (2010) proposed a PVNS (parallel variable neighborhood algorithm) that solves the FJSP (flexible job shop scheduling) to minimize makespan time. The computational results show that the proposed algorithm is a viable and effective approach for the FJSP.

Moslehi and Mahnam, (2011) proposed a new approach to solve the multi-objective flexible jobs hop scheduling problem based on a hybridization of the Particle Swarm and Local Search algorithms with different release time. They compared the proposed algorithm with other algorithms (weighted summation of objectives and Pareto approaches) to show the performance of presented algorithm.

3. PROBLEM DEFINITION AND MODELING

In this chapter, firstly a novel mixed integer multi-objective mathematical model for parallel machine scheduling problem is introduced. Next, the assumptions for the problem are presented. Finally, the comparison with other dispatching rules and the solutions for the considered problem are provided.

3.1. Problem Definition

The problem considered in this chapter regards with scheduling of unrelated parallel machine when job's processing time is dependent on the completion time of assigned machine. It is worth to mention that, the idea of the Sequence Job Minimum Completion Time (SJMCT) algorithm is associated with a common heuristic used in parallel machine scheduling the longest processing time rule (LPT) in some characteristic features.

In this study, processing times are known and deterministic. Assume that, there is limited number of jobs (2m+1) or more and each job has a single operation that can be performed on one machine only. Therefore, the problem will become an NP hard problem (Frenk and Rinnooy Kan, 1987).

Several researchers such as Tavakkoli-Moghaddam, Taheri and Bazzazi, (2008), Li et al., (2010), Li et al., (2012) and Bandyopadhyay and Bhattacharya, (2013) formulated a mathematical programming model for parallel machine scheduling problems with different assumptions. Also, Kamisli Ozturk and Sabti A.N., (2017) considered a mixed integer programming model for unrelated parallel machine scheduling problems.

In this study, the proposed algorithm Sequence Job Minimum Completion Time (SJMCT) deals with scheduling non-identical jobs $J_1, J_2,...,J_n$ on unrelated parallel machine $M_1,M_2,..,M_m$. Every job j is considered with a processing time p_{ij} and a due date d_{ij}. Let $p_{ij}= p_j$, $\forall\ i = j$ be the processing time to the first m scheduled job. The SJMCT algorithm is applied at two levels. In the first level, the new job is assigned to machine i which has the minimum completion time between the first m machines. In the second level, each job will be assigned iteratively to the machine which has the shortest completion time. The algorithm repeats the same operator to schedule all jobs to minimize the maximum completion time and the total tardiness as given in equations (3.1) and (3.2).

$$Min\ C_{max} = \max C_j, \forall\ j\ =\ 1, \dots, n \tag{3.1}$$

Where, C_j is the completion time of job j.

$$Min \sum_{j=1}^{n} T_j\ \ \forall\ j\ =\ 1, \dots, n \tag{3.2}$$

Where, T_j is the tardiness of job j and $T_j = max\ (0, Cj - d_j)$.

Furthermore, if the completion time C_j of job j is greater than its due date d_j, then this job is considered as tardy. Otherwise, the tardiness T_j of job j is equal to 0.

3.2. Assumptions

Before formulating the problem, the following assumptions are considered.

1. The machines are unrelated (the processing time of a job depends on the machine assignment).
2. The jobs are non- identical (jobs have different processing times on each machine).
3. Each machine can process only one job at a time.
4. Each machine is available at time zero.
5. Preemption and machine breakdowns are not allowed.
6. No setup time is required.

3.3. Mathematical Model of the Problem

As mentioned in Section 3.1, the two objectives are minimized simultaneously. The proposed multi-objective mathematical model for parallel machine scheduling model is proposed as follows, where;

<u>Indices and sets:</u>

n: number of jobs.

m: number of machines.

j, k : index for jobs, $j = 1, \dots, n$, $k=m+1 \in n$, $\{j{:}j=1,2,\dots,k, m+2, \dots ,n\}$.

i: index for machine, $i = 1,\dots, m$.

<u>Parameters:</u>

S_{ij}: starting time of job j at machine i. $i=j=1, \dots, m$, which equal to zero.

d_{ij}: due date of job j at machine i. $i=1 ,.., m$, $j = 1,\dots, n$.

p_{ij}: processing time of job j on machine i. $i=1, .., m$ and $j = 1,\dots, n$.

M: a great constant.

Decision variables:

C_{ij}: completion time of job *j* at machine *i*. *i* =*j*= 1, …, m.

C^*_{ij}: minimum completion time of job *j* at machine *i*, *i* =*j*= 1, …, m.

C_{ik}: completion time of job *k* at machine *i*. *i*=1, .., m, *k*=m+1.

C^*_{ik}: minimum completion time of job *k* at machine *i*. *i*=1, ..., m, *k* = m+1,…, n.

$C_{i(k+1)}$: completion time of job *k+1* at machine *i*. *i*=1, ..., m, *k*=m+1, …, n.

T_{ij}: max(*0*, C_{ij}-d_{ij}) the real tardiness of job *j*, *i*=1, ..., m, *j* = 1, …, n.

C_{max}: maximum completion time.

$$x_{ij} = \begin{cases} 1, & if \text{ job } j \text{ is assigned to machine } i \\ 0, & \text{otherwise} \end{cases}$$

$$x_{ik} = \begin{cases} 1, & if \text{ job } k \text{ is assigned to machine } i \\ 0, & \text{otherwise} \end{cases}$$

Formulated problem:

$$\text{Minimize } \left(C_{max}, \sum_{j=1}^{n} \sum_{i=1}^{m} T_{ij}\right) \quad (3.3)$$

Subject to

$$x_{ij} = 1 \qquad \forall\, i = j = 1, \ldots, m \quad (3.4)$$

$$S_{ij} = 0 \qquad \exists\ i = j = 1, \ldots, m \quad (3.5)$$

$$C_{ij} \geq S_{ij} + P_{ij}x_{ij} \qquad \exists\ i = j = 1, \ldots, m \quad (3.6)$$

Level-I:

$$C^*_{ij} = min\{C_{ij}\} \qquad \forall\, i,\ i = 1, \ldots, m \quad (3.7)$$

$$C_{ij} + M(1 - x_{ik}) \leq C^*_{ij} \quad \forall\,(i,j),\ i,j = 1, \ldots, m,\ k = m+1 \quad (3.8)$$

$$C_{ik} = C_{ij} + P_{ik}x_{ik} \qquad \forall\, i = j = 1, \ldots, m, k = m+1 \quad (3.9)$$

Level-II:

$$C^*_{ik} = min\{C_{ik}\} \qquad \forall\, i = 1, \ldots, m, \forall k = m+1, \ldots, n \quad (3.10)$$

$$C_{ik} + M\left(1 - x_{i(k+1)}\right) \leq C^*_{ik} \quad \forall\, i = 1, \ldots, m, \forall k = m+1 \quad (3.11)$$

$$C_{i(k+1)} = C_{ik} + P_{i(k+1)}x_{i(k+1)} \forall\, i = 1, \ldots, m, \forall k = m+1, \ldots, n \quad (3.12)$$

$$\sum_{i=1}^{m} x_{ik} = 1 \qquad \forall k = m+1, \dots, n \tag{3.13}$$

$$C_{max} = max\{C_{in}\} \qquad \forall\ i = j = 1, \dots, m,\ \forall k = m+1, \dots, n \tag{3.14}$$

$$T_{ij} \geq C_{ij} - d_{ij} \qquad \forall i = 1, \dots, m, \forall j = 1, \dots, n \tag{3.15}$$

$$T_{ij} \geq 0 \qquad \forall i = 1, \dots, m, \forall j = 1, \dots, n \tag{3.16}$$

$$x_{ik} = 0\ or\ 1 \qquad \forall i = 1, \dots, m, \forall k = m+1, \dots, n \tag{3.17}$$

Equation (3.3) represents the objective functions. Constraint set (3.4) assigns the first *m* jobs to *m* machines, such as 1^{st} job to 1^{st} machine, 2^{st} job to 2^{st} machine and so on. Constraint set (3.5) states that the starting times of the first *m* job on each machine equal to zero. Constraint set (3.6) relates the processing time of the first *m* job with start time. Constraint set (3.7) denotes to select the minimum completion time from the first *m* job. Constraint set (3.8) guarantees assigning k^{th} job to i^{th} machine which has minimum completion time. Constraint set (3.9) calculates the completion time for k^{th} job on machine *i*. Constraint set (3.10) selects the minimum completion time for all jobs from k^{th} to n^{th} job. Constraint set (3.11) assigns the $(k+1)^{th}$ job to the minimum completion time for all jobs from *k+1* to *n*. Constraint set (3.12) calculates the completion time from k^{th} to n^{th} job on machine *i*. Constraint set (3.13) guarantees that each job is assigned exactly to one machine. Constraint set (3.14) determines completion time as the maximum completion time of all machines. Constraint set (3.15) and (3.16) calculate the tardiness of job *j* ensure that only the positive value of lateness can be considered as tardiness. Constraint set (3.17) defines the decision variable x_{ik}, it is equal to 1 when job *k* assigned to machine *i*, 0 otherwise.

For this problem and for more clarity, the solution process can be summarized as follows:

Algorithm: Sequence Job Minimum Completion Time (SJMCT)

Step 1: Start with $2m+1$ or more jobs where m represents the number of unrelated parallel machine i = 1,…, m.

Level-I; *Starting from the first job to k^th^ job, let k*=m+1*:*

Step 2: Assign the first m jobs to machines respectively set i=j=1,…, m.

Step 3: Compute the minimum completion time (release date + processing time) for the first m job$\left(C_{ij}^{*}\right)$.

Step 4: Assign job k to machine which has the minimum job completion time.

Step 5: Update the completion time of job k, and go to step 6.

Level-II; *Starting from job k, where* (*k*=m+1,…, n)*:*

Step 6: Select the new minimum completion time$\left(C_{ik}^{*}\right)$.

Step 7: Assign the unscheduled job $k+1$ to machine has minimum job completion time.

Step 8: Compute the total completion time and repeat level-II in the same way until all jobs are scheduled.

The illustrative representation of SJMCT is given in Figure 3.1.

Jobs / Machines	J_1	J_2	J_3	J_m	J_k	J_{k+1}	…	J_n
M_1	C_{11}				C_{1k}^{*}	$C_{1(k+1)}$	…	C_{1n}
M_2		C_{22}^{*}			C_{2k}		…	C_{2n}
M_3			C_{33}				…	C_{3n}
M_4				C_{4m}			…	C_{4n}

Figure 3. 1. *The representation of SJMCT algorithm*

3.4. The Comparison of the SJMCT Algorithm with other Algorithms

In order to evaluate the performance of proposed algorithm SJMCT it compared with Balin's (2011) test problems and other dispatching rules (as given in chapter 2). The comparison with respect to one objective function represented by minimize the maximum completion time (makespan). In general, a formulation of the problem uses "binary" variables x_i where, (i=l,..., m; j=l,..., n), as follows:

$$x_{ij} = \begin{cases} 1 & if\ job\ j\ is\ assigned\ to\ machine\ i \\ 0 & otherwise \end{cases}$$

The positive variable C_{max} represents the maximum completion time and x_{ij} refers to assignment variables. The problem can be written as (Potts, 1985):

$$\text{Minimize} \quad C_{max} \tag{3.18}$$

Subject to

$$\sum_{j=1}^{n} P_{ij} x_{ij} \leq C_{\max} \qquad i = 1, \dots, m \tag{3.19}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad j = 1, \dots, n \tag{3.20}$$

$$x_{ij} \in \{0, 1\} \qquad i = 1, \dots, m\ ;\ j = 1, \dots, n \tag{3.21}$$

Constraint (3.19) ensures that C_{max} is at least as large as the total processing time on any machine, while constraints (3.20) and (3.21) ensure that each job is processed on exactly one machine.

Comparisons for some dispatching rules and the analysis of the results are given in the following subsections.

3.4.1. Scheduling with LPT Balin's rule

The scheduling problem solved by Balin, (2011) using LPT dispatching rule. The set data indicates to the processing times for nine jobs and four unrelated parallel machines are given at Table 3.1.

Table 3. 1. *Processing time of the jobs (Balin, 2011)*

Processing time (min)	**Job 1**	**Job 2**	**Job 3**	**Job 4**	**Job 5**	**Job 6**	**Job 7**	**Job 8**	**Job 9**
Machine 1	18	14	24	30	16	20	22	26	14
Machine 2	9	7	12	15	8	10	11	13	7
Machine 3	4.5	3.5	6	7.5	4	5	5.5	6.5	3.5
Machine 4	3.6	2.8	4.8	6	3.2	4	4.4	5.2	2.8

The obtained scheduling problem of Blain LPT dispatching rule are given in Table 3.2.

Table 3. 2. *Scheduling with LPT Balin's rule*

Machines	Scheduled job				C_i
M.1	Job 2				14.00
M.2	Job 7				11.00
M.3	Job 8	Job 6	Job 5		15.50
M.4	Job 4	Job 3	Job 1	Job 9	**17.20**

3.4.2. Scheduling with Balin (GAs)

Genetic algorithms (GAs) are adaptive heuristics search algorithm based on the concepts of natural genetics and natural selection theories proposed by Charles Darwin. In this algorithm the population is defined to be the collection of all the chromosomes. Each chromosome represents a possible solution to the optimization problem, often using strings of 0's and 1's as seen in Figure 3.2. Each bit typically corresponds to a gene. The value for a given gene is called alleles (Mishra and Patnaik, 2009).

Figure 3. 2 *Representation of chromosome*

The same scheduling problem is solved with GAs (Balin, 2011). A randomly generated population of 10 chromosomes is solved by using "work center". Several iterations are used to solve the problem and each iteration is provides one solution. The best solutions are given in 12 different schedules. The scheduling results and the minimum completion time at iterations 720 are given in Table 3.3

Table 3. 3. *Scheduling with GAs at iteration 720*

Machines	Scheduled job			C_i
M.1	Job 2			14.00
M.2	Job 1	Job 9		**16.00**
M.3	Job 5	Job 7	Job 3	15.50
M.4	Job 6	Job 8	Job 4	15.20

3.4.3. Scheduling with longest processing time dispatching rule (LPT)

A common heuristic used in parallel machine scheduling is the LPT rule. In parallel machine scheduling environments $P_m//C_{max}$, as Hong, Hang and Yu (1998) mentioned jobs are arranged in decreasing order with respect to the processing times, such that $p_1 \geq p_2 \geq ... \geq p_n$. At time t = 0, in this rule the jobs having large values of processing time are given high priority for scheduling on the parallel machine. The results of Balin's scheduling problem are resolved with LPT rule as given in Table 3.4.

Table 3. 4. *Scheduling with LPT rule*

Machines	Scheduled job			C_i
M.1	Job 4	Job 2	Job 9	**58.00**
M.2	Job 8	Job 5		21.00
M.3	Job 3	Job 1		10.50
M.4	Job 7	Job 6		8.40

3.4.4. Scheduling with shortest processing time dispatching rule (SPT)

In SPT dispatching rule, the job with the shortest processing time is chosen fist for processing (Jungwattanakit et al., 2008). The same test problem is solved again according to SPT rule. The obtained schedule is given in Table 3.5.

Table 3. 5. *Scheduling with SPT rule*

Machines	Scheduled job			C_i
M.1	Job 2	Job 8		**30.00**
M.2	Job 9	Job 3		19.00
M.3	Job 5	Job 7		9.50
M.4	Job 1	Job 6	Job 4	13.60

3.4.5. Scheduling with sequence job minimum completion time (SJMCT)

The proposed algorithm SJMCT with the same parameters given in Table 3.1 is solved by GAMS v. (24.5.6) optimization software and CPLEX solver. The obtained schedule and the scheduling chart of the algorithm are represented in Table 3.6 and Figure 3.3.

Table 3. 6. *Scheduling with sequence job minimum completion time algorithm (SJMCT)*

Machines	Scheduled job			C_i
M.1	Job 1			**18.00**
M.2	Job 2	Job 7		**18.00**
M.3	Job 3	Job 5	Job 8	16.50
M.4	Job 4	Job 6	Job 9	12.80

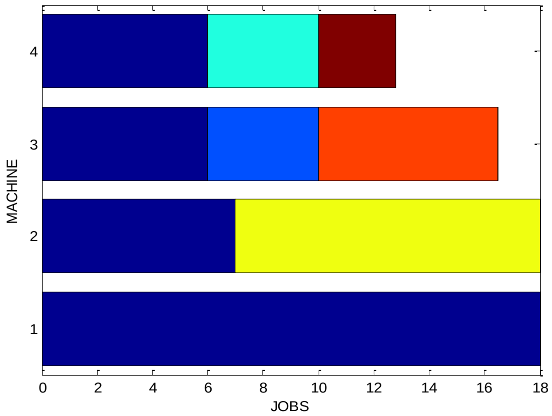


Figure 3. 3. *The scheduling chart of SJMCT algorithm*

3.4.6. The computational results and comparisons

The proposed algorithm is compared with all algorithms mentioned in Section 3.4. The computational results to Balin's problem with nine jobs which represented by the total and the maximum completion time and for each machine are given in Table 3.7.

Table 3. 7. *The total and maximum completion time for all comparison algorithms*

Machines	Completion time C_i				
	Balin LPT	**Balin GA**	**LPT**	**SPT**	**SJMCT**
M_1	14.00	14.00	**58.00**	**30.00**	**18.00**
M_2	11.00	**16.00**	21.00	19.00	**18.00**
M_3	15.50	15.50	10.50	9.50	16.50
M_4	**17.20**	15.20	8.40	13.60	12.80

As given in this table, the maximum completion time is equal to 17.20 at machine (4) in Balin's LPT rule, equal to 16 at machine (2) in Balin's GAs, equal to 58 at machine (1) in LPT dispatching rule, equal to 30 at machine (1) in SPT dispatching rule and equal to 18 at machine (1) and (2).

Among all the results obtained from Balin's test problems, the SJMCT algorithm is better than LPT and SPT dispatching rule because it has the smallest value of maximum completion time. Furthermore, SJMCT algorithm has more convergence as compared with other algorithms in computing the total completion time of each machine. That means, it gives a good assignment of jobs at the machines and it make a good balance in workload over the parallel machines. In addition, in SJMCT algorithm there is no order forced to submit certain job.

The dispatching rule mentioned before are easy to solve small size problems with one objective and it require little computer time. Moreover, it can't guarantee the optimal solution. For all these reasons, novel heuristic algorithms are proposed to solve large size and multi-objective parallel machine scheduling problems.

4. NOVEL MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

As given in the literature review section, multi-objective evolutionary algorithms (MOEA) are performed to solve multi-objective parallel machines scheduling problems. In this section two hybrid multi-objective evolutionary algorithms are proposed based on SJMCT algorithm.

4.1. Multi-objective Optimization

Many real-life optimization problems are actually multi-objective because they involve more than one objective. The solutions of multi-objective problems can provide deeper insights to the decision maker than those of single-objective problems. A multi-objective optimization problem (MOP) can be formulated to find the best solution under multiple objective functions each is either maximized or minimized. As in the single objective optimization problems, there may be some constraints that must be satisfied. In its general form, a multi-objective optimization problem can be formulated as follows (Kasimbeyli et al., 2015):

$$\min_{x \in X} [f_1(x), \ldots, f_n(x)]$$

Where X is a nonempty set of feasible solutions and $f_i: X \to R, i = 1, \ldots, n$ is real-valued functions. Let $\left(f_1(x), \ldots, f_n(x)\right)$ for every $x \in X$ and let $Y \coloneqq f(X)$.

For a nontrivial multi-objective optimization problem, there is not exist single solution that simultaneously optimizes each objective. Also, there exist a (possibly infinite) number of Pareto optimal solutions. In that case, a solution is called **non-dominated.** In the same way**,** (Ehrgott, 2006) introduced the idea of dominance as follows:

Definition 4.1. A feasible solution $\hat{x} \in X$ is called efficient or Pareto optimal, if there is no other $x \in X$ such that $f(x) \leq f(\hat{x})$. If $\hat{x}$ is efficient, $f(\hat{x})$ is called non-dominated point. If $x^1, x^2 \in X$ and $f(x^1) \leq f(x^2)$ we say x^1 dominates x^2 and $f(x^1)$ dominates $f(x^2)$. The set of all efficient solutions $\hat{x} \in X$ is denoted X_E and called the efficient set. The set of all non-dominated points $\hat{y} = f(\hat{x}) \in Y$, where $\hat{x} \in X_E$, is denoted Y_N and called the non-dominated set.

The definition of dominated and don-dominated solutions can also illustrate as follows (Ehrgott, 2006).

- **Domination:** A solution is said to be dominate another if it is better in **all objectives.**
- **Non-Domination:** A solution is said to be non-dominated if it is better than other solutions **in at least one objective.**

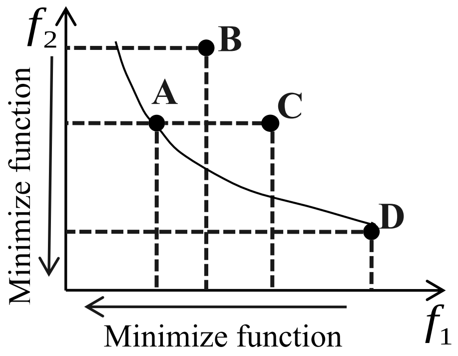


Figure 4. 1. *Non-dominated and dominated solution*

- ❖ *A* dominates *B* (better in both f_1 and f_2)
- ❖ *A* dominates *C* (same in f_1 but better in f_2)
- ❖ *A* does not dominate *D* (non-dominated points)
- ❖ *A* and *D* are in the "Pareto optimal front"
- ❖ These non-dominated solutions are called Pareto optimal solutions.
- ❖ This non-dominated curve is said to be Pareto front.

Before 1995, the conventional techniques such as linear programming, dynamic programming and nonlinear programming are the main approaches to solve multi and bi-objective problems (Reddy and Kumar, 2007). However, these methods can only solve the small size problems. The evolutionary algorithms have become the main path to solve multi-objective scheduling problems since 1995 (Lei, 2009). Non-dominated sorting genetic algorithm (NSGA), Strength Pareto evolutionary algorithm (SPEA), ant-colony optimization (ACO) and particle swarm optimization (PSO) are some examples of multi-objective evolutionary optimization algorithms.

4.2. Non-dominated Sorting Genetic Algorithm II (NSGA-II)

The Non-dominated Sorting Genetic Algorithm II (NSGA-II) introduced by (Srinivas and Deb, 1994) is an evolutionary multi-objective solution approach used to improve the adaptive fit of a population of candidate solutions to a Pareto front constrained by a set of objective functions. NSGA-II is an extension of the Genetic Algorithms for multi objective problems. It has a better sorting algorithm, incorporates elitism and no sharing parameter need to be chosen a priori (Seshadri, 2006).

Refers to (Srinivas and Deb, 1994), the selection procedure of NSGA-II orders the population into a hierarchy of non-dominated Pareto fronts. Also, sorts the solution by rank and crowding distance then, ranks the non-dominated front of level1 is constituted and includes all the non-dominated solutions. As (Godinez, Espinosa and Montes, 2010) and (Yusoff, Ngadiman and Zain, 2011) described the crowding distance is a measure of how close the solution to its neighbors. Large average crowding distance will result in a better diversity in the population (Seshadri, 2006). Here, the calculation of this quantity in Figure 4.3 and equations (4.1) and (4.2).

Two genetic operators' crossover and mutation with selection operator are used to update the current population and create a new population. The crossover operator combines two solutions (parents) to create two new solutions (children) that may be better than both of the parents. For crossover operators, the binary crossover (Memari et al., 2016) is used. Moreover, mutation operator is an important part of the evolution principle used to add diversity into current population and helps to escape from local optimal to enhance the algorithm and to find better solutions (Fallah-Mehdipour et al., 2012).

4.3. SJMCT- Based NSGA-II (SJMCT -NSGA-II Algorithm)

Non-dominated Sorting Genetic Algorithm (NSGA-II) is combined with proposed SJMCT algorithm to create one unified population able to represent the best possible solutions for multi-objective parallel machine scheduling problem.

The procedure of SJMCT-NSGA-II can be described as follows, where t represents number of generations:

1. Generate uniform random processing time P_t and due date D_t.
2. Evaluate the objective function values based on SJMCT constraints.
3. Initialize the population of NSGA-II algorithm randomly and evaluate the objective function values of SJMCT algorithm.
4. Create Q_t (offspring) with the operators of selection, crossover and mutation.
5. Evaluate the solutions.

6. Combine populations P_t and Q_t to create new population R_t of size $2N$.
7. Sort the solutions of R_t in different non dominated front.
8. In the new population P_{t+1} add the best solutions (the best front and the best value of the crowding distance). Use non-dominated and crowding distance equations (4.1) and (4.2) to fulfill the new generation if the number of these solutions is less than the population size.

The crowding distance represents the average distance of two solutions on either side of solutions i along each of the objectives to get an estimate of the density of solutions surrounding a particular solution i in the population (Chand and Mohanty, 2013).

$$C.D(i) = \sum_{j=1}^{n} \left| \frac{f_j(i+1) - f_j(i-1)}{f_j(max) - f_j(min)} \right| \tag{4.1}$$

$$C.D\left(f_j(max)\right) = C.D\left(f_j(max)\right) = \infty \tag{4.2}$$

Figure 4.2 represents the crowding distance calculation as follows:

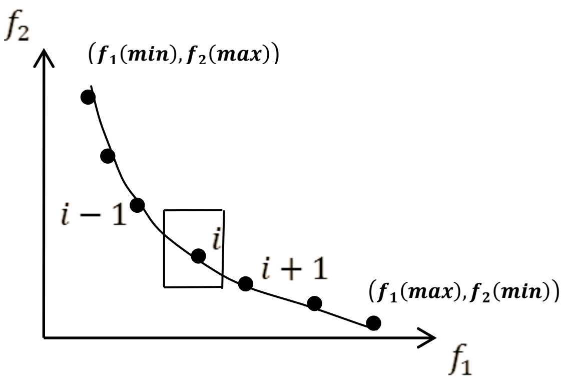


Figure 4. 2. *Crowding distance calculation*

Where, $f_j(max)$: The maximum value of objective j.

$f_j(min)$: The minimum value of objective j.

j= 1, 2,…, n numbers of objective functions.

The crowding tournament selection operator is a measure that guides the selection process at the various stages of the algorithm toward Pareto optimal front, when the following conditions are true:

- If rank i < rank j , (i has a better rank).

- If rank i = rank j but $C.D_{(i)} > C.D_{(j)}$, (i has a better crowding distance).

9. Repeat the steps 4-6 till the maximum number of generation is reached.

A schematic representation of the NSGA-II procedure is given in Figure 4.3.

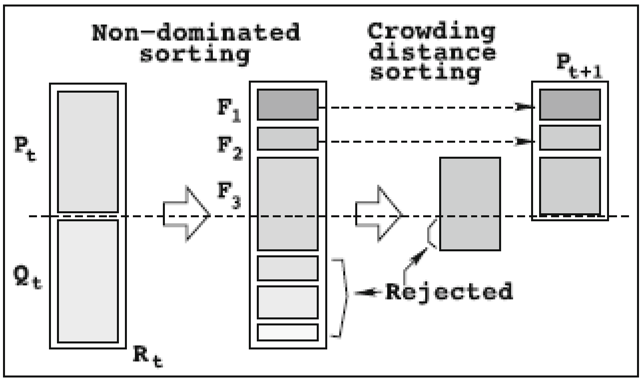


Figure 4. 3. *Schematic representation of the NSGA-II procedure (Wang 2011)*

Crossover and mutation schemes that were developed by (Deb et al., 2000) are employed. The crossover operator used in this study can be seen in the following equations:

$$x_1^{child} = \frac{1}{2}\left[(1+b) * x_1^{parent} + (1-b) * x_2^{parent}\right] \quad (4.3)$$

$$x_2^{child} = \frac{1}{2}\left[(1-b) * x_1^{parent} + (1+b) * x_2^{parent}\right] \quad (4.4)$$

Where:

$$b = \begin{cases} (2 * r)^{\left(\frac{1}{\mu+1}\right)} & if \quad r \leq 0.5 \\ \left(\frac{1}{2*(1-r)}\right)^{\left(\frac{1}{\mu+1}\right)} & if \quad r > 0.5 \end{cases} \quad (4.5)$$

b: difference between the objective function values of parents and children.
μ: a constant which shows the difference between the objective function values of parents and children; a large value of μ gives a higher probability for creating near-parent solutions. r: a random value in [0, 1].

The mutation operator is also applied as seen in equations (4.6) and (4.7).

$$d = \begin{cases} (2 * r)^{\left(\frac{1}{\eta+1}\right)-1} & if \quad r \leq 0.5 \\ \left(1 - \left(2 * (1-r)\right)\right)^{\left(\frac{1}{\eta+1}\right)} & if \quad r > 0.5 \end{cases} \quad (4.6)$$

Where: r: is a random value in [0, 1]

η: distribution constant of mutation

d: mutation value. This parameter is added to the parent gene value, as given in equation (4.7).

$$x^{child} = x^{parent} + d \qquad (4.7)$$

The flow chart of SJMCT-NSGA-II is given in Figure 4.4.

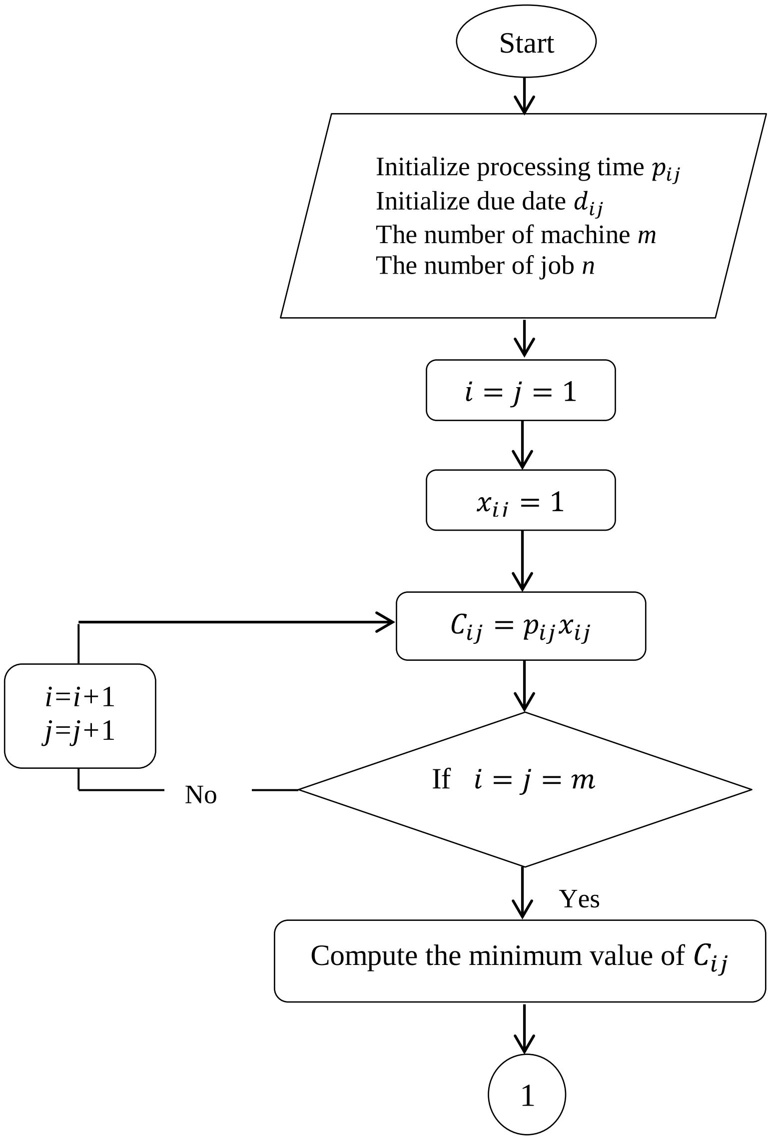


Figure 4.4. *Flow chart of SJMCT-NSGA-II*

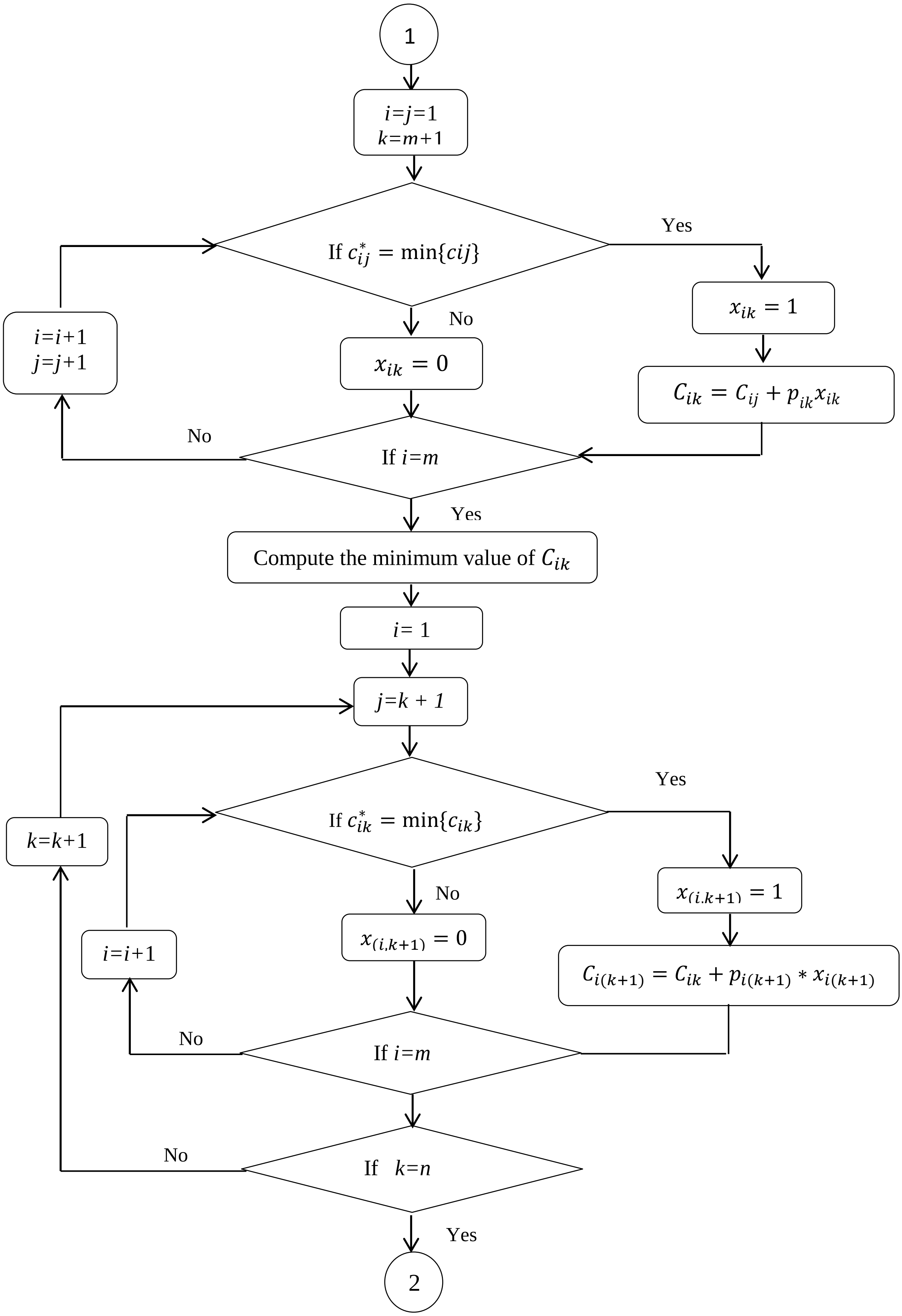


Figure 4.4. (Continue) *Flow chart of SJMCT-NSGA-II*

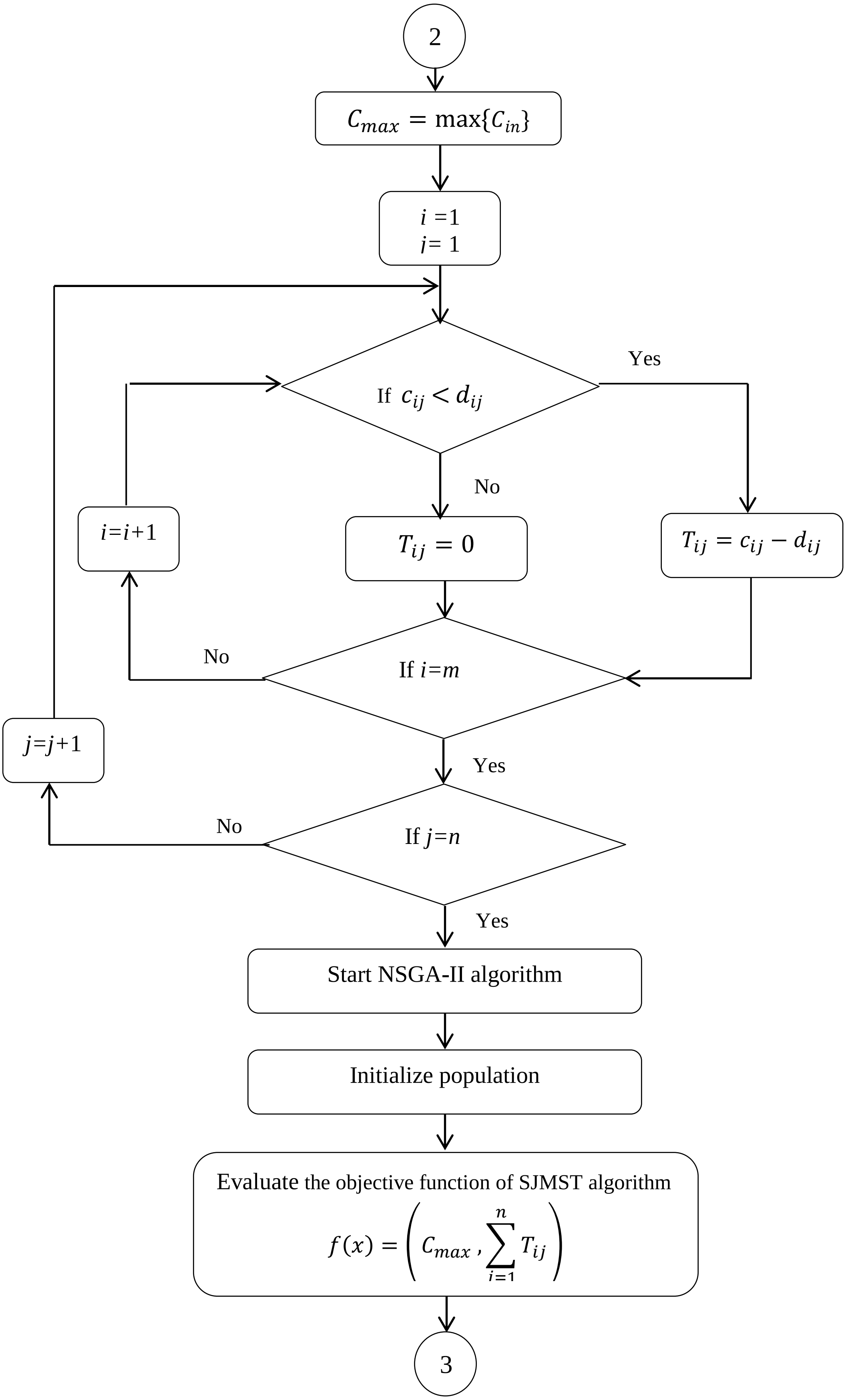


Figure 4.4. (Continue) *Flow chart of SJMCT-NSGA-II*

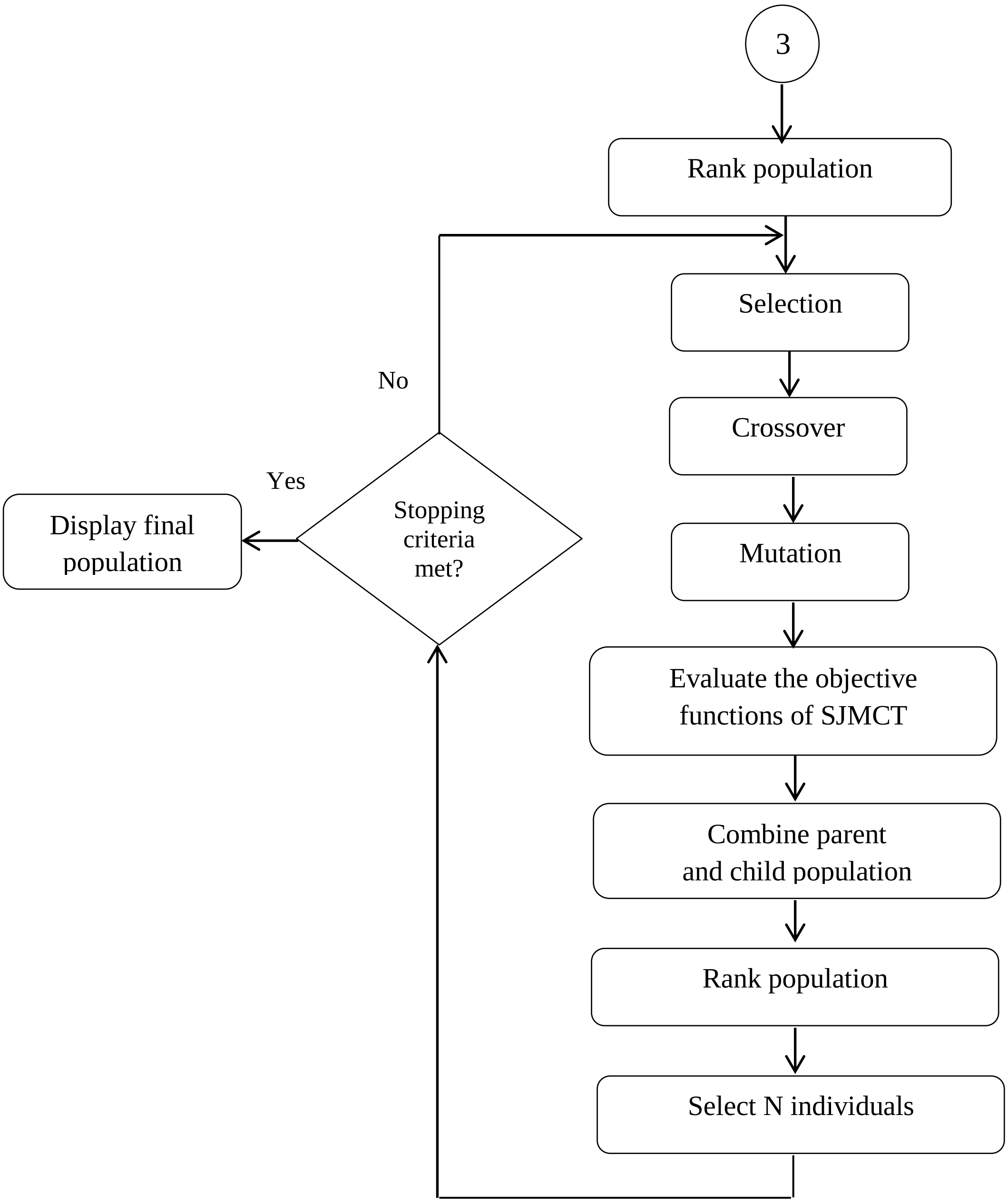


Figure 4. 4. (Continue) *Flow chart of SJMCT-NSGA-II*

4.4. Strength Pareto Evolutionary Algorithm II (SPEA-II)

Strength Pareto Evolutionary Algorithm (SPEA-II) is an extension of the Genetic Algorithms for multi objective problems. It has been proposed by (Zitzler, Laumanns and Thiele, 2001). Generally, SPEA-II algorithm uses a regular population and archive (external set) to find Pareto optimal set. It is used as an evolutionary algorithm to locate and maintain a set of Pareto optimal solutions.

The algorithm started with an initial population and an empty archive. The raw fitness function represents the summation of the strength values of its dominators in both archive and population. The density function as given in equation (4.11) estimates the density of an area of the Pareto front. The candidate population with the best remaining (non-dominated solution) fills the new archive in order to fitness. It removes the smallest distance values in the archive population by using truncated procedure. It selects the parents from a population using binary tournament selection to fill the archive population. The two genetic operators, crossover and mutation as represented in equations 4.3-4.6.

4.5. SJMCT- Based SPEA-II (SJMCT- SPEA-II Algorithm)

Strength Pareto Evolutionary Algorithm (SPEA-II) is an elitist evolutionary algorithm. The proposed SJMCT algorithm is combined with the mean process of SPEA-II as follows:

1. **Input:** n (number of jobs), m (number of machines), $\bar{N}$ (archive size), T (maximum number of generation).
2. **Initialization-I:** At first generation t =0, use the uniform random to initialize the processing time P_0 and due date D_0 for SJMCT algorithm.
3. **Initialization-II:** Initialize the population of SPEA-II to evaluate the objective function values of SJMCT algorithm and create the empty archive $\bar{P}_0 = \emptyset$.
4. **Fitness assignment:** for each individual i in the archive $\bar{P}_t$ and the population P_t there is $S(i)$ called the **strength Pareto- solution** which represents the number of dominated solution:

$$S(i) = |\{j | j \in P_t + \bar{P}_t \wedge i \succ j \}| \tag{4.8}$$

Where: the symbol + represents multi set union, the symbol $\succ$ corresponds to the Pareto dominance relation, the symbol $\wedge$ means AND (Gharari et al., 2016).

For SPEA-II, fitness $F(i)$ is defined by equation (4.9).

$$F(i) = R(i) + D(i) \tag{4.9}$$

The raw fitness function $R(i)$ of an individual *i* is calculated by the following equation:

$$R(i) = \sum_{j \in Pt+\bar{P}_t, j \succ i} S(j) \tag{4.10}$$

Here it is important to note that, fitness is to be minimized, i.e., $R(i) = 0$ corresponds to a non-dominated individual. The additional density information is incorporated to discriminate between individuals having same raw fitness, where the density at any point is a (decreasing) function of the distance to the k^{th} nearest data point. To be more precise, for each individual *i* the distances (in objective space) to all individuals *j* in archive and population are calculated and stored in a list. After sorting the list in increasing order, the k^{th} element gives the distance denoted as σ_i^k, the density function is defined by:

$$D(i) = \frac{1}{\sigma_i^k + 2} \tag{4.11}$$

Where: σ_i^k represents the objective-space distance between the i^{th} and k^{th} nearest neighbors and $k = \sqrt{N + \bar{N}}$ in equation (4.11).

5. **Environmental selection:** In this operator, all non-dominated solutions are copied from population and archived to the archive of new iteration $\bar{P}_{t+1}$. If the archive is too small $|\bar{P}_{t+1}| < \bar{N}$ then $\bar{P}_{t+1}$ is filled with best dominated solutions from P_t and $\bar{P}_t$. Otherwise, if the archive is too large $|\bar{P}_{t+1}| > \bar{N}$ an **archive truncation procedure is used** until $|\bar{P}_{t+1}| = \bar{N}$. Here, at each iteration individual *i* is chosen for removal for which *i*, $i \leq_d j$ for all $j \in \bar{P}_{t+1}$ with $i \leq_d j \;:\Leftrightarrow\; \forall\, 0 < k < |\bar{P}_{t+1}| \;:\; \sigma_i^k = \sigma_j^k \;\vee$

$$\exists\, 0 < k < |\bar{P}_{t+1}| : \left[\left(\forall\, 0 < l < k \;:\; \sigma_i^l = \sigma_j^l\right) \wedge \sigma_i^k < \sigma_j^k\right] \tag{4.12}$$

In equation (4.12), *i* and *j* are the individuals, and also $i \leq_d j$ means that individual *i* dominated individual *j* and σ_i^k denotes the distance of *i* to its k^{th} nearest neighbor in $\bar{P}_{t+1}$. In other words, at each iteration, the individual which has the minimum distance to another individual is chosen (a connection is broken by considering the second smallest distances and so forth), as given in Figure 4.5.

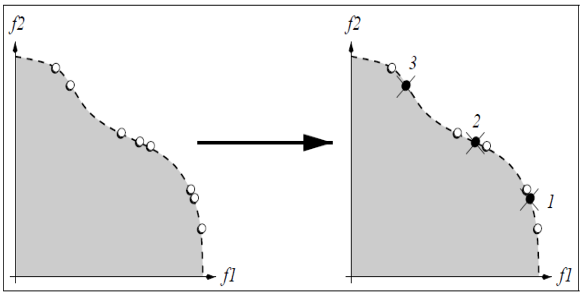


Figure 4. 5. *Illustration of the archive truncation method used in SPEA-II*

On the right, a non-dominated set is shown. On the left, it is depicted which solutions are removed in which order by the truncate operator (assuming that $\bar{\boldsymbol{N}} = \mathbf{5}$) (Zitzler, Laumanns, and Thiele, 2001)

6. **Termination:** If $t \geq T$ then the archive members $\bar{P}_{t+1}$ presented as a Pareto set, otherwise go to step 3.
7. **Mating selection:** In order to fill the mating pool use binary tournament selection with replacement on $\bar{P}_{t+1}$.
8. **Variation:** Apply mutation and crossover operators to the mating pool and fill P_{t+1} with the generated solutions. Set $t=t+1$ and go back to step 4.

The flow chart of SJMCT-SPEA-II is given in Figure 4.6.

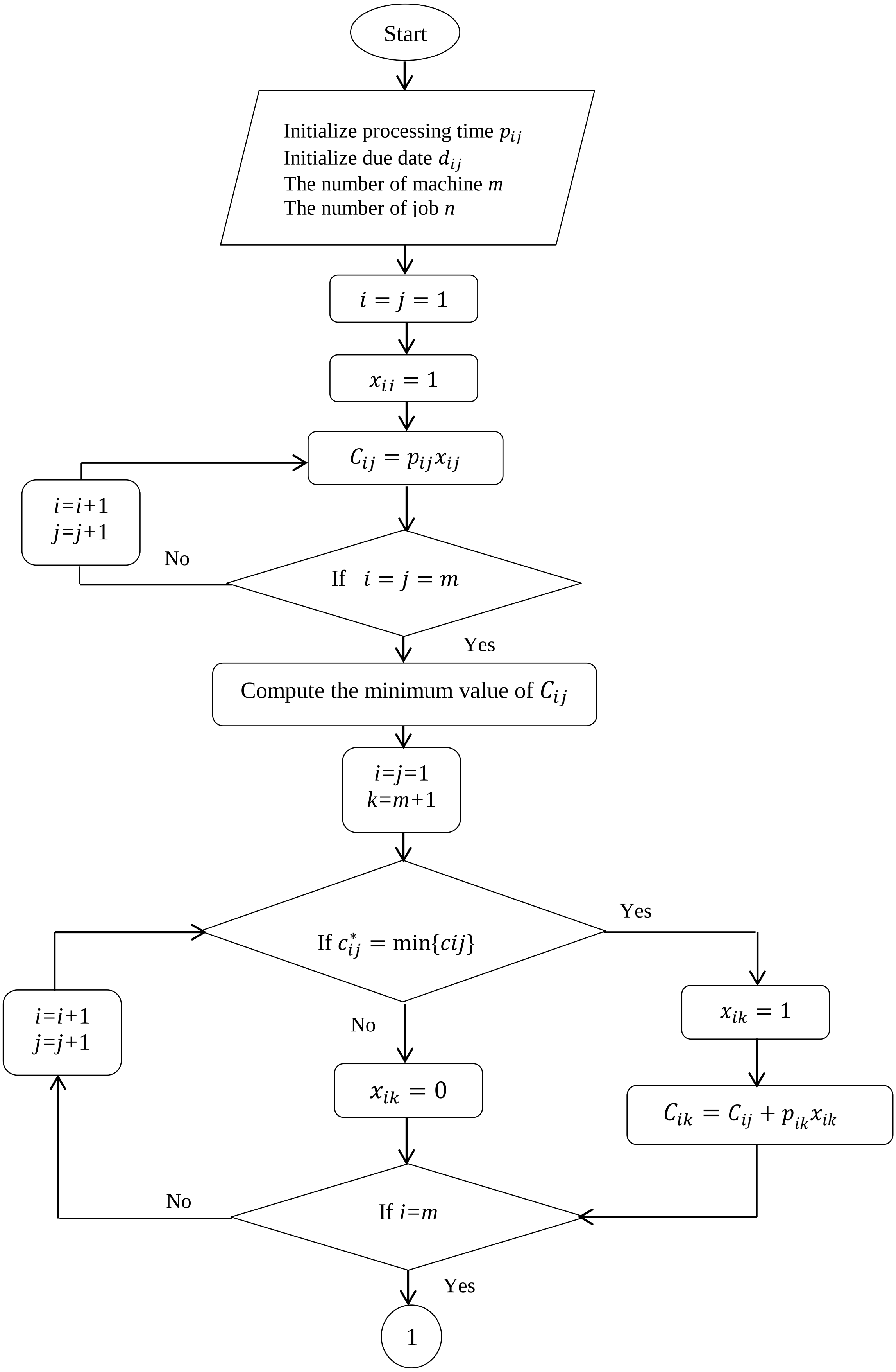


Figure 4.6. *Flow chart of SJMCT-SPEA-II*

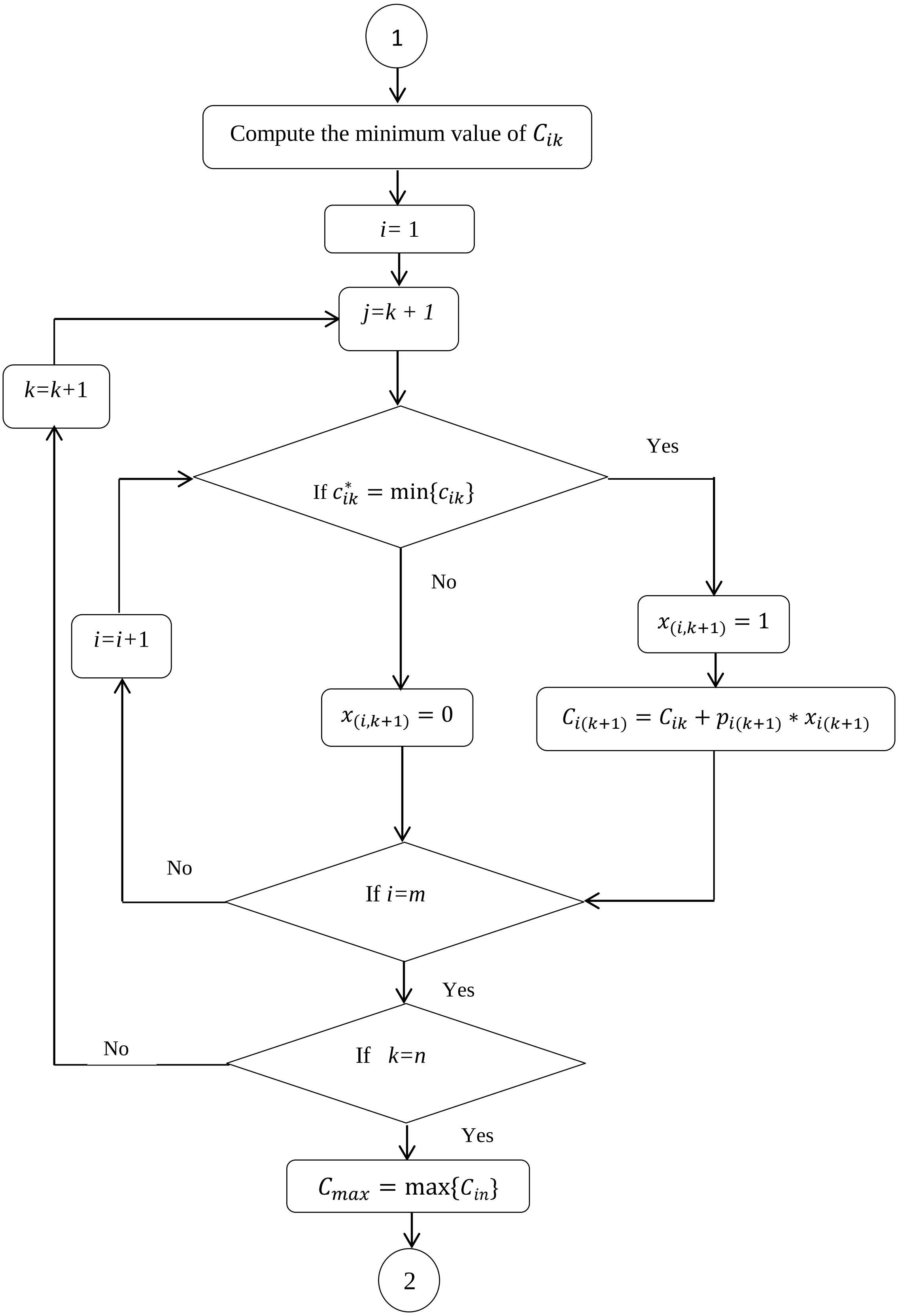


Figure 4.6. (Continue) *Flow chart of SJMCT-SPEA-II*

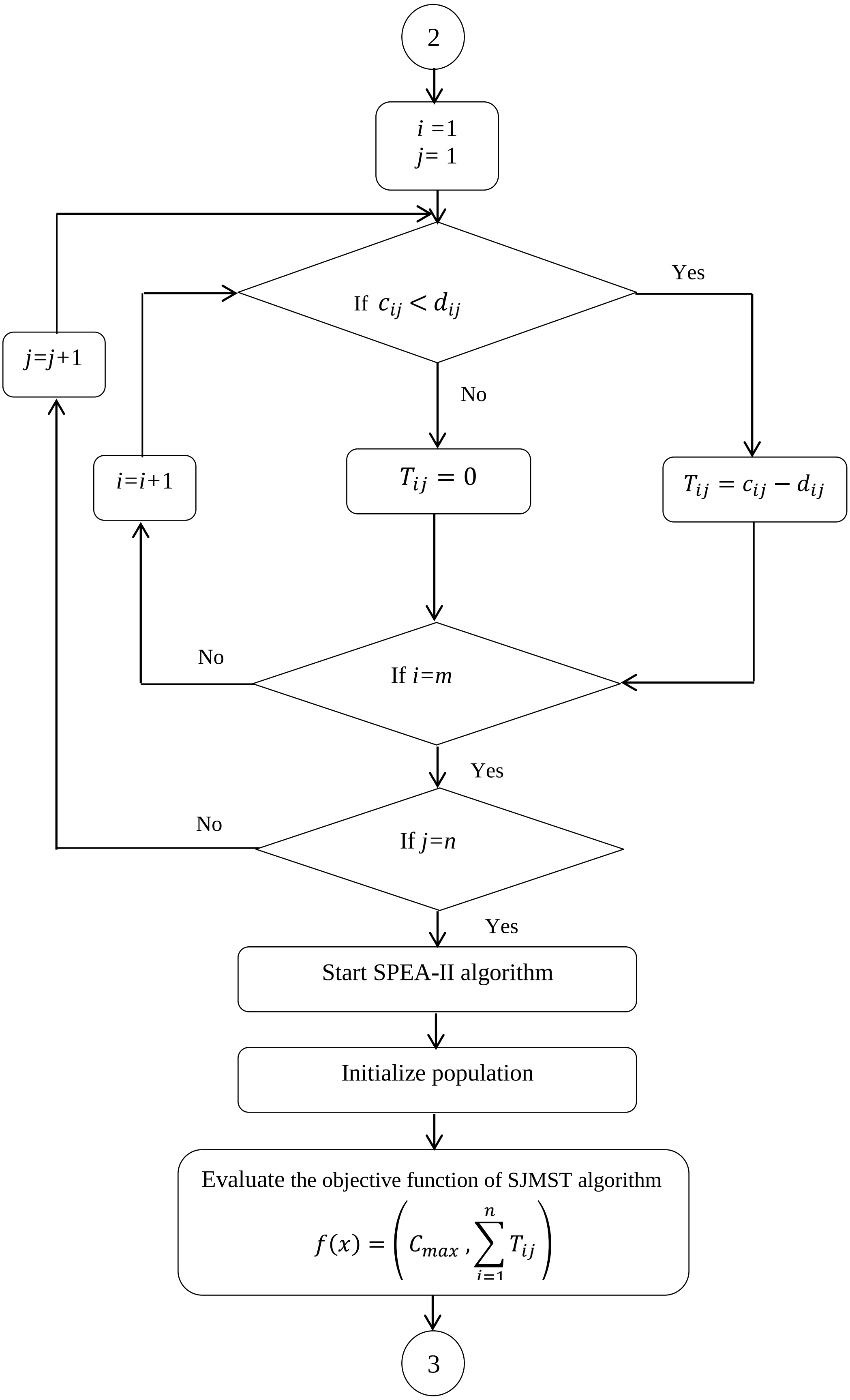


Figure 4.6. (Continue) *Flow chart of SJMCT-SPEA-II*

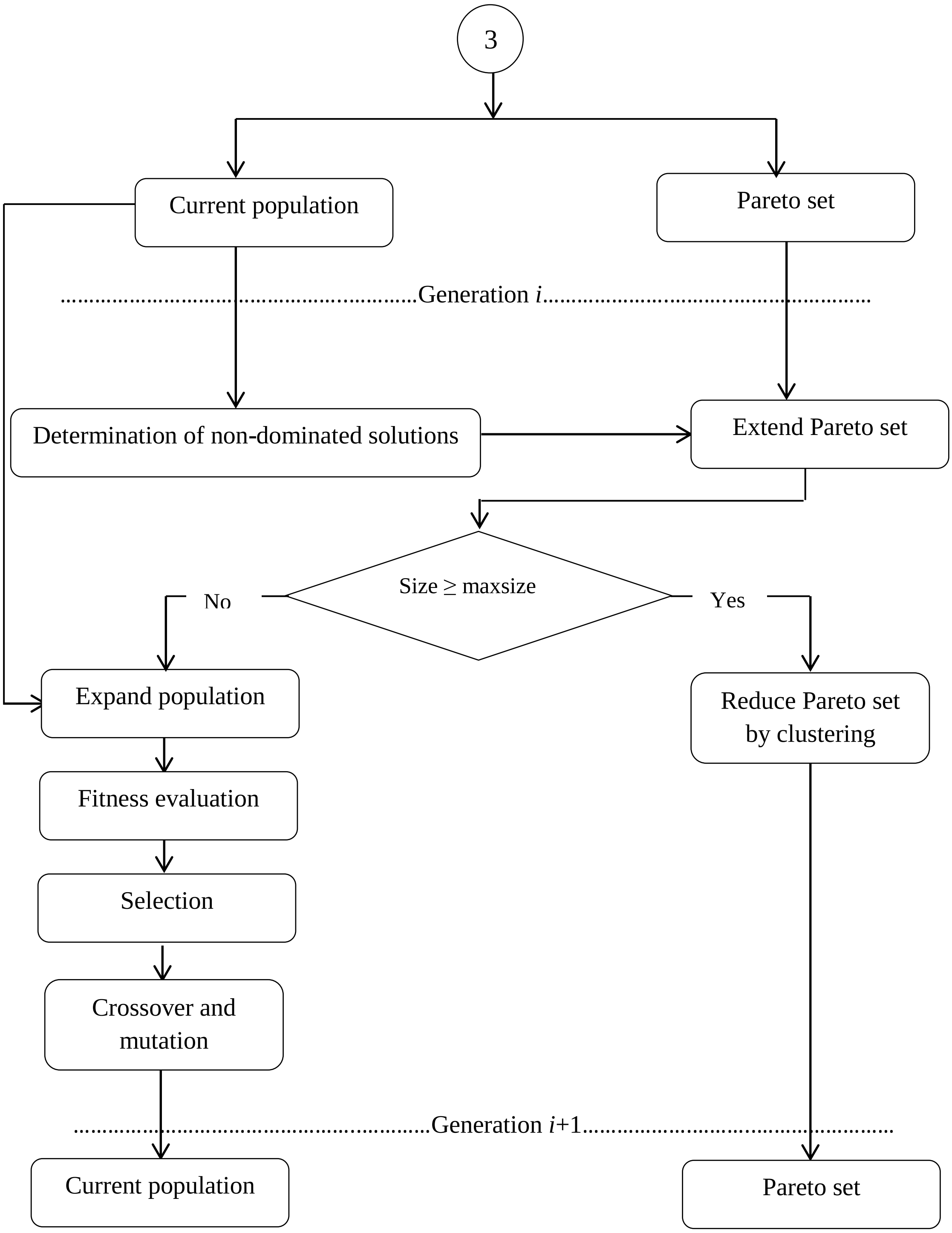


Figure 4. 6. (Continue) *Flow chart of SJMCT-SPEA-II*

5. COMPUTATIONAL RESULTS

In this section, different parameter values are considered to simulate different cases and to analyze the performances of the proposed algorithms SJMCT-NSGA-II and SJMCT-SPEA-II. For five parallel machines the first test problems is described with 60 jobs and different generation numbers. Thereafter, the second test problems are described with generation 500 and different number of jobs. The Pareto-optimal front are represented to minimize the two criteria scheduling problems, the makespan which represents the completion time of the final job and the total tardiness which represents the sum of tardiness of every job.

5.1. Experimental Design

The processing times and due dates of jobs are generated uniformly between 1 and 20, the population size equals to 100 in each algorithm. Different crossover probabilities (0.6, 0.7, 0.8 and 0.9) and mutation probabilities (0.4, 0.3, 0.2 and 0.1) are used in these tests. In particular, the experiments are designed to test the performance of the proposed algorithms by changing the parameters. The algorithms are tested firstly with 60 jobs and different generation numbers (40, 100, 300 and 500). Secondly, the algorithms are tested with different number of jobs (20, 60 and 100) and number of generation equals to 500. Table 5.1 describes the couple of different parameters setting on both algorithms SJMCT-NSGA-II and SJMCT-SPEA-II in order to show the final Pareto behavior after changing the parameters. In all cases, the number of archive used in SJMCT-SPEA-II algorithm is equal to 60. Moreover, the lower and upper bounds are selected between [-15, 15].

Table 5. 1. *Parameters used for each algorithm*

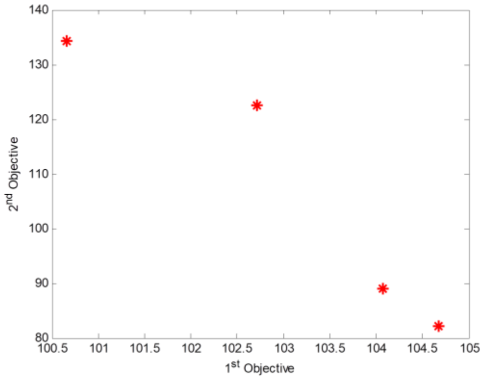
Var Min	Var Max	nArchive	nPop	Var Size [Machine Job]	Generation Numbers	Crossover Probability	Mutation Probability
-15	15	60	100	[5 20]	40	0.6	0.4
				[5 60]	100	0.7	0.3
				[5 100]	300	0.8	0.2
					500	0.9	0.1

5.2. Computational Results

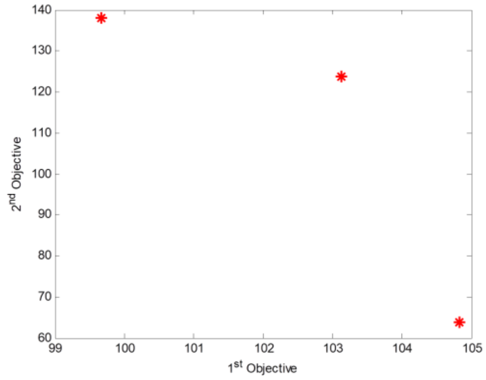
In this subsection, scheduling problem with 5 parallel machines, 60 jobs and with the parameters given in Table 5.1 is considered. In the first test problems, multiple cases study the effect of increasing the generation numbers from 40 to 500. All test problems for the proposed algorithms are implemented by MATLAB programming Version 8.3.0.532 (R2014a). Figures 5.1-5.4 depict the simulation results obtained by SJMCT-NSGA-II algorithm. Figures 5.5-5.9 give the Pareto solutions obtained by SJMCT-SPEA-II algorithm. In each test the crossover probabilities are 0.6, 0.7, 0.8 and 0.9 respectively.

5.2.1. Computational results for SJMCT-NSGA-II algorithm

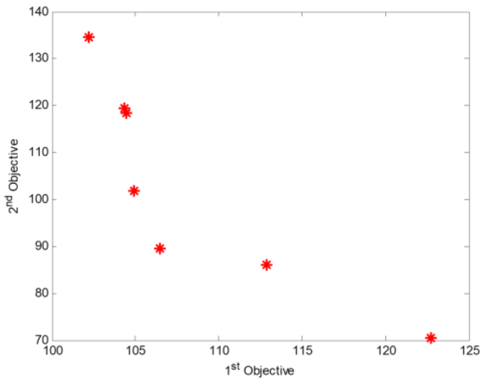
Test 1: In the first test problems for 60 jobs, the best solution is obtained for SJMCT-NSGA-II algorithm at generation 40 with number of population 100 and with crossover probabilities 0.6, 0.7, 0.8 and 0.9.



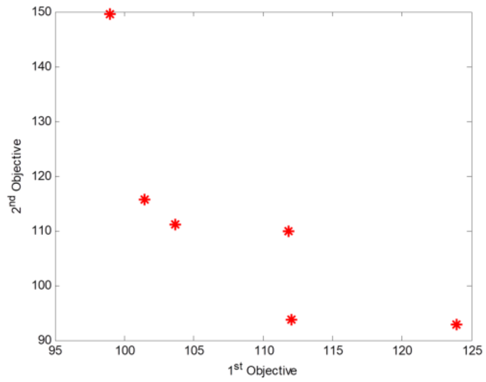
(a) Crossover Probability 0.6



(b) Crossover Probability 0.7



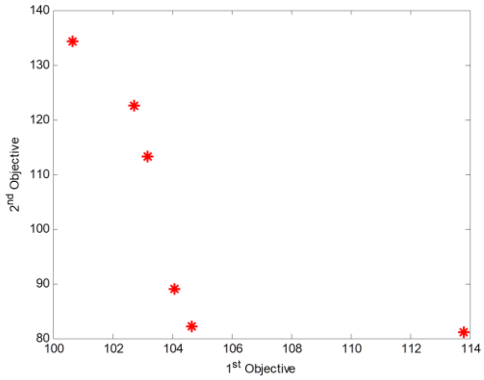
(c) Crossover Probability 0.8



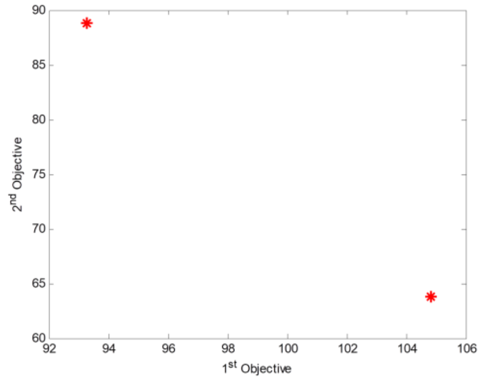
(d) Crossover Probability 0.9

Figure 5. 1. *Pareto optimal solutions for SJMCT- NSGA-II with generation 40 and different crossover probabilities*

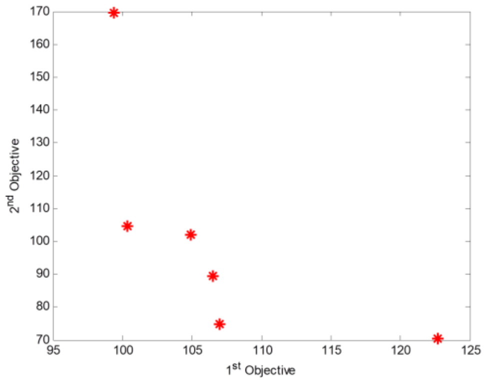
Test 2: In the first test problems for 60 jobs, the best solution is obtained for SJMCT-NSGA-II algorithm at generation 100 with number of population 100 and with crossover probabilities 0.6, 0.7, 0.8 and 0.9.



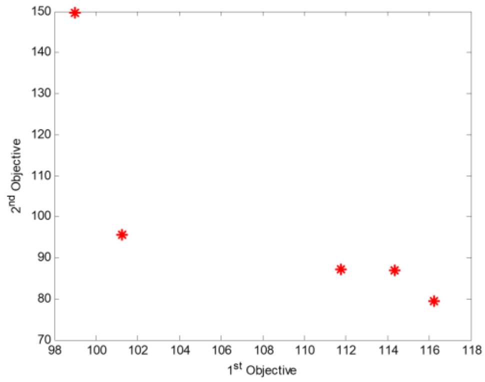
(a) Crossover Probability 0.6



(b) Crossover Probability 0.7



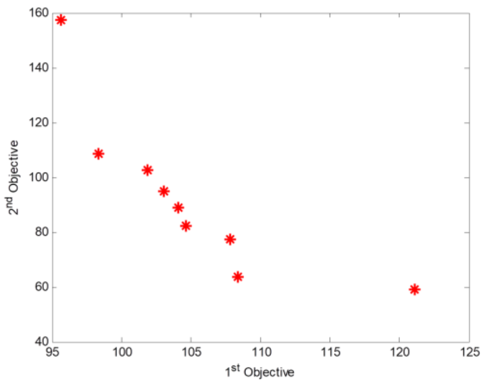
(c) Crossover Probability 0.8



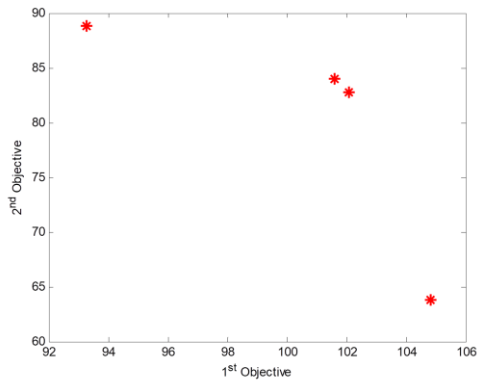
(d) Crossover Probability 0.9

Figure 5. 2. *Pareto optimal solutions for SJMCT- NSGA-II with generation 100 and different crossover probabilities*

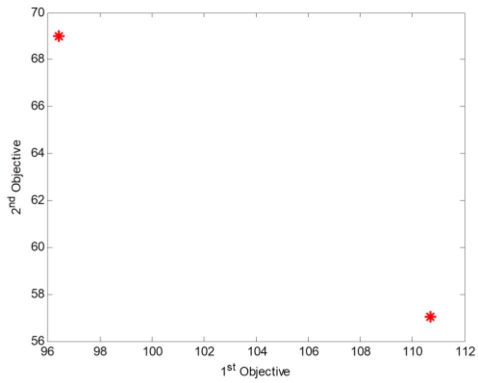
Test 3: In the first test problems for 60 jobs, the best solution is obtained for SJMCT-NSGA-II algorithm at generation 300 with number of population 100 and with crossover probabilities 0.6, 0.7, 0.8 and 0.9.



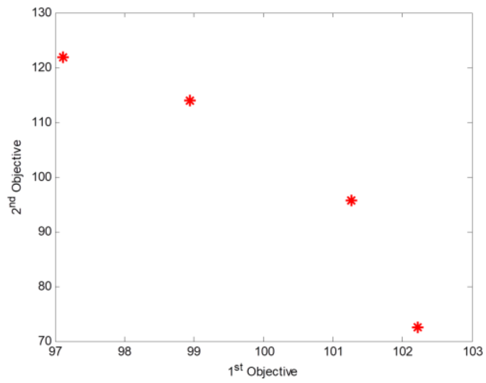
(a) Crossover Probability 0.6



(b) Crossover Probability 0.7



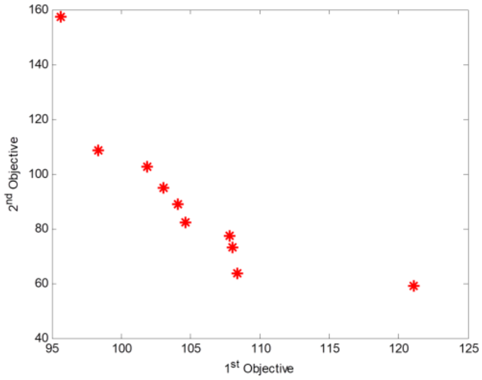
(c) Crossover Probability 0.8



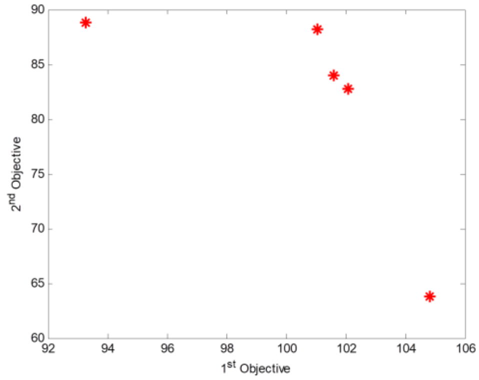
(d) Crossover Probability 0.9

Figure 5. 3. *Pareto optimal solutions for SJMCT- NSGA-II with generation 300 and different crossover probabilities*

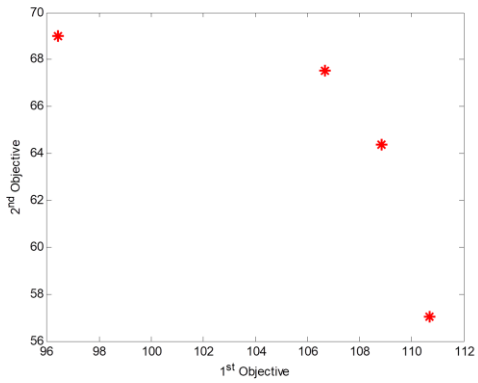
Test 4: In the first test problems for 60 jobs, the best solution is obtained for SJMCT-NSGA-II algorithm at generation 500 with number of population 100 and with crossover probabilities 0.6, 0.7, 0.8 and 0.9.



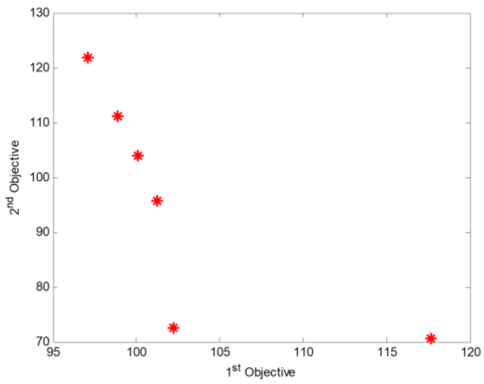
(a) Crossover Probability 0.6



(b) Crossover Probability 0.7



(c) Crossover Probability 0.8

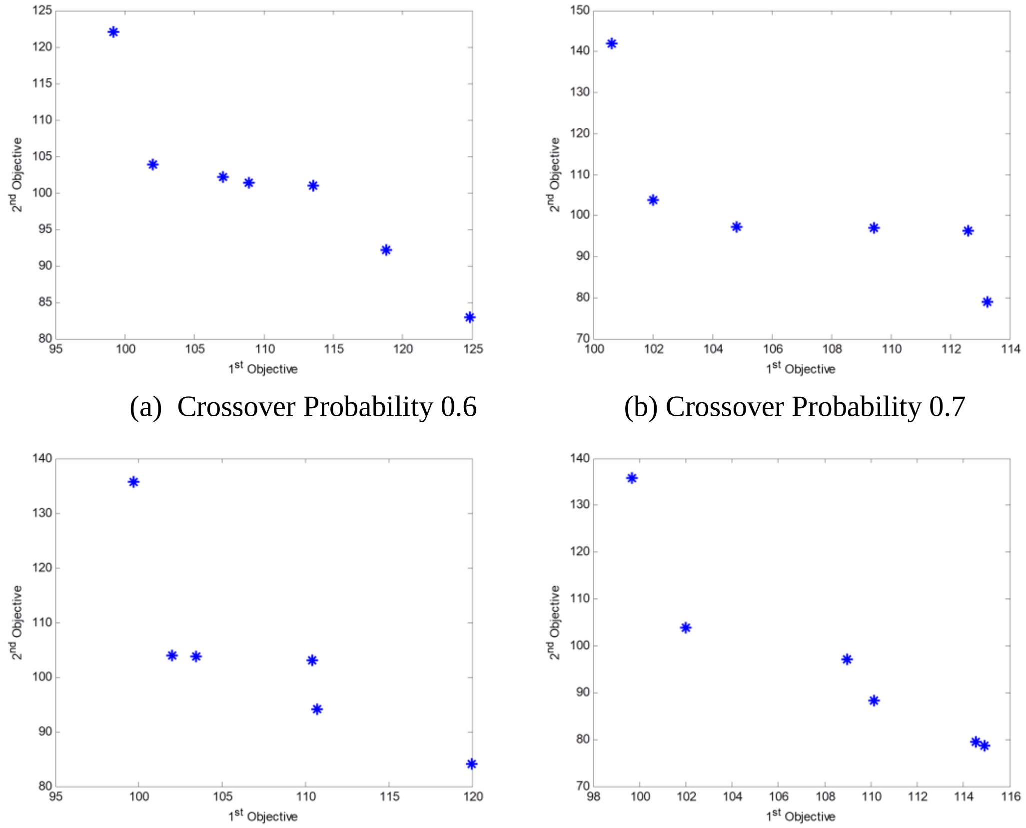


(d) Crossover Probability 0.9

Figure 5. 4. *Pareto optimal solutions for SJMCT- NSGA-II with generation 500 and different crossover probabilities*

5.2.2. Simulation results for SJMCT-SPEA-II algorithm

Test 1: In the first test problems for 60 jobs, the best solution is obtained for SJMCT-SPEA-II algorithm at generation 40 with number of population 100 and with crossover probabilities 0.6, 0.7, 0.8 and 0.9.



(a) Crossover Probability 0.6

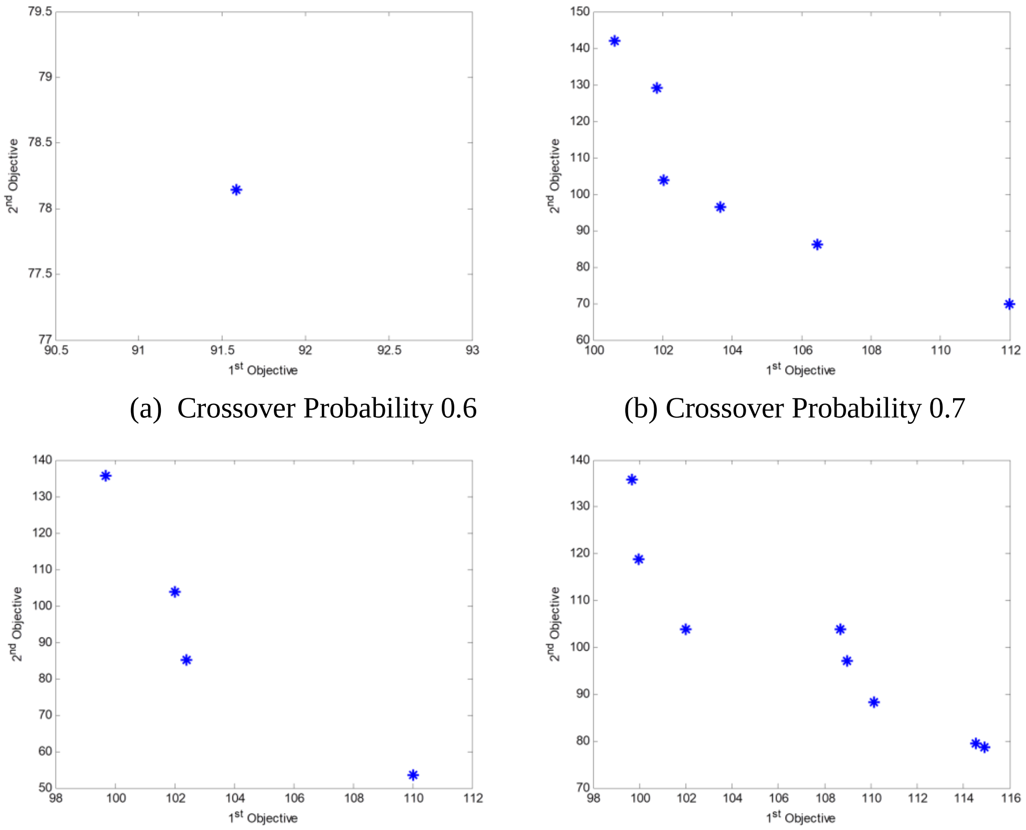
(b) Crossover Probability 0.7

(c) Crossover Probability 0.8

(d) Crossover Probability 0.9

Figure 5. 5. *Pareto optimal solutions for SJMCT- SPEA-II with generation 40 and different crossover probabilities*

Test 2: In the first test problems for 60 jobs, the best solution is obtained for SJMCT-SPEA-II algorithm at generation 100 with number of population 100 and with crossover probabilities 0.6, 0.7, 0.8 and 0.9.



(a) Crossover Probability 0.6

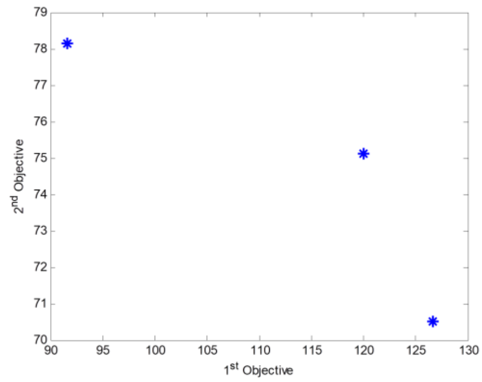
(b) Crossover Probability 0.7

(c) Crossover Probability 0.8

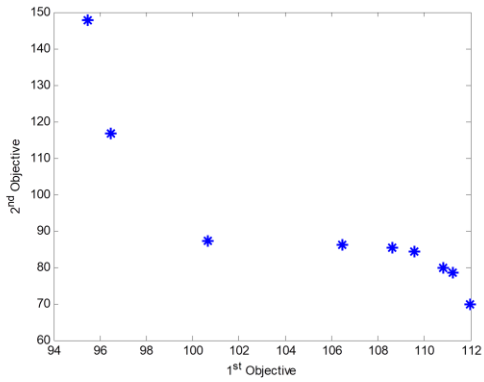
(d) Crossover Probability 0.9

Figure 5. 6. *Pareto optimal solutions for SJMCT- SPEA-II with generation 100 and different crossover probabilities*

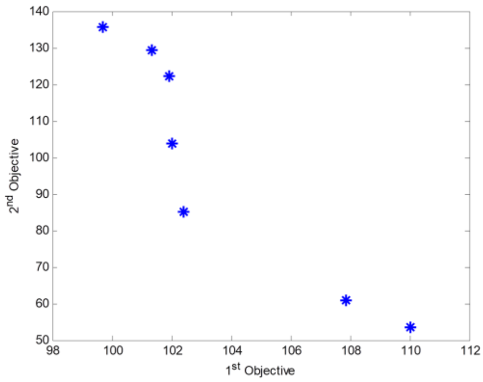
Test 3: In the first test problems for 60 jobs, the best solution is obtained for SJMCT-SPEA-II algorithm at generation 300 with number of population 100 and with crossover probabilities 0.6, 0.7, 0.8 and 0.9.



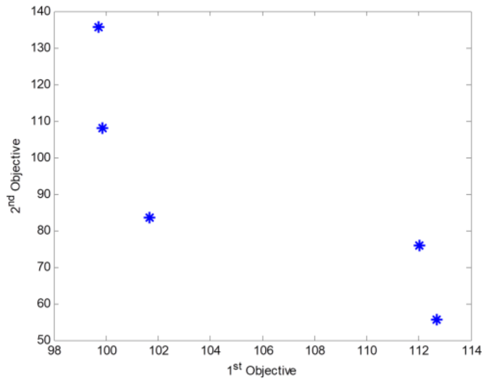
(a) Crossover Probability 0.6



(b) Crossover Probability 0.7



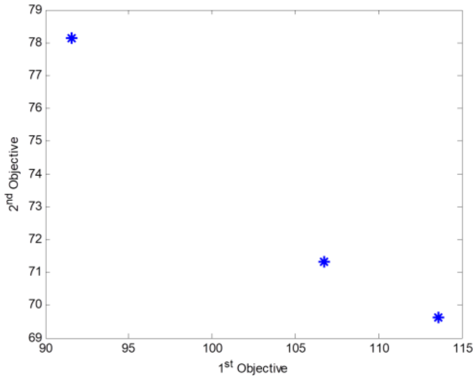
(c) Crossover Probability 0.8



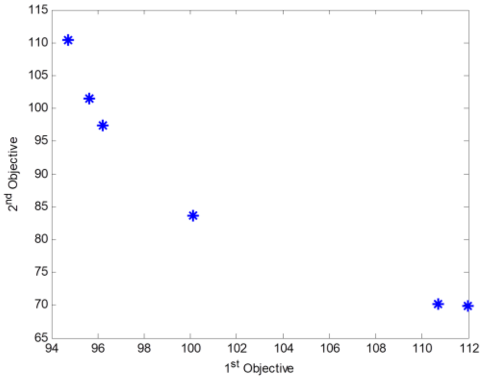
(d) Crossover Probability 0.9

Figure 5. 7. *Pareto optimal solutions for SJMCT- SPEA-II with generation 300 and different crossover probabilities*

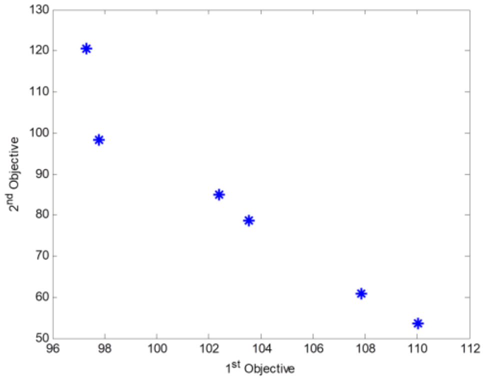
Test 4: In the first test problems for 60 jobs, the best solution is obtained for SJMCT-SPEA-II algorithm at generation 500 with number of population 100 and with crossover probabilities 0.6, 0.7, 0.8 and 0.9.



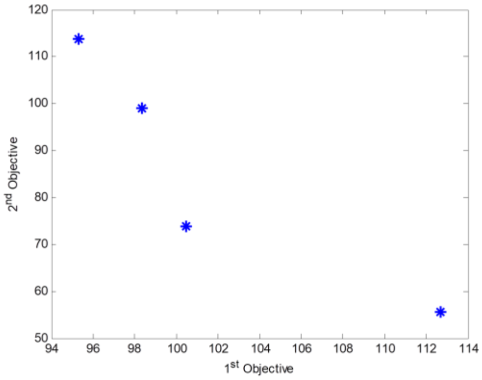
(a) Crossover Probability 0.6



(b) Crossover Probability 0.7



(c) Crossover Probability 0.8



(d) Crossover Probability 0.9

Figure 5. 8. *Pareto optimal solutions for SJMCT- SPEA-II with generation 500 and different crossover probabilities*

For more clarification, to discover the best configuration of SJMCT-NSGA-II and SJMCT-SPEA-II, Tables 5.2-5.17 and Figures 5.9-5.24 describe all results obtained from the first test problems represented before (in Figures 5.1-5.8) for each algorithm.

Table 5. 2. *The values of the best non-dominated front for 5 machines and 60 jobs with generation 40 and crossover probability 0.6*

Generation	Number of job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	Objective2	Objective1	Objective2
40	60	0.6	100.656	134.337	102.019	103.893
			104.677	**82.166**	124.826	82.961
			102.716	122.604	**99.185**	121.977
			104.072	89.047	107.065	102.157
					108.955	101.377
					118.807	92.171
					113.584	101.012

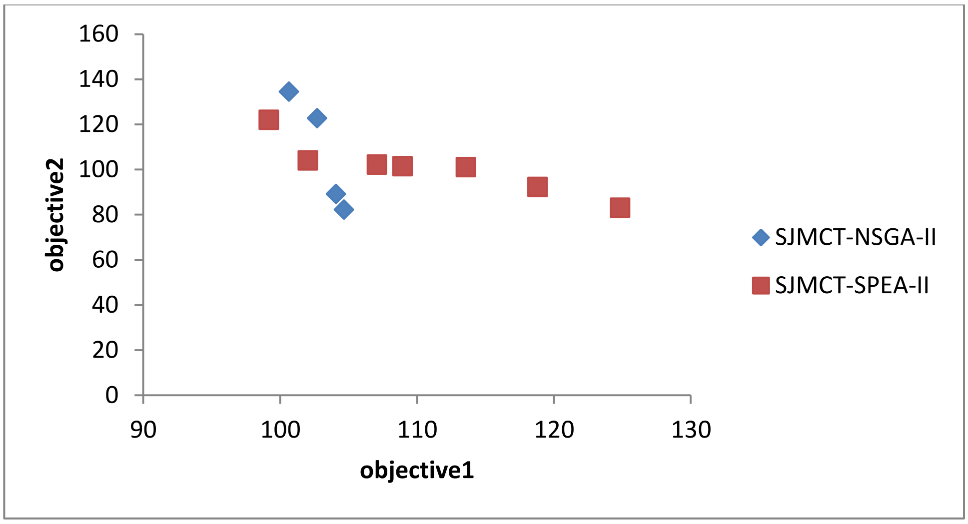


Figure 5. 9. *Solutions at generation 40 for 60 jobs (Crossover probability 0.6)*

In Table 5.2 and Figure 5.9 for 60 jobs, at generation 40 and crossover probability 0.6, the minimum value of objective1 is **99.185** at SJMCT-SPEA-II algorithm and the minimum value of objective2 equals to **82.166** at SJMCT-NSGA-II algorithm. That means, the Pareto set is all non-dominated individuals between **(99.185, 121.977)** and **(104.677, 82.166)** solutions.

Table 5. 3. *The values of the best non-dominated front for 5 machines and 60 jobs with generation 40 and crossover probability 0.7*

Generation	Number of job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	**Objective2**	**Objective1**	**Objective2**
40	60	0.7	**99.671**	137.937	100.620	141.913
			104.821	**63.836**	102.019	103.893
			103.123	123.708	113.239	79.024
					104.825	97.297
					109.451	97.028
					112.597	96.278

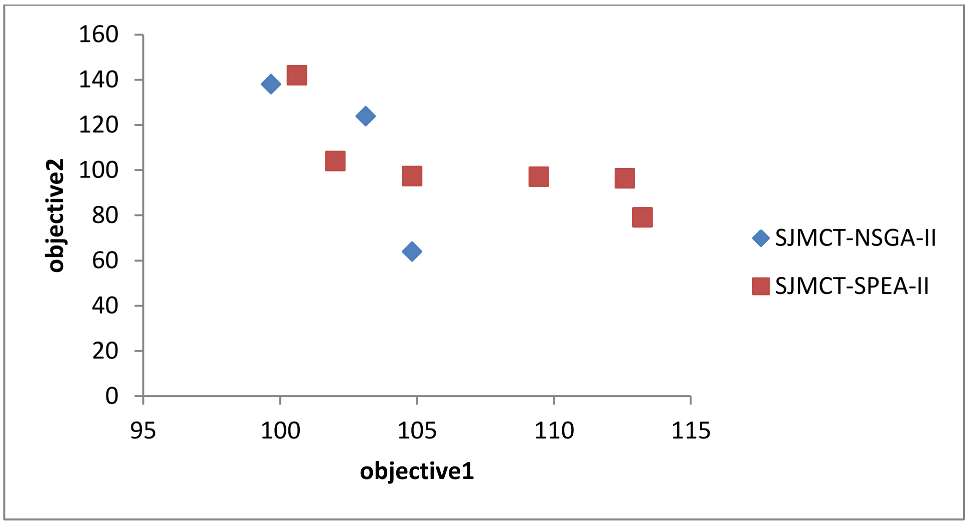


Figure 5. 10. *Solutions at generation 40 for 60 jobs (Crossover probability 0.7)*

In Table 5.3 and Figure 5.10 for 60 jobs, at generation 40 and crossover probability 0.7, the minimum value of objective1 is **99.671** at SJMCT- NSGA-II algorithm and the minimum value of objective2 equals to **63.836** at SJMCT-NSGA-II algorithm. That means, the Pareto set is all non-dominated individuals between **(99.671, 137.937)** and **(104.821, 63.836)** solutions.

Table 5. 4. The values of the best non-dominated front for 5 machines and 60 jobs with generation 40 and crossover probability 0.8

Generation	Number of job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	Objective2	Objective1	Objective2
40	60	0.8	102.229	134.565	**99.700**	135.708
			122.704	**70.499**	102.019	103.893
			112.861	86.100	119.957	84.093
			106.509	89.587	103.459	103.713
			104.911	101.917	110.686	94.117
			104.357	119.334	110.393	103.002
			104.459	118.323		

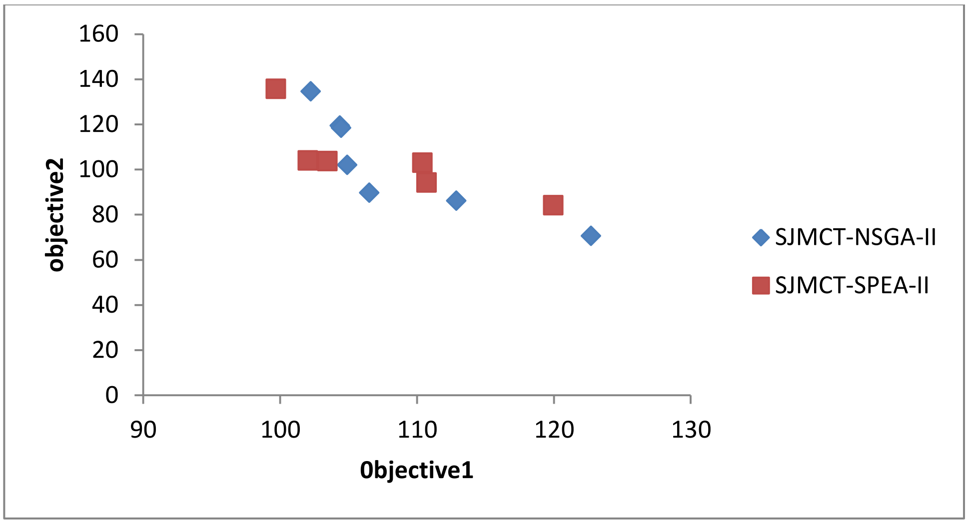


Figure 5. 11. *Solutions at generation 40 for 60 jobs (Crossover probability 0.8)*

In Table 5.4 and Figure 5.11 for 60 jobs, at generation 40 and crossover probability 0.8, the minimum value of objective1 is **99.700** at SJMCT- SPEA-II algorithm and the minimum value of objective2 equals to **70.499** at SJMCT-NSGA-II algorithm. That means, the Pareto set is all non-dominated individuals between **(99.700, 135.708)** and **(122.704, 70.499)** solutions.

Table 5. 5. *The values of the best non-dominated front for 5 machines and 60 jobs with generation 40 and crossover probability 0.9*

Generation	Number of job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	Objective2	Objective1	Objective2
40	60	0.9	123.954	92.965	99.700	135.708
			98.995	149.581	102.019	103.893
			101.449	115.698	114.911	**78.628**
			112.074	93.737	114.547	79.531
			111.837	109.917	110.128	88.279
			103.685	111.078	108.962	97.073

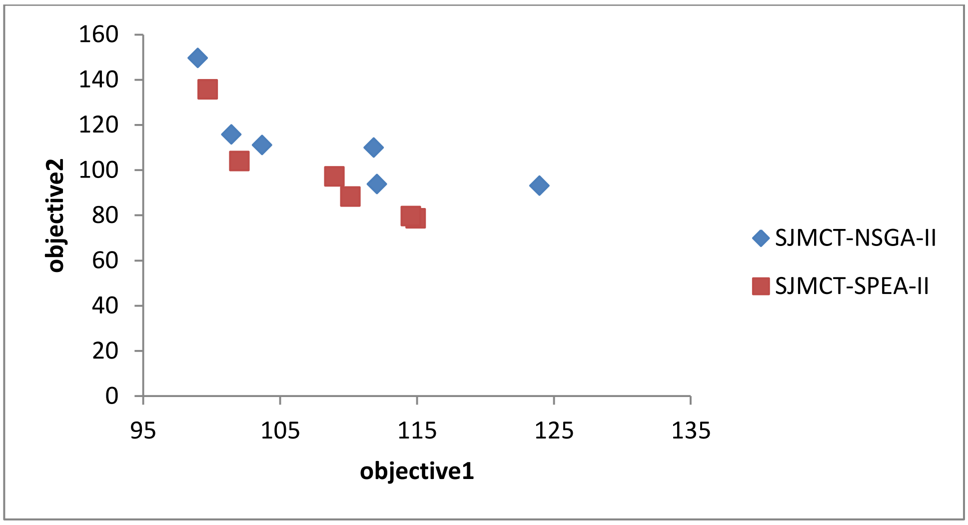


Figure 5. 12. *Solutions at generation 40 for 60 jobs (Crossover probability 0.9)*

In Table 5.5 and Figure 5.12 for 60 jobs, at generation 40 and crossover probability 0.9, the minimum value of objective1 is **98.995** at SJMCT-NSGA-II algorithm and the minimum value of objective2 equals to **78.628** at SJMCT-SPEA-II algorithm. That means, the Pareto set is all non-dominated individuals between **(98.995, 149.581)** and **(114.911, 78.628)** solutions.

Table 5. 6. *The values of the best non-dominated front for 5 machines and 60 jobs with generation 100 and crossover probability 0.6*

Generation	Number of job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	**Objective2**	**Objective1**	**Objective2**
100	60	0.6	100.656	134.337	**91.587**	**78.141**
			113.802	81.107		
			104.677	82.166		
			103.183	113.331		
			104.072	89.047		
			102.716	122.604		

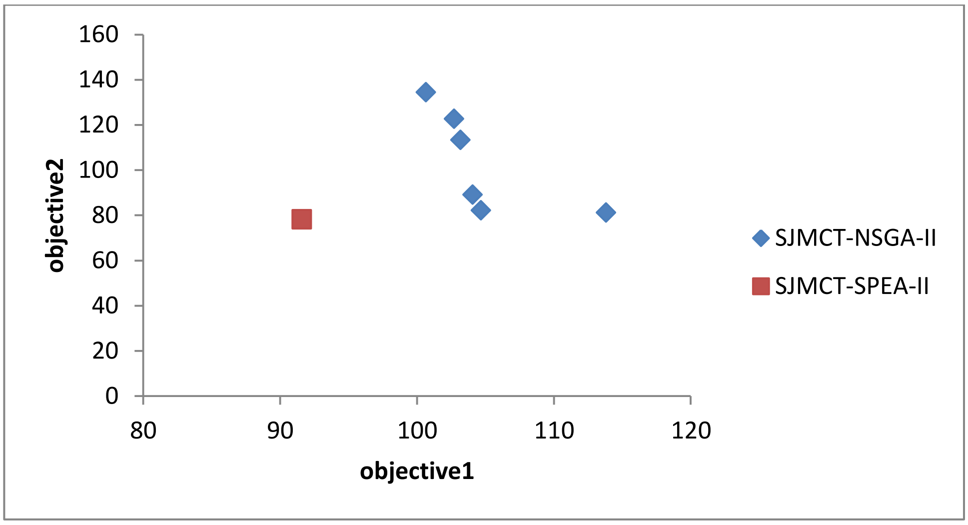


Figure 5. 13. *Solutions at generation 100 for 60 jobs (Crossover probability 0.6)*

In Table 5.6 and Figure 5.13 for 60 jobs, at generation 100 and crossover probability 0.6, the minimum value of objective1 is **91.587** at SJMCT-SPEA-II algorithm and the minimum value of objective2 equals to **78.141 at** SJMCT-SPEA-II algorithm. That means, the Pareto set is the non-dominated solution **(91.587, 78.141)**.

Table 5. 7. *The values of the best non-dominated front for 5 machines and 60 jobs with generation 100 and crossover probability 0.7*

Generation	Number of job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	Objective2	Objective1	Objective2
100	60	0.7	104.821	**63.836**	111.988	69.860
			93.275	88.818	100.620	141.913
					106.452	86.186
					102.019	103.893
					103.661	96.431
					101.834	128.962

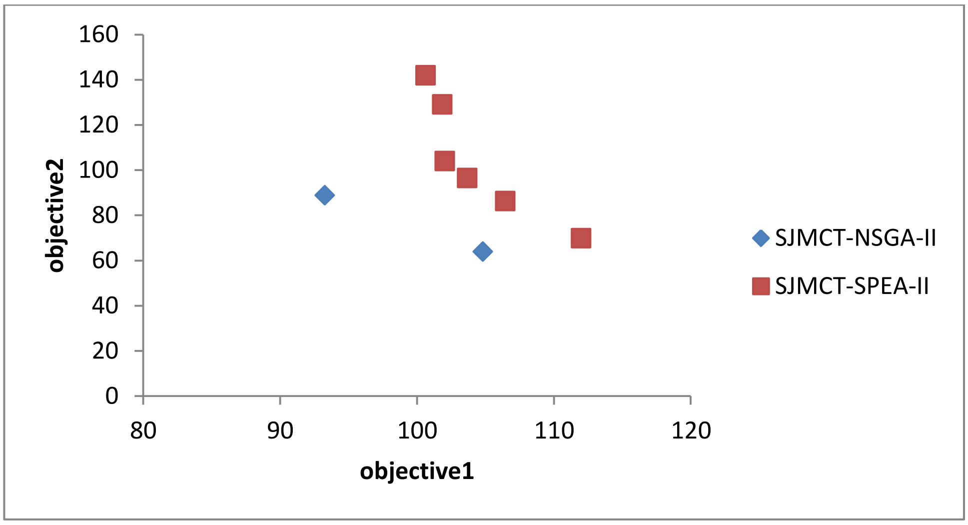


Figure 5. 14. *Solutions at generation 100 for 60 jobs (Crossover probability 0.7)*

In Table 5.7 and Figure 5.14 for 60 jobs, at generation 100 and crossover probability 0.7, the minimum value of objective1 is **93.275** at SJMCT-NSGA-II algorithm and the minimum value of objective2 equals to **63.836** at SJMCT-NSGA-II algorithm. That means, the Pareto set is all non-dominated individuals between **(93.275, 88.818)** and **(104.821, 63.836)** solutions.

Table 5. 8. *The values of the best non-dominated front for 5 machines and 60 jobs with generation 100 and crossover probability 0.8*

Generation	Number of job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	**Objective2**	**Objective1**	**Objective2**
100	60	0.8	122.704	70.499	110.019	**53.667**
			99.356	169.563	102.406	85.020
			100.331	104.701	99.700	135.708
			107.004	74.735	102.019	103.893
			104.911	101.917		
			106.509	89.587		

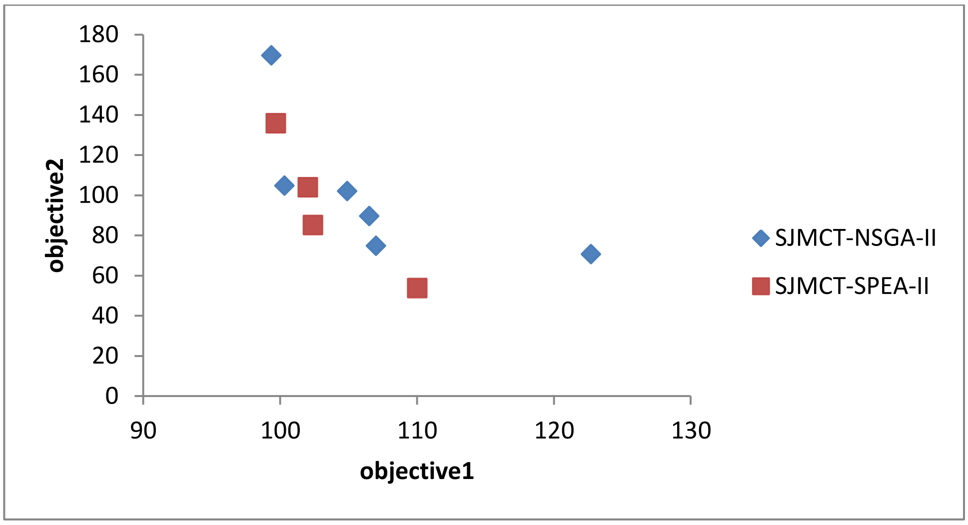


Figure 5. 15. *Solutions at generation 100 for 60 jobs (Crossover probability 0.8)*

In Table 5.8 and Figure 5.15 for 60 jobs, at generation 100 and crossover probability 0.8, the minimum value of objective1 is **99.356** at SJMCT-NSGA-II algorithm and the minimum value of objective2 equals to **53.667** at SJMCT-SPEA-II algorithm. That means, the Pareto set is all non-dominated individuals between **(99.356, 169.563)** and **(110.019, 53.667)** solutions.

Table 5. 9. *The values of the best non-dominated front for 5 machines and 60 jobs with generation 100 and crossover probability 0.9*

Generation	Number of job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	Objective2	Objective1	Objective2
100	60	0.9	**98.9954**	149.5807	99.700	135.708
			116.2029	79.48637	102.019	103.893
			101.2643	95.724	99.974	118.774
			111.7504	87.20371	114.911	**78.628**
			114.3319	86.9915	114.547	79.531
					110.128	88.279
					108.962	97.073
					108.696	103.788

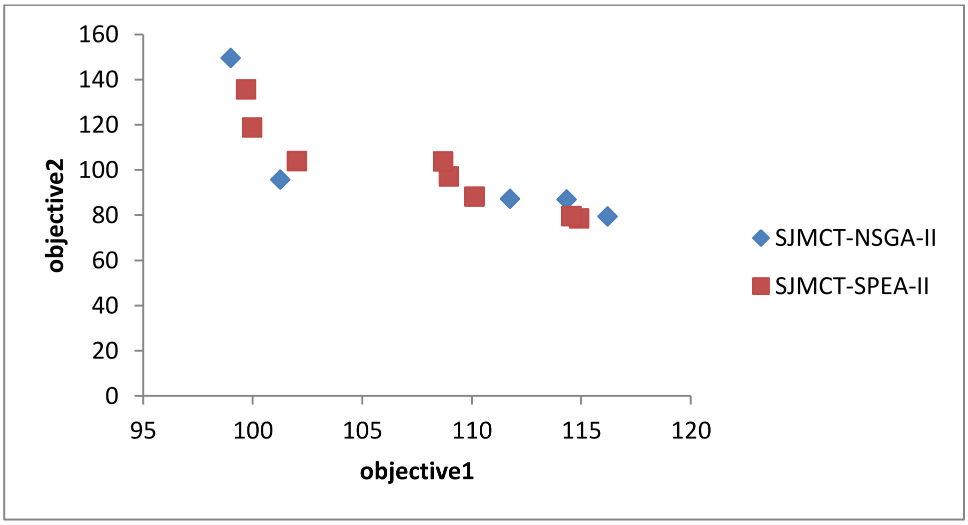


Figure 5. 16. *Solutions at generation 100 for 60 jobs (Crossover probability 0.9)*

In Table 5.9 and Figure 5.16 for 60 jobs, at generation 100 and crossover probability 0.9, the minimum value of objective1 is **98.9954** at SJMCT-NSGA-II algorithm and the minimum value of objective2 equals to **78.628** at SJMCT-SPEA-II algorithm. That means, the Pareto set is all non-dominated individuals between **(98.9954, 149.5807)** and **(114.911, 78.628)** solutions.

Table 5. 10. *The values of the best non-dominated front for 5 machines and 60 jobs with generation 300 and crossover probability 0.6*

Generation.	Number of job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	Objective2	Objective1	Objective2
300	60	0.6	121.113	**59.114**	**91.587**	78.141
			95.650	157.273	126.673	70.516
			98.355	108.603	120.030	75.110
			108.409	63.721		
			107.792	77.312		
			101.888	102.758		
			104.677	82.166		
			103.079	94.841		
			104.072	89.047		

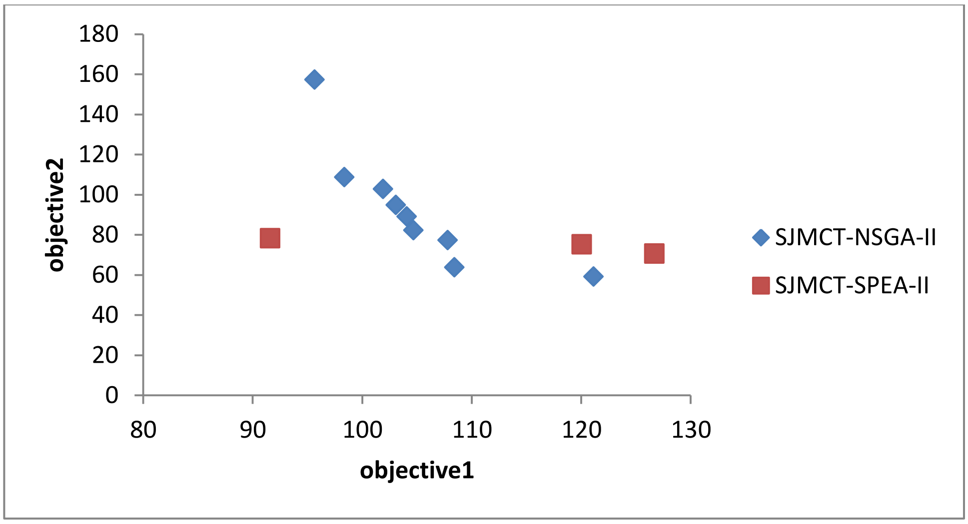


Figure 5. 17. *Solutions at generation 300 for 60 jobs (Crossover probability 0.6)*

In Table 5.10 and Figure 5.17 for 60 jobs, at generation 300 and crossover probability 0.6, the minimum value of objective1 is **91.587** at SJMCT-SPEA-II algorithm and the minimum value of objective2 equals to **59.114** at SJMCT-NSGA-II algorithm. That means, the Pareto set is all non-dominated individuals between **(91.587, 78.141)** and **(121.113, 59.114)** solutions.

Table 5. 11. *The values of the best non-dominated front for 5 machines and 60 jobs with generation 300 and crossover probability 0.7*

Generation	Number of job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	Objective2	Objective1	Objective2
300	60	0.7	104.821	**63.836**	95.489	147.707
			93.275	88.818	96.482	116.788
			102.093	82.724	100.651	87.274
			101.583	83.964	111.988	69.860
					111.249	78.447
					110.818	79.762
					106.452	86.186
					108.612	85.413
					109.557	84.451

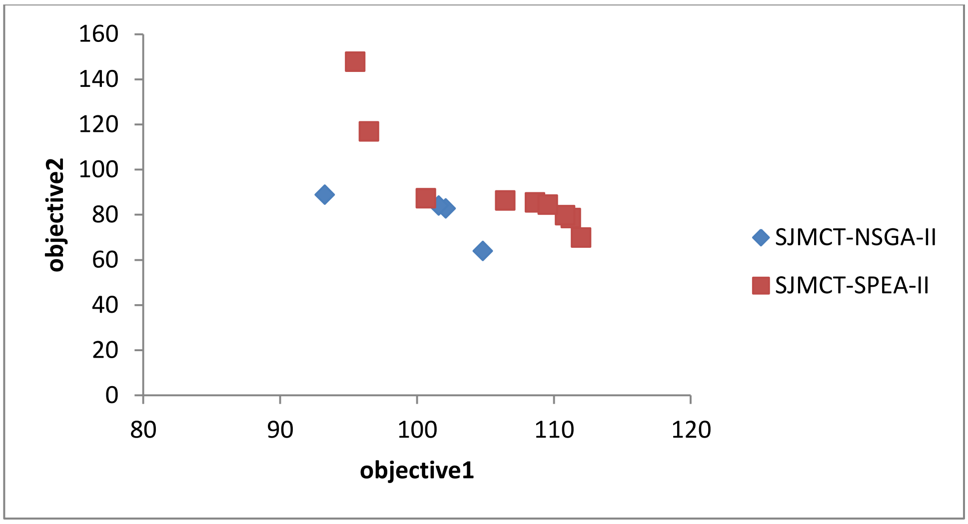


Figure 5. 18. *Solutions at generation 300 for 60 jobs (Crossover probability 0.7)*

In Table 5.11 and Figure 5.18 for 60 jobs, at generation 300 and crossover probability 0.7, the minimum value of objective1 is **93.275** at SJMCT-NSGA-II algorithm and the minimum value of objective2 equals to **63.836** at SJMCT-NSGA-II algorithm. That means, the Pareto set is all non-dominated individuals between **(93.275, 88.818)** and **(104.821, 63.836)** solutions

Table 5. 12. *The values of the best non-dominated front for 5 machines and 60 jobs with generation 300 and crossover probability 0.8*

Generation	Number of job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	Objective2	Objective1	Objective2
300	60	0.8	110.691	57.063	110.019	**53.667**
			96.414	68.981	107.844	60.935
					99.700	135.708
					101.338	129.323
					102.406	85.020
					101.929	122.292
					102.019	103.893

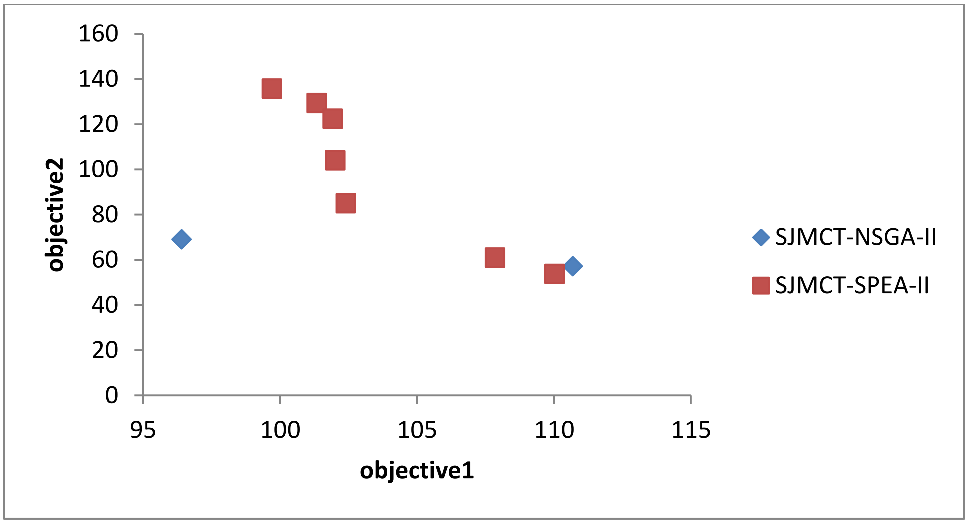


Figure 5. 19. *Solutions at generation 300 for 60 jobs (Crossover probability 0.8)*

In Table 5.12 and Figure 5.19 for 60 jobs, at generation 300 and crossover probability 0.8, the minimum value of objective1 is **96.414** at SJMCT-NSGA-II algorithm and the minimum value of objective2 equals to **53.667** at SJMCT-SPEA-II algorithm. That means, the Pareto set is all non-dominated individuals between **(96.414, 68.981)** and **(110.019, 53.667)** solutions.

Table 5. 13. *The values of the best non-dominated front for 5 machines and 60 jobs with generation 300 and crossover probability 0.9*

Generation	Number of job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	Objective2	Objective1	Objective2
300	60	0.9	102.218	72.547	112.674	**55.712**
			97.116	121.903	101.679	83.513
			101.264	95.724	99.700	135.708
			98.946	114.026	99.844	108.079
					112.010	75.979

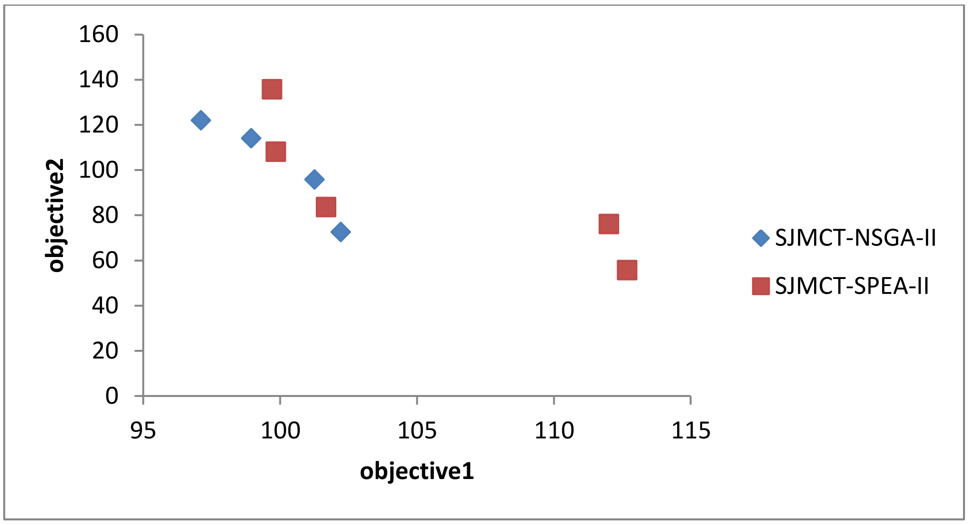


Figure 5. 20. *Solutions at generation 300 for 60 jobs (Crossover probability 0.9)*

In Table 5.13 and Figure 5.20 for 60 jobs, at generation 300 and crossover probability 0.9, the minimum value of objective1 is **97.116** at SJMCT-NSGA-II algorithm and the minimum value of objective2 equals to **55.712** at SJMCT-SPEA-II algorithm. That means, the Pareto set is all non-dominated individuals between **(97.116, 121.903)** and **(112.674, 55.712)** solutions.

Table 5. 14. *The values of the best non-dominated front for 5 machines and 60 jobs with generation numbers 500 and crossover probability 0.6*

Generation	Number of job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	**Objective2**	**Objective1**	**Objective2**
500	60	0.6	121.113	**59.114**	**91.587**	78.141
			95.650	157.273	106.759	71.317
			98.355	108.603	113.615	69.618
			108.409	63.721		
			101.888	102.758		
			104.677	82.166		
			103.079	94.841		
			107.792	77.312		
			104.072	89.047		
			108.035	73.248		

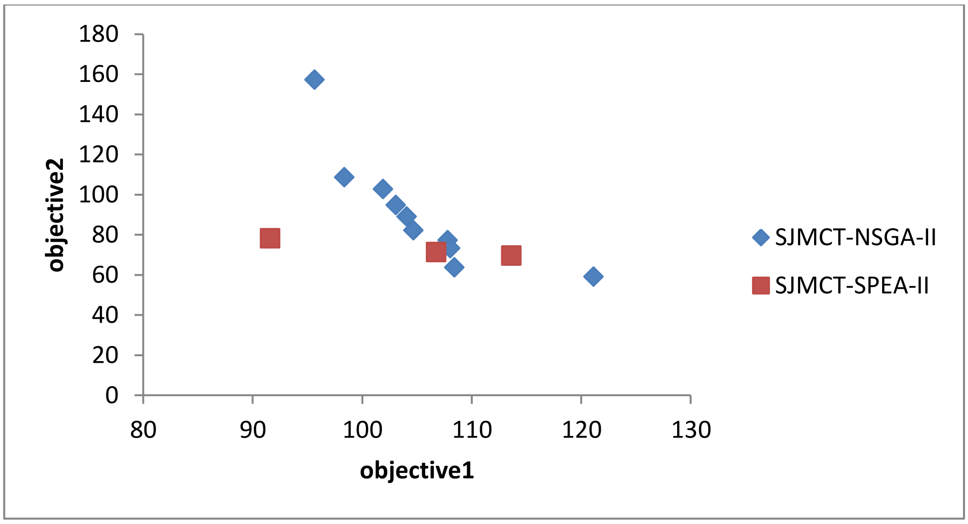


Figure 5. 21. *Solutions at generation 500 for 60 jobs (Crossover probability 0.6)*

In Table 5.14 and Figure 5.21 for 60 jobs, at generation 500 and crossover probability 0.6, the minimum value of objective1 is **91.587** at SJMCT-SPEA-II algorithm and the minimum value of objective2 equals to **59.114** at SJMCT-NSGA-II algorithm. That means, the Pareto set is all non-dominated individuals between **(91.587, 78.141)** and **(121.113, 59.114)** solutions.

Table 5. 15. *The values of the best non-dominated front for 5 machines and 60 jobs with generation 500 and crossover probability 0.7*

Generation	Number of job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	**Objective2**	**Objective1**	**Objective2**
500	60	0.7	104.821	**63.836**	94.736	110.466
			93.275	88.818	95.634	101.489
			102.093	82.724	96.236	97.437
			101.040	88.182	100.119	83.699
			101.583	83.964	111.988	69.860
					110.708	70.256

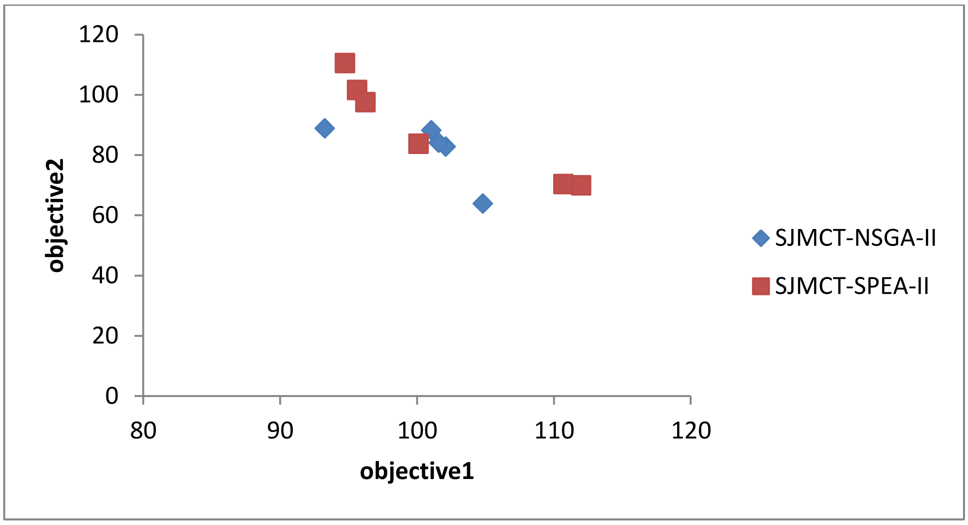


Figure 5. 22. *Solutions at generation 500 for 60 jobs (Crossover probability 0.7)*

In Table 5.15 and Figure 5.22 for 60 jobs, at generation 500 and crossover probability 0.7, the minimum value of objective1 is **93.275** at SJMCT-NSGA-II algorithm and the minimum value of objective2 equals to **63.836** at SJMCT-NSGA-II algorithm. That means, the Pareto set is all non-dominated individuals between **(93.275, 88.818)** and **(104.821, 63.836)** solutions.

Table 5. 16. *The values of the best non-dominated front for 5 machines and 60 jobs with generation 500 and crossover probability 0.8*

Generation	Number of job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	Objective2	Objective1	Objective2
500	60	0.8	110.691	57.063	110.019	**53.667**
			96.414	68.981	107.844	60.935
			106.689	67.534	97.786	98.376
			108.864	64.371	97.304	120.473
					103.522	78.553
					102.406	85.020

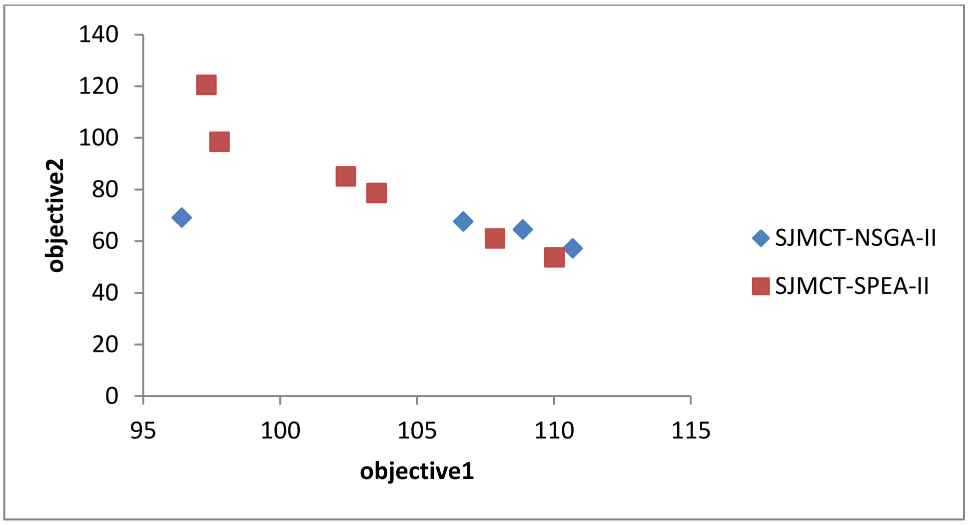


Figure 5. 23. *Solutions at generation 500 for 60 jobs (Crossover probability 0.8)*

In Table 5.16 and Figure 5.23 for 60 jobs, at generation 500 and crossover probability 0.8, the minimum value of objective1 is **96.414** at SJMCT-NSGA-II algorithm and the minimum value of objective2 equals to **53.667** at SJMCT-SPEA-II algorithm. That means, the Pareto set is all non-dominated individuals between **(96.414, 68.981)** and **(110.019, 53.667)** solutions.

Table 5. 17. *The values of the best non-dominated front for 5 machines and 60 jobs with 500 generation and crossover probability 0.9*

Generation	Number of job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	Objective2	Objective1	Objective2
500	60	0.9	97.116	121.903	112.674	**55.712**
			117.669	70.647	100.471	73.837
			102.218	72.547	**95.291**	113.669
			101.264	95.724	98.359	99.067
			98.870	111.135		
			100.098	103.991		

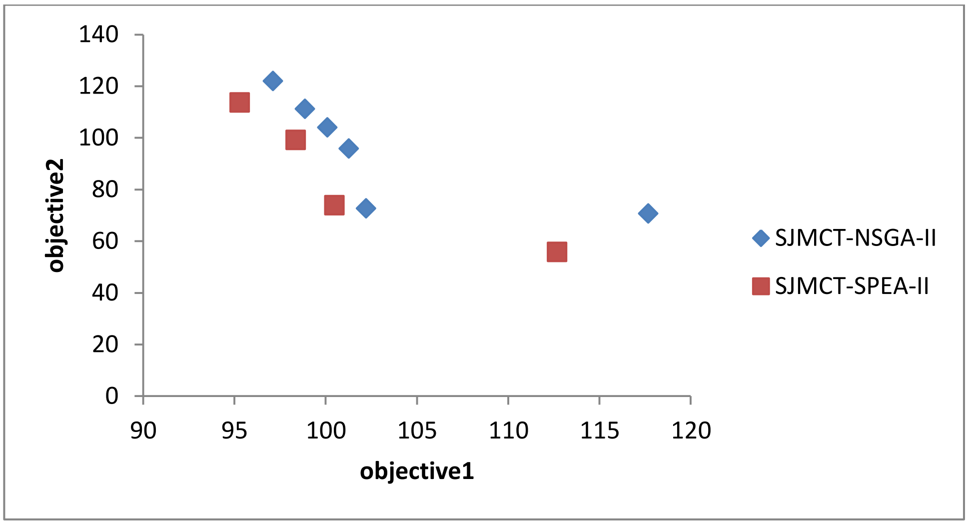


Figure 5. 24. *Solutions at generation 500 for 60 jobs (Crossover probability 0.9)*

In Table 5.17 and Figure 5.24 for 60 jobs, at generation 500 and crossover probability 0.9, the minimum value of objective1 is **95.291** at SJMCT-SPEA-II algorithm and the minimum value of objective2 equals to **55.712** at SJMCT-SPEA-II algorithm. That means, the Pareto set is all non-dominated individuals between **(95.291, 113.669)** and **(112.674, 55.712)** solutions.

As seen in Tables 5.2-5.17 and Figures 5.9-5.24 it is difficult to know the best algorithm. Therefore, we decided to use the performance measures in Section 5.4. Also, as further study, the minimum and average values for the first test problems represented in Table 5.18.

Table 5. 18. *Minimum and average values for 60 jobs to all algorithm numbers and objectives*

Generation numbers	Crossover Probability	Minimum and Average	Objective 1		Objective 2	
			SJMCT-NSGA-II	**SJMCT-SPEA-II**	**SJMCT-NSGA-II**	**SJMCT-SPEA-II**
40	0.6	**Min.**	**100.656**	**99.185**	**82.166**	**82.961**
		Ave.	103.030	110.635	107.039	100.793
100		**Min.**	**100.656**	**91.587**	**81.107**	**78.141**
		Ave.	104.851	91.587	103.766	78.141
300		**Min.**	**95.650**	**91.587**	**59.114**	**70.516**
		Ave.	105.004	112.763	92.760	74.589
500		**Min.**	**95.650**	**91.587**	**59.114**	**69.618**
		Ave.	105.307	103.987	90.808	73.025
40	0.7	**Min.**	**99.671**	**100.620**	**63.836**	**79.024**
		Ave.	102.538	107.125	108.494	102.572
100		**Min.**	**93.275**	**100.620**	**63.836**	**69.860**
		Ave.	99.048	104.429	76.327	104.541
300		**Min.**	**93.275**	**95.489**	**63.836**	**69.860**
		Ave.	100.443	105.700	79.836	92.877
500		**Min.**	**93.275**	**94.736**	**63.836**	**69.860**
		Ave.	100.563	101.570	81.505	88.868
40	0.8	**Min.**	**102.229**	**99.700**	**70.499**	**84.093**
		Ave.	108.290	107.702	102.904	104.088
100		**Min.**	**99.356**	**99.700**	**70.499**	**53.667**
		Ave.	106.803	103.536	101.834	94.572
300		**Min.**	**96.414**	**99.700**	**57.063**	**53.667**
		Ave.	103.552	103.608	63.022	98.691
500		**Min.**	**96.414**	**97.304**	**57.063**	**53.667**
		Ave.	105.665	103.147	64.487	82.837
40	0.9	**Min.**	**98.995**	**99.700**	**92.965**	**78.628**
		Ave.	108.666	108.378	112.163	97.185
100		**Min.**	**98.995**	**99.700**	**79.486**	**78.628**
		Ave.	108.509	107.367	99.797	100.709
300		**Min.**	**97.1156**	**99.700**	**72.5474**	**55.712**
		Ave.	99.8861	105.181	101.0502	91.798
500		**Min.**	**97.116**	**95.291**	**70.647**	**55.712**
		Ave.	102.873	101.699	95.991	85.571

Table 5.18 leads to the best generation will be selected in next test problems to indicate the efficiency of proposed algorithms. More details about the effects of parameters for Tables 5.2-5.18 are explained in Section 5.3.

5.3. The Effect of Parameters

The effect of crossover, mutation probabilities and the effect of generation numbers of the best non-dominated front for 5 machines and 60 jobs can be discussed as follows:

- **Effect of crossover and mutation probabilities:**

The crossover operator used to generate two good individuals, called offspring, from the two selected parents (Vallada and Ruiz, 2011). A standard one-point crossover is used in this study to produce two offspring from two parent solutions and the mutation operator selects two random genes and then exchanges their positions.

Testing different crossover and mutation operators gives us the variety of Pareto frontier sets.

- **Effect of generation numbers:**

Due to the first test problems concerned with 60 jobs for all objectives, the minimum values and averages at most cases obtained by increasing the generation numbers from 40 to 500 as seen in Table 5.18. Moreover, this table shows that the best minimum value of each objective obtained when the generation number is 500 for each algorithm. So we conclude, there is a need for the second test problems that will be performed at different seeds when the generation number is 500.

Since the comparison of two Pareto front is too difficult because each front is a set of non-dominated solution. Therefore, the diversity metrics of multi-objective optimization (MOO) in Section 5.4 are used to define the best evolutionary performance of SJMCT-NSGA-II and SJMCT-SPEA-II algorithms. The mean and variance of spacing and spread metrics to the second test problems with 20, 60 and 100 jobs and generation number is 500 for 10 runs implemented by MATLAB programming are given respectively in Table 5.19 and Table 5.20.

5.4. Performance Measures

In multi-objective optimization the most important consideration is the quantitative metrics used for defining the optimality of different solution sets. However, comparing two sets of solutions is more complex because of the multi-objectives. These metrics make the comparison between algorithms is relatively easy. Typically, the performance measures help us to find the convergence and the diversity between the Pareto optimal front PF_{Known} and the obtained solutions PF_{True}. Veldhuizen and Lamont, (2000) display the small example to show the relationship between PF_{True} and PF_{Known} as given in Figure (5.25):

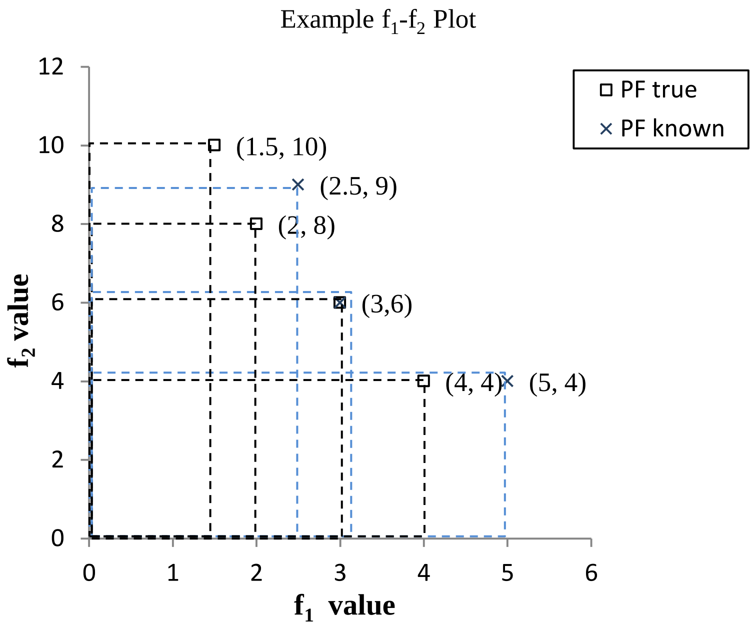


Figure 5. 25. *PF_{known} / PF_{true} example (Veldhuizen and Lamont, 2000)*

Jiang et al., (2014) considered four types of the MOO metrics based on capacity, convergence and diversity of performance criteria as follows:

A. Capacity metrics: This group of metrics calculates the number or ratio of non-dominated solutions in *S* (where, *S* solutions of the best non-dominated front PF_{True}) that satisfies given predefined requirements.
B. Convergence metrics: These are the metrics for measuring the proximity of solution set *S* to optimal solution *P* (where *P* is the optimal Pareto front PF_{Known}).
C. Diversity metrics: These metrics include two forms of information:
   1) "Distribution" measures how evenly the solutions of S in the objective space scattered.
   2) Spread indicates how well do the solutions of *S* arrive at the extreme of true PF_S.
D. Convergence-diversity metrics: They indicate both the convergence and diversity of *S* on a single scale.

A. Capacity Metrics

The error ratio (ER) measure, indicates the percentage of solutions that are not members of the Pareto optimal set (Godinez, Espinosa, and Montes, 2010).

$$ER = \frac{\sum_i^n e_i}{n} \tag{5.1}$$

Where, *n* is the number of vectors in the current set of non-dominated vectors available, $e_i = 0$ indicates an ideal behavior and $ER = 0$. If vector *i* is a member of the Pareto optimal set that mean $e_i = 1$.

B. Convergence Metrics

Ghosh and Das, (2008) and Veldhuizen and Lamont, (2000) represented generational distance GD convergence metrics, which measure the degree of proximity based on the distance between the solutions in *S* to those in *P*.

$$GD(S,P) = \frac{\left|\sum_{i=1}^{|S|} d_i^q\right|^{1/q}}{|S|} \tag{5.2}$$

Where; $d_i = \min_{\vec{p}\in P}\left\|F(\vec{S}_i) - F(\vec{p})\right\|, \vec{S}_i \in S \; and \; q = 2.$

d_i is a smallest distance from $\vec{S} \in S$ to the closet solution in *P*.

C. Diversity Metrics

Diversity metrics indicate the distribution and spread of solutions in the non-dominated solution set *S*.

1) Distribution Metrics: (Deb et al., 2000) proposed a metric Δ' that compares all the solutions' consecutive distances with the average distance.

$$\Delta'(S) = \sum_{i=1}^{|S|} \frac{(d_i - \overline{d})}{|S|} \tag{5.3}$$

Where; d_i is the Euclidean distance between consecutive solutions in *S*, and $\overline{d}$, is the average of d_i. If all the pair of consecutive solutions have equal distance, then $d_i = \overline{d}$, $\Delta'(S) = 0$, and *S* has a perfect distribution.

Another distribution metrics is spacing metric proposed by (Schott, 1995). A metric measuring the closet distance of pairwise solutions in *S*. (Veldhuizen and Lamont, 2000) defined this metric as given in equation (5.4):

$$SP(S) = \sqrt{\sum_{i=1}^{|S|}\left(\overline{d} - d_i\right)^2 / |S| - 1} \tag{5.4}$$

Where, $d_i = min_j\left(\left|f_1^i(\vec{x}) - f_1^j(\vec{x})\right| + \left|f_2^i(\vec{x}) - f_2^j(\vec{x})\right|\right) \quad i,j = 1, \dots, S$

$\overline{d}$ is the mean of all d_i and S is the number of obtained solutions. A value of zero for this metric indicates all members of S are equidistantly spaced.

2) Spread Metric: The overall Pareto spread (OS) quantifies how much of the extreme regions are covered by set S (Jiang et al., 2014).

$$OS(S, P_G, P_B) = \prod_{k=1}^{m} \frac{\left|\max_{\vec{S}\in S} f_k(\vec{S}) - \min_{\vec{S}\in S} f_k(\vec{S})\right|}{|f_k(P_B) - f_k(P_G)|} \tag{5.5}$$

Where $\max_{\vec{S}\in S} f_k(\vec{S})$, $\min_{\vec{S}\in S} f_k(\vec{S})$ are the maximum and minimum values of the k^{th} objective in *S*. For more details see (Wu and Azarm, 2000).

3) Distribution and Spread Metrics: (Deb et al., 2002) have proposed Diversity Metric Δ. This metric consider the distribution and spread of obtained solution *S* simultaneously. It is defined in Equation (5.6):

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1}|(d_i - \bar{d})|}{d_f + d_l + (S-1)\bar{d}} \tag{5.6}$$

$$d_i = \sqrt{\left(f_1^i(\vec{x}) - f_1^j(\vec{x})\right)^2 + \left(f_2^i(\vec{x}) - f_2^j(\vec{x})\right)^2}$$

d_i is the Euclidean distance between consecutive solutions (Ghosh and Das, 2008) and $\bar{d}$ is the average of all distances d_i. d_f *and* d_l represent the Euclidean distance between the extreme solutions and the boundary solutions of the obtained non-dominated set. As seen in Figure 5.26. d_i, *i*=1,2,…,(S-1) and (*S-1*) the consecutive distance of the best non-dominated front. In this metric, lesser value is the better result (Deb et al., 2000).

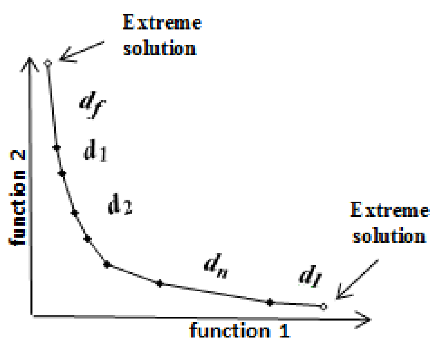


Figure 5. 26. *The Euclidean distance in the solutions*

D. Convergence-Diversity Metrics

The metric Inverted General Distance (IGD) is introduced by (Jiang et al., 2014) measures the quality of the optimal solution set *S* in terms of convergence and diversity on a single scale.

$$IGD(S,P) = \frac{\left|\sum_{i=1}^{|P|} d_i^q\right|^{1/q}}{|P|} \tag{5.7}$$

Where, $d_i = \min_{\vec{S}\in S}\left\|F(\vec{P_i}) - F(\vec{S})\right\|$, $\vec{P_i} \in P$ and $q = 2$

d_i is a smallest distance from $\vec{P} \in P$ to the closet solution in *S*.

Instead of measuring the average distance in IGD, the maximum Pareto front error (MPFE) is defined as:

$$MPFE(P,S) = \max_{\vec{P}\in P} \sqrt{\min_{\vec{S}\in S} \sum_{k=1}^{m}\left|f_k(\vec{S}) - f_k(\vec{P})\right|^2} \tag{5.8}$$

This metric finds the maximum distance from solutions in *P* to the closest solution in *S*.

In order to satisfy the comparison purpose, the second simulation test problems for each algorithm at generation 500 with different seeds and with different number of jobs (20, 60 and 100) are represented in appendices A, B, C and D.

In appendix A, Tables 1-3 and Figures (Appendix A.1 - Appendix A.30) include the Pareto solutions for unrelated multi-objective parallel machine scheduling problem for 5 machines and (20, 60 and 100) jobs with crossover probability 0.6.

In Appendix B, Tables 1-3 and Figures (Appendix B.1 - Appendix B.30) consist of the Pareto solutions for unrelated multi-objective parallel machine scheduling problem for 5 machines and (20, 60 and 100) jobs with crossover probability 0.7.

In Appendix C, Tables1-3 and Figures (Appendix C.1 - Appendix C.30) consist of the Pareto solutions for unrelated multi-objective parallel machine scheduling problem for 5 machines and (20, 60 and 100) jobs with crossover probability 0.8.

Finally, in Appendix D, Tables1-3 and Figures (Appendix D.1 - Appendix D.30) contain the Pareto solutions for the same problem for 5 machines and (20, 60 and 100) jobs with crossover probability 0.9.

In general, the obtained results show the ability of each algorithm to determine the final non-dominated solutions but it cannot determine the best algorithm because the Pareto solutions are closed to each other. Therefore, the Diversity metrics (spacing diversity metric SP, distribution and spread diversity metric Δ) are selected from all previous measures because it dependent on the obtained Pareto front. Tables 5.19 and 5.20 represent the mean and variance of diversity metrics for 10 run trails to each algorithm.

Table 5. 19. *The mean of diversity metrics for non-dominated front to each algorithm for 10 runs (second test problems)*

Number of job	Crossover probability	Spacing Diversity Metric (SP)		Distribution and Spread Diversity Metric (Δ)	
		SJMCT-NSGA-II	SJMCT-SPEA-II	SJMCT-NSGA-II	SJMCT-SPEA-II
20	0.6	3.653	**2.415**	0.677	**0.631**
	0.7	4.445	**4.115**	0.717	**0.711**
	0.8	**1.925**	2.928	**0.601**	0.682
	0.9	3.362	**2.775**	0.644	**0.600**
60	0.6	8.352	**6.159**	0.702	**0.644**
	0.7	**4.803**	7.746	**0.617**	0.673
	0.8	**3.917**	5.497	**0.583**	0.621
	0.9	8.150	**7.388**	**0.686**	0.711
100	0.6	**11.946**	15.168	**0.751**	0.769
	0.7	13.681	**9.330**	0.763	**0.601**
	0.8	15.592	**9.175**	0.824	**0.762**
	0.9	10.726	**10.258**	0.783	**0.734**

Table 5. 20. *The variance of diversity metrics for non-dominated front to each algorithm for 10 runs (second test problems)*

Number of job	Crossover probability	Spacing Diversity Metric(SP)		Distribution and Spread Diversity Metric (Δ)	
		SJMCT-NSGA-II	SJMCT-SPEA-II	SJMCT-NSGA-II	SJMCT-SPEA-II
20	0.6	3.671	**1.645**	0.013	**0.007**
	0.7	8.983	**6.356**	0.033	**0.004**
	0.8	**2.460**	3.129	0.018	0.019
	0.9	2.963	**2.278**	0.024	**0.005**
60	0.6	22.399	**6.469**	**0.010**	0.045
	0.7	7.479	**7.159**	0.022	**0.004**
	0.8	**3.672**	13.873	**0.008**	0.044
	0.9	21.302	**11.560**	**0.011**	0.018
100	0.6	123.890	**116.747**	0.024	**0.016**
	0.7	70.393	**66.193**	**0.024**	0.100
	0.8	47.570	**33.945**	**0.021**	0.026
	0.9	**41.014**	67.066	0.018	**0.013**

Tables 5.19 and 5.20 consider the diversity metric values for the second test problems. In order to enhance the best performance the comparison between the two algorithms as follows:

- In Table 5.19 the smallest mean value of spacing metric is 1.925 in SJMCT-NSGA-II. That means, SJMCT-NSGA-II has the small distance at test with crossover probability 0.8 and with 20 jobs. While the smallest mean value equal to 2.415 in SJMCT-SPEA-II at test with crossover probability 0.6 and with 20 jobs.
- In Table 5.19 the smallest mean value of spread metric is 0.583 in SJMCT-NSGA-II at test with crossover probability 0.8 and with 60 jobs. Also, the smallest mean value 0.600 at test with crossover probability 0.9 and with 20 jobs.
- In Table 5.20 the smallest variance value of spacing metric is 1.645 at test with crossover probability 0.6 and with 20 jobs in SJMCT-SPEA-II. Also, it equel to 2.460 at test with crossover probability 0.8 and with 20 jobs in SJMCT-NSGA-II. Furthermore, the smallest variance value of spread metric is 0.004 at test with crossover probability 0.7 with 20 and 60 jobs in SJMCT-SPEA-II. While, it equals to 0.008 in SJMCT-NSGA-II algorithm at test with crossover probability 0.8 and with 60 jobs.

In other words, for each job Tables 5.19 and 5.20 can be explained as follows:

- For 20 jobs the mean and variance values of diversity metrics in SJMCT-SPEA-II is smaller than SJMCT-NSGA-II by 75% precent.
- For 60 jobs the mean and the variance values of spread metric in SJMCT-NSGA-II is smaller than SJMCT-SPEA-II by 75% precent. Also, the mean values of spacing metric in both algorithms equal to 50% precent and the variance values in 60 jobs of spacing metric in SJMCT-SPEA-II is smaller than SJMCT-NSGA-II by 75% precent.
- For 100 jobs the mean and the variance values of spacing metric in SJMCT-SPEA-II is smaller than SJMCT-NSGA-II by 75% precent. Also, and the mean values of spread metric in SJMCT-SPEA-II is smaller than SJMCT-NSGA-II by 75% precent and the variance values of spread metric in both algorithms equal to 50% precent.

According to the experimintal results, on most cases, SJMCT-SPEA-II algorithm is better than the SJMCT-NSGA-II algorithm based on the mean and variance values of diversity metrics. That means the SJMCT-SPEA-II algorithm outperformes than SJMCT-NSGA-II.

5.5. The time of implementation

In this section, the feasible running time during executing the proposed algorithms SJMCT-NSGA-II and SJMCT-SPEA-II for all jobs with respect to crossover probabilities (0.6, 0.7, 0.8 and 0.9) are given in Table 5.21.

Table 5. 21. *Time in second for the best solution to each algorithm for all the second test problems*

Number of jobs	Run	Crossover Prob. 0.6		Crossover Prob. 0.7		Crossover Prob. 0.8		Crossover Prob. 0.9	
		Time in second SJMCT-NSGA-II	Time in second SJMCT-SPEA-II	Time in second SJMCT-NSGA-II	Time in second SJMCT-SPEA-II	Time in second SJMCT-NSGA-II	Time in second SJMCT-SPEA-II	Time in second SJMCT-NSGA-II	Time in second SJMCT-SPEA-II
20	1	967.471	571.390	1058.079	689.900	1085.146	683.118	1106.223	721.679
	2	980.014	583.901	**1245.825**	626.850	1164.926	701.433	1059.521	646.834
	3	1104.007	635.357	1084.479	662.924	1065.065	641.520	1039.019	635.490
	4	998.972	680.116	1132.613	611.302	1084.335	737.335	1080.692	803.987
	5	1007.078	666.725	1174.816	645.403	1078.374	623.905	1109.272	680.713
	6	1026.416	580.067	1041.795	670.298	1102.594	694.027	1107.050	648.016
	7	952.461	586.704	1236.804	704.621	1202.466	727.222	1106.783	799.539
	8	998.456	592.224	1183.025	613.109	1024.877	800.919	1122.158	843.626
	9	1050.007	**552.026**	1094.522	786.645	1187.786	732.809	1126.215	790.031
	10	1115.048	557.017	1127.016	805.425	1081.074	766.746	1116.350	655.506
60	1	2046.756	2391.281	1957.160	1812.959	2120.108	**1587.196**	2246.470	1623.477
	2	2553.467	2273.719	2398.855	2037.075	2244.856	1713.006	2560.084	2133.748
	3	2374.902	2338.787	2411.098	2089.031	2543.429	2138.496	2633.918	2179.575
	4	2551.688	2315.890	2423.819	2070.147	2378.141	2180.710	2707.688	2234.209
	5	2581.791	2316.141	2409.290	2221.823	2334.725	2183.359	2658.536	2244.689
	6	2537.604	2400.167	2505.478	2199.132	**3551.508**	2459.062	2653.767	2226.643
	7	2590.568	2305.287	2520.043	2243.515	2798.671	2230.224	2615.379	2300.624
	8	2603.123	2317.850	2580.543	2204.830	2757.490	2277.624	2702.401	2257.214
	9	2508.108	2316.629	2640.918	2129.395	3306.445	2340.241	2741.937	2211.559
	10	2564.135	2372.476	2643.406	2181.727	2777.679	2181.572	2757.501	2211.103
100	1	2884.634	2777.606	3097.011	**2565.551**	2928.928	2990.946	3507.810	3146.403
	2	4026.407	3688.663	3020.067	3214.600	3876.872	3551.773	3987.180	3573.405
	3	4049.246	3486.287	4080.368	3676.209	4057.971	3664.288	4013.143	3708.259
	4	4089.591	3646.479	4147.180	3456.658	4115.976	3519.977	3962.430	3662.431
	5	4003.891	3713.938	4022.216	3716.522	4067.952	3727.317	4035.916	3598.593
	6	4059.865	3681.047	4046.430	3576.281	4143.247	3639.772	3972.169	3614.173
	7	**4151.922**	3620.690	4056.137	3727.200	3938.903	3672.502	4097.809	3693.397
	8	4099.609	3669.987	4039.054	3713.942	4086.299	3521.890	4055.563	3639.965
	9	3964.667	3659.561	4026.973	3592.622	4029.191	3760.300	4072.633	3679.504
	10	4087.830	3629.579	4100.362	3581.727	4120.590	3655.493	4085.536	3702.196

Figures 5.27-5.29 represent the starting and ending time in seconds to each algorithm for (20, 60 and 100) jobs respectively.

For 20 jobs, in view of Figure 5.27 and Table 5.21, the smallest time is **552.026** seconds in SJMCT-SPEA-II at crossover probability 0.6. Moreover, the largest time is **1245.825** seconds in SJMCT-NSGA-II at crossover probability 0.7.

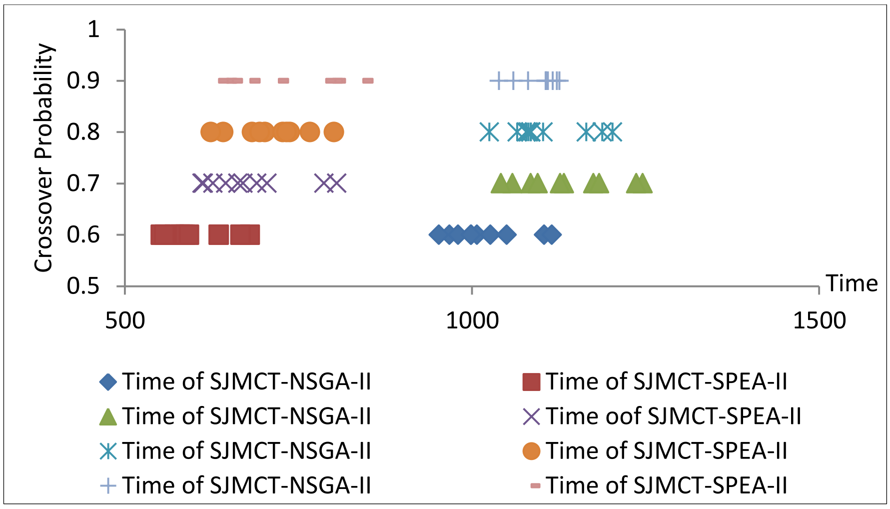


Figure 5. 27. *Time in second with all crossover probabilities for 20 jobs*

For 60 jobs, Figure 5.28 and Table 5.21 illustrate the smallest time is **1587.196** seconds in SJMCT-SPEA-II at crossover probability 0.8. The largest time is **3551.508** seconds in SJMCT-NSGA-II at crossover probability 0.8.

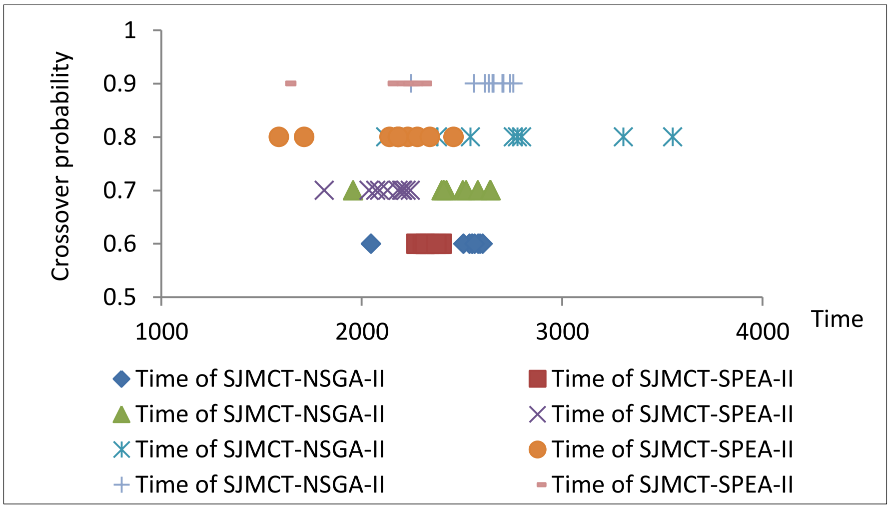


Figure 5. 28. *Time in second with all crossover probabilities for 60 jobs*

For 100 jobs, it can be observed from Figure 5.29 and Table 5.21, the smallest time is **2565.551** seconds in SJMCT-SPEA-II at crossover probability 0.7. The largest time is **4151.922** seconds in SJMCT-NSGA-II at crossover probability 0.6.

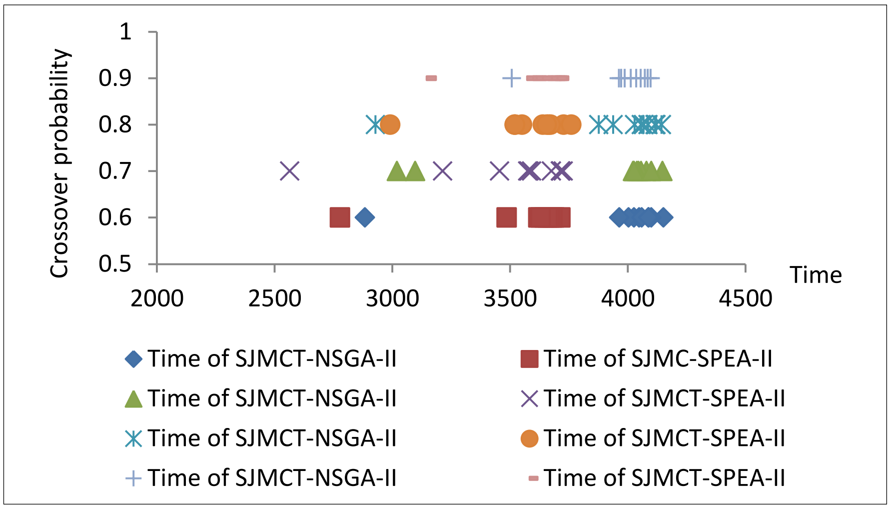


Figure 5. 29. *Time in second with all crossover probabilities for 100 jobs*

During the performance of the two algorithms, it is clear to see that SJMCT-SPEA-II algorithm has the smallest running time as compared with SJMCT-NSGA-II as seen in Table 5.21 and Figures 5.27-5.29.

6. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this thesis, a novel algorithm with name Sequence Job Minimum Completion Time (SJMCT) is proposed to represent the scheduling of unrelated parallel machines with non-identical jobs. The proposed algorithm was compared with other dispatching rules (LPT and SPT). Numerical example is solved by using GAMS-CPLEX programming to show the efficiency of proposed algorithm. The associated promising result with small size one objective problem is a motivation to use it with large size multi-objective problems.

As seen in the literature review, many real life multi-objective scheduling problems solved by mathematical programming, dispatching rules, neighborhood search, genetic and heuristic algorithms. Therefore, the main contribution of this thesis is to develop the multi-objective hybrid evolutionary algorithms and find the best Pareto front with more than one objective.

Two algorithms named Sequence Job Minimum Completion Time based on Non-dominated Sorting Genetic Algorithm (SJMCT-NSGA-II) and Sequence Job Minimum Completion Time based on Strength Pareto (SJMCT-SPEA-II) have been proposed to minimize the maximum completion time and the total tardiness.

The performance of the two algorithms SJMCT-NSGA-II and SJMCT-SPEA-II are tested by using MATLAB programming Version 8.3.0.532 (R2014a). It is interested to know, this program is suitable to solve large particular scheduling problem with small changes.

The proposed algorithms are able to find the best non-dominated Pareto front by each algorithm for big dimensional multi-objective parallel machine scheduling problem.

An intensive work of numerical experimentations has been performed. The first test problems are done with 5 parallel machines and 60 jobs and generation numbers from 40-500. The second test problems are done with 5 parallel machines and 20, 60 and 100 jobs and generation 500.

For most problems, several good solutions are introduced by changing the crossover and mutation probabilities.

To compare multi-objective evolutionary algorithms performance, we need to use some metrics. Therefore, the results of two algorithms have been compared by using two performance diversity metrics as spacing and spread metrics. In the simulation results of 60 jobs, a reasonably good minimum value of solutions and good spread are obtained at generation 500. Therefore, in order to observe the consistency of outcome of the proposed algorithms with different initial populations are selected at generation equals to 500.

During the performance evaluation of proposed algorithm, it is observed that, the SJMCT-SPEA-II algorithm has the smaller mean and variance values for each spacing and spread metrics in most of the second test problems. Also, the performance of SJMCT-SPEA-II has smallest running time than SJMCT-NSGA-II in second test problems. The smallest running time of SJMCT-SPEA-II was between 9 minutes at 20 jobs and 43 minutes at 100 jobs, while the running time of SJMCT-NSGA-II was between 21 minutes at 20 jobs and 69 minutes at 100 jobs.

In general, we conclude that, the proposed algorithm SJMCT has more convergence as compared with other algorithms in computing the total completion time of each machine. That means, it gives a good assignment of jobs at the machines and it make a good balance in workload over the parallel machines. In addition, there is no order forced to submit certain job. Also, the two hybrid algorithms are efficient and practical for solving large size problems. Moreover, SJMCT-SPEA-II has the highest quality performance than SJMCT-NSGA-II in both efficiency and the running time.

In future work, some comparison for the performance of proposed algorithm with other metaheuristic method can be done. It may also interest to apply other genetic operator (crossover and mutation) and generate a new different offspring. Also, other performance measures can be implemented as a future research direction.

Another future research direction is related with it could be interesting to develop other complex scheduling problems, such as flow shop problems, preceding constraints, deterioration or the machine with interrupted and unavailability periods. In addition, the current scheduling model can be developed by adding the rejection job constraint and rejection penalty.

Another opportunity for this research is the consideration of the problem with the other optimization objectives such as minimization of early and tardy penalties or weighted completion time and weighted tardiness. It also could be interesting to extend this study for more than two objectives.

REFERENCES

Akyol, D. (2006). Neural network based optimization in production scheduling. Ph.D.Thesis. Izmir: Dokuz Eylul University, Industrial Engineering.

Allahverdi, A., Gupta, J. N. D. and Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega* 27(2): 219–239.

Muralidhar, A. and Alwarsamy, T. (2013). Multi-objective optimization of parallel machine scheduling using neural networks. *International Journal of Latest Trends in Engineering and Technology (IJLTET)* 2(2): 127-132.

Arroyo, J.E.C. and Armentano, V.A. (2005). Genetic local search for multi-objective fowshop scheduling problems. *European Journal of Operational Research* 167(3). *Multicriteria Scheduling*: 717–738.

Bagchi, T.P. (2001). Pareto-optimal solutions for multi-objective production scheduling problems. Evolutionary multi-criterion optimization Pp. 458–471. *Springer, Berlin, Heidelberg*. https://link.springer.com/chapter/10.1007/3-540-44719-9_32, accessed March 17, 2017.

Balakrishnan, N., Kanet, J.J. and Sridharan, V. (1999). Early/tardy scheduling with sequence dependent setups on uniform parallel machines. *Computers & Operations Research* 26(2): 127–141.

Balasubramanian, H., Fowler, J., Keha, A. and Pfund, M. (2009). Scheduling interfering job sets on parallel machines. *European Journal of Operational Research* 199(1): 55–67.

Balin, S. (2011). Non-identical parallel machine scheduling using genetic algorithm. *Expert Systems with Applications* 38(6): 6814–6821.

Bandyopadhyay, S. and Bhattacharya, R. (2013). Solving multi-objective parallel machine scheduling problem by a modified NSGA-II. *Applied Mathematical Modelling* 37(10–11): 6718–6729.

Chand, P. and Mohanty, J.R. (2013). Solving vehicle routing problem with proposed non-dominated sorting genetic algorithm and comparison with classical evolutionary algorithms. *International Journal of Computer Applications* 69(26): 34–41.

Chang, P-C., Chen, S-H. and Hsieh J-C. (2006). A global archive sub-population genetic algorithm with adaptive strategy in multi-objective parallel-machine scheduling problem. *Advances in Natural Computation* Pp. 730–739. Springer, Berlin, Heidelberg.://link.springer.com/chapter/10.1007/11881070_98, accessed March 17, 2017.

Cheng, B., Yang S., Hu, X. and Chen, B. (2012). Minimizing makespan and total completion time for parallel batch processing machines with non-identical job sizes. *Applied Mathematical Modelling* 36(7): 3161–3167.

Cheng, R. and Gen, M. (1997). Parallel machine scheduling problems using memetic algorithms. *Computers & Industrial Engineering* 33(3). Selected Papers from the Proceedings of 1996 ICC&IC: 761–764.

Cochran, J.K., Horng, S-M. and Fowler, J.W. (2003). A Multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. *Computers & Operations Research* 30(7): 1087–1102.

Crauwels, H.A.J., Potts, C.N. and Van Wassenhove L.N. (1998). Local search heuristics for the single machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing* 10(3): 341–350.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2): 182–197.

Deb, K., Agrawal, S. Pratap, A. and Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *Parallel Problem Solving from Nature PPSN VI* Pp. 849–858. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. https://link.springer.com/chapter/10.1007/3-540-45356-3_83, accessed November 15, 2017.

Dyer, M.E. and Wolsey, L.A. (1990). Formulating the Single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics* 26(2): 255–270.

Ehrgott, M. (2006). Multicriteria optimization, 2nd Edition. *Springer science & business media.*

Eroglu, D.Y., Ozmutlu, H.C. and Ozmutlu, S. (2014). Genetic algorithm with local search for the unrelated parallel machine scheduling problem with sequence-dependent set-up times. *International Journal of Production Research* 52(19): 5841–5856.

Fallah-Mehdipour, E., Haddad, O.B., Tabari, M.M.R. and Mariño, M.A. (2012). Extraction of decision alternatives in construction management projects: application and adaptation of NSGA-II and MOPSO. *Expert Systems with Applications* 39(3): 2794–2803.

França, P.M., Mendes, A. and Moscato, P. (2001). A memetic algorithm for the total tardiness single machine scheduling problem. *European Journal of Operational Research* 132(1): 224–242.

Frenk, J.B.G., and Rinnooy Kan, A.H.G. (1987). The asymptotic optimality of the LPT rule. *Mathematics of Operations Research* 12(2): 241–254.

Gharari, R., Poursalehi, N., Abbasi, M., and Aghaie, M. (2016). Implementation of strength Pareto evolutionary algorithm II in the multi-objective burnable poison placement optimization of KWU pressurized water reactor. *Nuclear Engineering and Technology* 48(5): 1126–1139.

Ghosh, A. and Das, M.K. (2008). Non-dominated rank based sorting genetic algorithms. *Fundamenta Informaticae* 83(3): 231–252.

Godinez, A.C., Espinosa, L.E.M. and Montes, E.M. (2010). An experimental comparison of multi-objective algorithms: NSGA-II and OMOPSO. *2010 IEEE Electronics, Robotics and Automotive Mechanics Conference* Pp. 28–33.

Gupta, J.N.D. and Ho, J.C. (2001). Minimizing makespan subject to minimum flowtime on two identical parallel machines. *Computers & Operations Research* 28(7): 705–717.

Hong, T-P., Huang, C-M. and Yu, K-M. (1998). LPT scheduling for fuzzy tasks. Fuzzy Sets and Systems 97(3): 277-286.

Ishibuchi, H. and Murata, T. (1998). A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 28(3): 392–403.

Jaszkiewicz, A. (2002). Genetic local search for multi-objective combinatorial optimization. *European Journal of Operational Research* 137(1): 50–71.

Jiang, S., Ong, Y-S., Zhang, J. and Feng, L. (2014). Consistencies and contradictions of performance metrics in multiobjective optimization. *IEEE Transactions on Cybernetics* 44(12): 2391–2404.

Joo, C.M. and Kim, B.S. (2015). Hybrid genetic algorithms with dispatching rules for unrelated parallel machine scheduling with setup time and production availability. *Computers & Industrial Engineering* 85: 102–109.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P. and Werner, F. (2008). Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *The International Journal of Advanced Manufacturing Technology* 37(3–4): 354–370.

Kacem, I., Hammadi, S. and Borne, P. (2002). Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 32(1): 1–13.

Kamisli Ozturk, Z. and Sabti, A. (2017). Novel solution approaches for multi-objective parallel machine scheduling. 21st Conference of the international federation of operational research societies. Quebec, Canada / 17.07.2017-21.07.2017.

Kasimbeyli, R., Ozturk, Z.K., Kasimbeyli, N., Yalcin, G.D. and Icmen, B. (2015). Conic scalarization method in multiobjective optimization and relations with other scalarization methods. *Modelling, Computation and Optimization in Information Systems and Management Sciences* Pp. 319–329. Advances in Intelligent Systems and Computing. Springer, Cham. https://link.springer.com/chapter/10.1007/978-3-319-18161-5_27, accessed November 5, 2017.

Koulamas, C. and Kyparisis, G.J. (2009). A modified LPT algorithm for the two uniform parallel machine makespan minimization problem. *European Journal of Operational Research* 196(1): 61–68.

Laguna, M., Barnes, J.W. and Glover, F.W. (1991). Tabu search methods for a single machine scheduling problem. *Journal of Intelligent Manufacturing* 2(2): 63–73.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. and Shmoys, D.B. (1993). Chapter 9 sequencing and scheduling: algorithms and complexity. Handbooks in Operations Research and Management Science Pp. 445–522. Logistics of Production and Inventory. Elsevier. http://www.sciencedirect.com/science/ article/pii/S0927050705801896.

Lei, D. (2009). Multi-objective production scheduling: A survey. *The International Journal of Advanced Manufacturing Technology* 43(9–10): 926.

Li, X., Amodeo, L., Yalaoui, F. and Chehade, H. (2010).A multiobjective optimization approach to solve a parallel machines scheduling problem. *Adv. in Artif. Intell*. 2010: 2:1–2:10.

Li, X., Yalaoui, F., Amodeo, L. and Chehade, H. (2012). Metaheuristics and exact methods to solve a multiobjective parallel machines scheduling problem. *Journal of Intelligent Manufacturing* 23(4): 1179–1194.

Lin, C-H., and Liao, C-J. (2008). Makespan minimization for multiple uniform machines. *Computers & Industrial Engineering* 54(4): 983–992.

Lin, Y-K. and Lin, H-C. (2015). Bicriteria scheduling problem for unrelated parallel machines with release dates. *Computers & Operations Research* 64: 28–39.

Loukil, T., Teghem, J. and Tuyttens, D. (2005). Solving multi-objective production scheduling problems using metaheuristics. *European Journal of Operational Research* 161(1). IEPM: Focus on Scheduling: 42–61.

Lu, L., Zhang, L. and Yuan, J. (2008). The unbounded parallel batch machine scheduling with release dates and rejection to minimize makespan. *Theoretical Computer Science* 396(1): 283–289.

Ma, Y., Chu, C. and Zuo, C. (2010). A Survey of scheduling with deterministic machine availability constraints. *Computers & Industrial Engineering* 58(2). Scheduling in Healthcare and Industrial Systems: 199–211.

Massabò, I., Paletta, G. and Ruiz-Torres, A.J. (2016). A note on longest processing time algorithms for the two uniform parallel machine makespan minimization problem. *Journal of Scheduling* 19(2): 207–211.

Mazdeh, M.M., Zaerpour, F., Zareei, A. and Hajinezhad, A. (2010). Parallel machines scheduling to minimize job tardiness and machine deteriorating cost with deteriorating jobs. *Applied Mathematical Modelling* 34(6): 1498–1510.

Memari, A., Rahim, A.R.A., Hassan, A. and Ahmad, R. (2016). A tuned NSGA-II to optimize the total cost and service level for a just-in-time distribution network. *Neural Computing and Applications*: 1–15.

Mirabedini, S.N. and Mina, H. (2012). Multi-objective optimization research on multi parallel machine with different preventive maintenance planning and scheduling with genetic algorithm. *International Journal of Academic Research in Business and Social Sciences* 2(12): 129–139.

Mishra, B. and Patnaik R.K. (2009). Genetic Algorithm and Its Variants: Theory and Applications. BTech. http://ethesis.nitrkl.ac.in/199/ accessed January 9, 2018.

Moslehi, G. and Mahnam, M. (2011). A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics* 129(1): 14–22.

Murata, T., Ishibuchi, H. and Tanaka, H. (1996). Multi-objective genetic algorithm and its applications to flowshop scheduling. *Computers & Industrial Engineering* 30(4): 957–968.

Ou, J., Zhong, X. and Wang, G. (2015). An improved heuristic for parallel machine scheduling with rejection. *European Journal of Operational Research* 241(3): 653–661.

Öztürk, C. and Ornek, A.M. (2014). Operational extended model formulations for advanced planning and scheduling systems. *Applied Mathematical Modelling* 38(1): 181–195.

Parker, R. G.(1996). Deterministic scheduling theory. CRC Press.

Pinedo, M. L. (2008). Scheduling: Theory, algorithms, and systems. 3rd edition. Springer Publishing Company, Incorporated.

Pinedo, M.L. (2005). Planning and scheduling in manufacturing and services. Springer Series in Operations Research. New York: Springer-Verlag. http://link.springer.com/10.1007/b139030.

Potts, C.N. (1985). Analysis of a linear programming heuristic for scheduling unrelated parallel machines. Discrete Applied Mathematics 10(2): 155–164.

Rajendran, C. and Ziegler, H. (2003). Scheduling to minimize the sum of weighted flowtime and weighted tardiness of jobs in a flowshop with sequence-dependent setup times. *European Journal of Operational Research* 149(3): 513–522.

Reddy M.J., and Kumar D.N. (2007). Multi-objective differential evolution with application to reservoir system optimization. *Journal of Computing in Civil Engineering* 21(2): 136–146.

Ruiz, R. and Andrés-Romano, C. (2011). Scheduling unrelated parallel machines with resource-assignable sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology* 57(5–8): 777–794.

Schott, J.R. (1995). Fault tolerant design using single and multi-criteria genetic algorithm optimization. AFIT/CI/CIA-95-039. air force inst of tech wright-patterson afb oh, air force inst of tech wright-patterson afb oh. http://www.dtic.mil/docs/citations/ADA296310.

Senthiil, P.V., Selladurai, V. and Rajesh, R. (2007). Parallel machine scheduling (pms) in manufacturing systems using the ant colonies optimization algorithmic rule. Journal of Applied Sciences 7: 208–213.

Seshadri, A., (2006). Multi-objective optimization using evolutionary algorithms (MOEA). https://www.scribd.com/document/263374723/NSGA-2-tutorial.

Sivrikaya-Şerifoğlu, F. and Ulusoy, G. (1999). Parallel machine scheduling with earliness and tardiness penalties. *Computers & Operations Research* 26(8): 773–787.

Srinivas, N. and Deb, K. (1994). Multi-objective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation* 2: 221–248.

Strusevich, V.A. and Rustogi, K. (2017). Scheduling with time-changing effects and rate-modifying activities, vol.243. International Series in Operations Research & Management Science. Cham: Springer International Publishing. http://link.springer.com/10.1007/978-3-319-39574-6.

Suresh, V. and Chaudhuri, D. (1996). Bicriteria scheduling problem for unrelated parallel machines. *Computers & Industrial Engineering* 30(1): 77–82.

Tavakkoli-Moghaddam, R., Taheri, F. and Bazzazi, M. (2008). Multi-objective unrelated parallel machines scheduling with sequence-dependent setup times and precedence constraints. *International Journal of Engineering - Transactions A: Basics* 21(3): 269-278.

Torabi, S.A., Sahebjamnia, N., Mansouri, S. A. and Bajestani, M.A. (2013). A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem. *Applied Soft Computing* 13(12): 4750–4762.

Uma, R.N., Wein, J. and Williamson, D.P. (2006). On the relationship between combinatorial and LP-based lower bounds for NP-hard scheduling problems. *Theoretical Computer Science* 361(2). 241–256.

Unlu, Y. and Mason, S.J. (2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Comput. Ind. Eng*. 58(4): 785–800.

Vallada, E. and Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research* 211(3): 612–622.

Veldhuizen, D.A.V. and Lamont, G.B. (2000). On measuring multi-objective evolutionary algorithm performance. *In* Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512) Pp. 204–211 vol.1.

Wang, S. and Liu, M. (2015). Multi-objective optimization of parallel machine scheduling integrated with multi-resources preventive maintenance planning. *Journal of Manufacturing Systems* 37, Part 1: 182–192.

Weng, M.X., Lu, J. and Ren, H. (2001). Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. *International Journal of Production Economics* 70(3): 215–226.

Wu, J. and Azarm, S. (2000). Metrics for quality assessment of a multi-objective design optimization solution set. *Journal of Mechanical Design* 123(1): 18–25.

Xing, W. and Zhang, J. (2000). Parallel machine scheduling with splitting jobs. *Discrete Applied Mathematics* 103(1–3): 259–269.

Yang, S-J. (2013). Unrelated parallel-machine scheduling with deterioration effects and deteriorating multi-maintenance activities for minimizing the total completion time. *Applied Mathematical Modelling* 37(5): 2995–3005.

Yazdani, M., Amiri, M. and Zandieh, M. (2010). Flexible job-shop scheduling with parallel variable neighborhood search algorithm. *Expert Systems with Applications* 37(1): 678–687.

Yeh, W-C., Chuang, M-C. and Lee, W-C. (2015). Uniform parallel machine scheduling with resource consumption constraint. *Applied Mathematical Modelling* 39(8): 2131–2138.

Yeh, W-C., Lai, P-J., Lee, W-C. and Chuang, M-C. (2014). Parallel-machine scheduling to minimize makespan with fuzzy processing times and learning effects. *Information Sciences* 269: 142–158.

Yusoff, Y., Ngadiman, M.S. and Zain, A.M. (2011). Overview of NSGA-II for optimizing machining process parameters. *Procedia Engineering 15*. CEIS 2011: 3978–3983.

Zhang, M. and Luo, C. (2013). Parallel-machine scheduling with deteriorating jobs, rejection and a fixed non-availability interval. *Applied Mathematics and Computation* 224: 405–411.

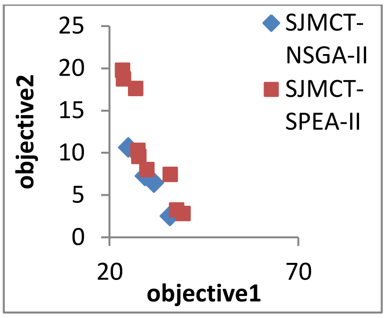
Zitzler, E., Laumanns, M. and Thiele, L. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. http://kdd.cs.ksu.edu/Courses/Spring-2007/ CIS830 / Handouts/P8.pdf

APPENDIX A

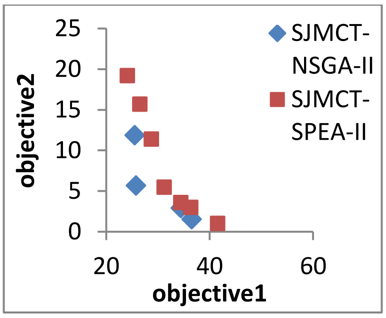
Simulation results for second test problems to each algorithm for 5 machines and **20 jobs**. The values of the best non-dominated front at generation 500 with number of population are 100 and **crossover probability 0.6** as given in APPENDIX A. Table 1.

APPENDIX A. Table 1 *The values of the best non-dominated front for **20 jobs** to each algorithm at crossover probability 0.6*

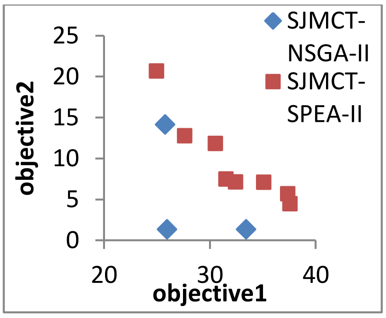
Run	Job	Crossover probability	SJMCT- NSGA-II Objective1	SJMCT- NSGA-II Objective2	SJMCT-SPEA-II Objective1	SJMCT-SPEA-II Objective2
1	20	0.6	25.021	10.590	23.496	19.752
			35.994	2.487	23.717	18.722
			31.792	6.433	27.585	10.272
			29.422	7.226	27.809	9.549
					26.970	17.568
					39.481	2.802
					29.906	7.988
					36.069	7.421
					37.767	3.207
2	20	0.6	36.556	1.501	24.120	19.202
			25.589	11.867	41.584	1.006
			25.786	5.658	26.543	15.684
			34.456	2.882	31.245	5.443
					34.463	3.556
					28.755	11.392
					36.421	2.981
3	20	0.6	33.437	1.307	24.976	20.661
			25.807	14.119	27.641	12.737
			25.986	1.339	37.578	4.470
					31.524	7.458
					37.366	5.660
					32.432	7.122
					30.530	11.831
					35.096	7.081



Appendix A.1. Solutions at run1 for 20 jobs (Crossover prob. 0.6)



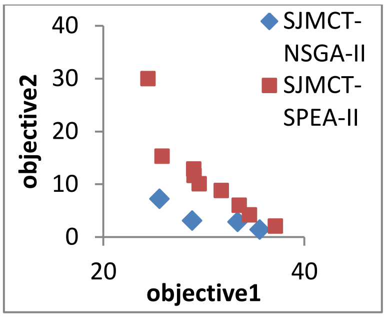
Appendix A.2. Solutions at run 2 for 20 jobs (Crossover prob. 0.6)



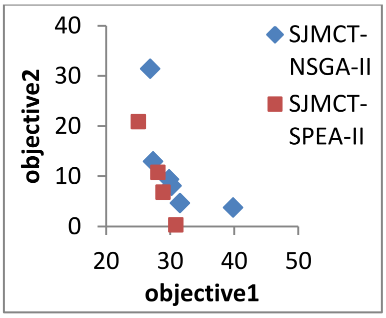
Appendix A.3. Solutions at run 3 for 20 jobs (Crossover prob. 0.6)

APPENDIX A. Table 1 (Continue) *The values of the best non-dominated front for* ***20 jobs*** *to each algorithm at crossover probability 0.6*

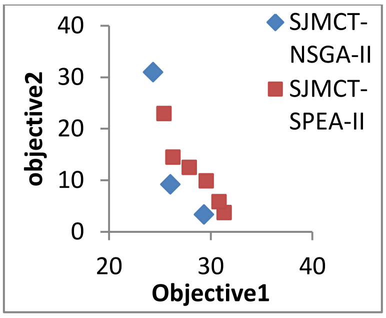
Run	Job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	*Objective2*	*Objective1*	*Objective2*
4	20	0.6	25.620	7.221	24.479	29.934
			35.584	1.404	25.863	15.282
			28.848	3.104	37.128	2.054
			33.376	2.820	34.548	4.192
					29.064	11.643
					29.558	10.066
					33.551	5.989
					29.015	12.835
					31.767	8.807
5	20	0.6	26.918	31.410	25.090	20.867
			39.866	3.733	30.905	0.269
			27.405	12.943	28.151	10.735
			31.592	4.619	28.930	6.813
			29.870	9.353		
			30.304	8.080		
6	20	0.6	29.360	3.311	25.439	22.909
			24.366	30.934	26.322	14.422
			26.055	9.143	27.911	12.467
					31.340	3.688
					30.847	5.786
					29.581	9.851
7	20	0.6	26.183	22.040	25.224	29.298
			37.430	5.047	35.398	1.549
			32.107	5.230	27.448	9.878
			28.700	16.036	31.667	3.838
			29.159	10.500	29.118	6.381
			30.914	6.028	27.838	9.861
			29.807	9.226	30.847	4.944
			28.111	20.052	26.694	28.497
			28.474	18.516	26.965	20.485
			26.771	21.472		
			29.673	9.989		



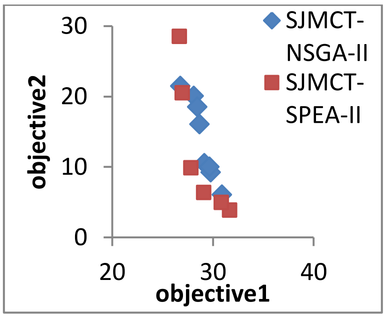
Appendix A.4. Solutions at run 4 for 20 jobs(Crossover prob. 0.6)



Appendix A.5. Solutions at run 5 for 20 jobs(Crossover prob. 0.6)



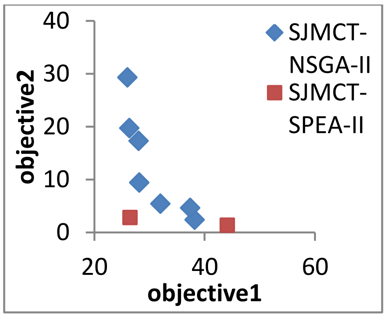
Appendix A.6. Solutions at run 6 for 20 jobs (Crossover prob. 0.6)



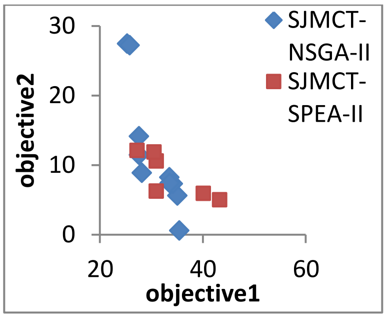
Appendix A.7. Solutions at run 7 for 20 jobs (Crossover prob. 0.6)

APPENDIX A. Table 1 (Continue) *The values of the best non-dominated front for **20 jobs** to each algorithm at crossover probability 0.6*

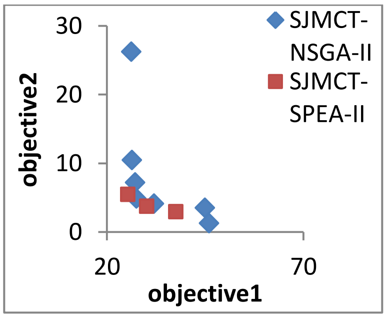
Run	Job	Crossover probability	SJMCT- NSGA-II Objective1	SJMCT- NSGA-II Objective2	SJMCT-SPEA-II Objective1	SJMCT-SPEA-II Objective2
8	20	0.6	38.196	2.347	44.108	1.336
			26.028	29.290	26.523	2.812
			31.997	5.407		
			28.205	9.432		
			37.412	4.612		
			26.381	19.747		
			28.046	17.310		
9	20	0.6	25.321	27.425	43.291	4.991
			35.420	0.593	40.109	5.936
			27.578	14.107	27.233	12.092
			25.798	27.216	30.980	6.219
			28.137	8.861	30.929	10.556
			33.479	8.213	30.505	11.855
			35.077	5.583		
			27.603	11.448		
			34.166	7.319		
			33.554	7.412		
10	20	0.6	46.032	1.265	25.374	5.488
			26.268	26.242	30.149	3.738
			32.011	4.117	37.530	2.945
			45.017	3.529		
			26.445	10.488		
			27.573	4.872		
			27.201	7.199		



Appendix A.8. Solutions at run 8 for 20 jobs (Crossover prob. 0.6)



Appendix A.9. Solutions at run 9 for 20 jobs (Crossover prob. 0.6)

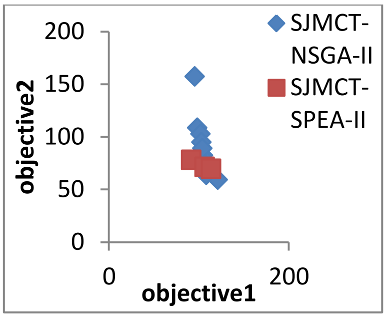


Appendix A.10. Solutions at run 10 for 20 jobs (Crossover prob. 0.6)

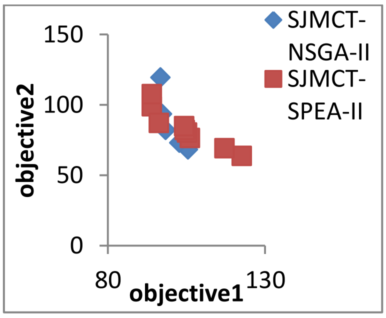
Simulation results for second test problems to each algorithm for 5 machines and **60 jobs**. The values of the best non-dominated front at generation 500 with number of population are 100 and **crossover probability 0.6** as given in APPENDIX A. Table 2.

APPENDIX A. Table 2 *The values of the best Non-dominated front for* ***60 jobs*** *to each algorithm at crossover probability 0.6*

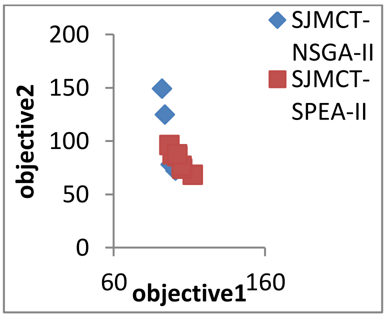
Run	Job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	*Objective2*	*Objective1*	*Objective2*
1	60	0.6	121.113	59.114	91.587	78.141
			95.650	157.273	106.759	71.317
			98.355	108.603	113.615	69.618
			108.409	63.721		
			101.888	102.758		
			104.677	82.166		
			103.079	94.841		
			107.792	77.312		
			104.072	89.047		
			108.035	73.248		
2	60	0.6	96.736	119.493	94.009	98.918
			105.499	68.327	96.218	87.134
			102.750	73.093	122.600	63.796
			98.352	82.241	93.975	107.597
			97.272	93.487	106.122	76.479
					105.219	80.293
					104.487	82.218
					117.126	69.234
					104.248	84.974
3	60	0.6	92.030	149.055	112.328	68.482
			100.803	72.282	99.014	87.615
			93.857	124.826	96.924	96.352
			97.739	78.066	105.360	74.060
					102.220	85.967
					104.823	76.788
					101.953	87.612



Appendix A.11. Solutions at run 1 for 60 jobs (Crossover prob. 0.6)



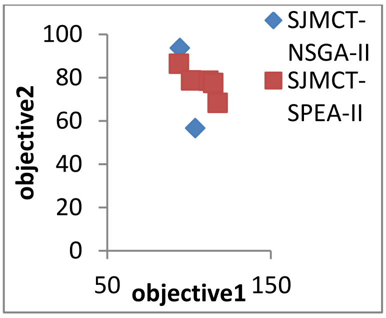
Appendix A.12. Solutions at run 2 for 60 jobs (Crossover prob. 0.6)



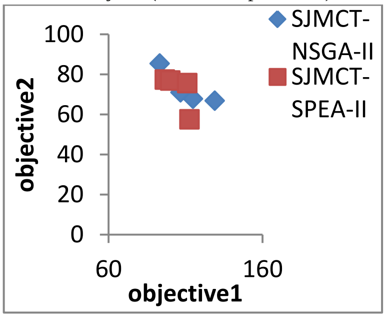
Appendix A.13. Solutions at run 3 for 60 jobs (Crossover prob. 0.6)

APPENDIX A. Table 2 (Continue) *The values of the best Non-dominated front for **60 jobs** to each algorithm at crossover probability 0.6*

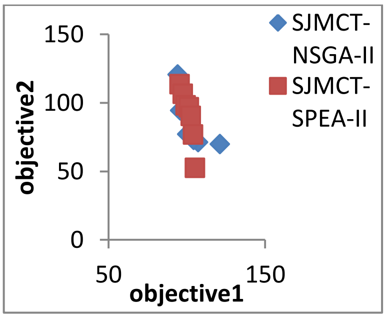
Run	Job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	*Objective2*	*Objective1*	*Objective2*
4	60	0.6	94.642	93.708	93.984	86.393
			103.948	56.587	101.320	78.663
					117.544	68.344
					111.868	78.541
					114.928	77.468
5	60	0.6	93.347	85.354	96.814	77.332
			129.167	66.819	112.821	57.596
			105.110	74.072	100.306	76.899
			115.067	67.858	111.340	75.652
			107.053	70.942		
6	60	0.6	121.290	69.968	105.385	52.590
			94.213	120.628	95.500	113.680
			95.976	94.404	97.644	106.724
			100.401	77.097	99.359	98.362
			107.462	71.268	104.217	76.864
			104.604	72.912	101.531	96.879
					102.772	90.545
7	60	0.6	95.857	139.358	91.216	66.180
			130.800	71.350		
			104.574	72.307		
			99.109	115.622		
			101.735	81.098		
			100.126	95.858		
			101.239	95.627		



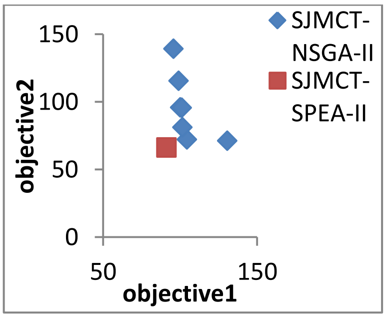
Appendix A.14. Solutions at run 4 for 60 jobs (Crossover prob. 0.6)



Appendix A.15. Solutions at run 5 for 60 jobs (Crossover prob. 0.6)



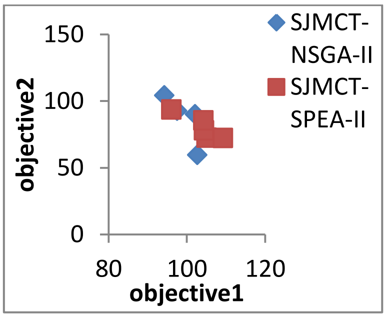
Appendix A.16. Solutions at run 6 for 60 jobs (Crossover prob. 0.6)



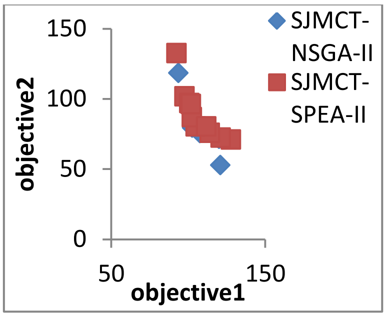
Appendix A.17. Solutions at run 7 for 60 jobs (Crossover prob. 0.6)

APPENDIX A. Table 2 (Continue) *The values of the best non-dominated front for **60 jobs** to each algorithm at crossover probability 0.6.*

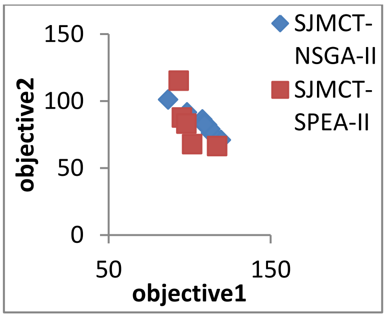
Run	Job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	*Objective2*	*Objective1*	*Objective2*
8	60	0.6	94.336	104.296	96.095	93.662
			102.754	59.522	105.106	72.494
			102.155	90.233	104.461	78.103
			97.572	92.709	109.237	72.291
					104.307	85.480
9	60	0.6	93.808	118.336	127.836	70.998
			121.025	52.721	92.481	132.596
			120.227	71.409	121.189	72.247
			107.624	75.554	97.634	101.831
			98.222	100.635	114.165	75.737
			102.768	79.327	105.097	80.293
			100.862	91.676	100.887	97.045
					102.481	86.818
					111.800	80.194
					101.854	96.160
10	60	0.6	119.581	70.939	93.406	115.177
			86.839	101.173	95.474	87.660
			98.558	91.515	98.270	83.079
			108.070	86.177	101.747	67.711
			113.419	76.239	117.240	66.243
			112.000	79.988		
			111.006	82.272		
			113.069	78.986		
			108.475	83.400		
			109.380	82.740		



Appendix A.18. Solutions at run8 for 60 jobs (Crossover prob. 0.6)



Appendix A.19. Solutions at run 9 for 60 jobs (Crossover prob. 0.6)

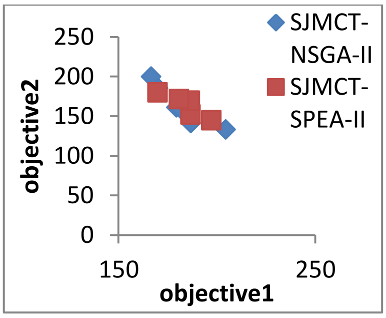


Appendix A.20. Solutions at run 10 for 60 jobs (Crossover prob. 0.6)

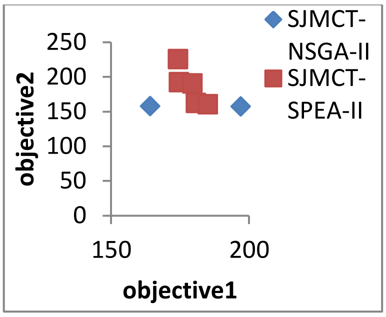
Simulation results for second test problems to each algorithm for 5 machines and **100 jobs**. The values of the best non-dominated front at generation 500 with number of population are 100 and **crossover probability 0.6** as given in APPENDIX A. Table 3.

APPENDIX A. Table 3 *The values of the best Non-dominated front for **100 jobs** to each algorithm at crossover probability 0.6.*

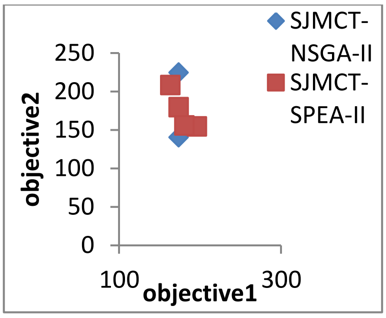
Run	Job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	*Objective2*	*Objective1*	*Objective2*
1	100	0.6	166.570	199.784	169.921	179.988
			204.570	133.042	197.226	145.079
			186.807	141.220	186.611	152.402
			170.260	185.250	186.402	169.524
			179.558	160.923	180.864	171.506
			181.668	159.463		
2	100	0.6	164.236	157.312	180.764	162.159
			197.090	157.086	174.666	192.319
					185.072	160.156
					179.480	190.181
					174.270	225.228
3	100	0.6	173.692	140.047	163.607	208.228
			173.649	224.271	196.345	154.137
					173.575	179.472
					180.795	155.445



Appendix A.21. Solutions at run 1 for 100 jobs (Crossover prob. 0.6)



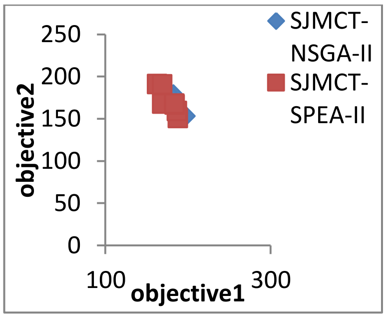
Appendix A.22. Solutions at run 2 for 100 jobs (Crossover prob. 0.6)



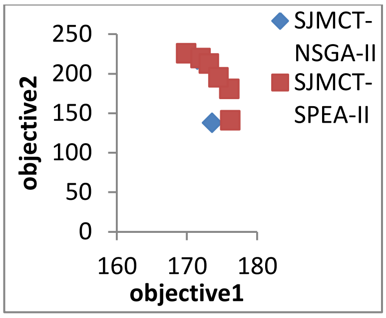
Appendix A.23. Solutions at run 3 for 100 jobs (Crossover prob. 0.6)

APPENDIX A. Table 3 (Continue): *The values of the best Non-dominated front for **100 jobs** to each algorithm at crossover probability 0.6*

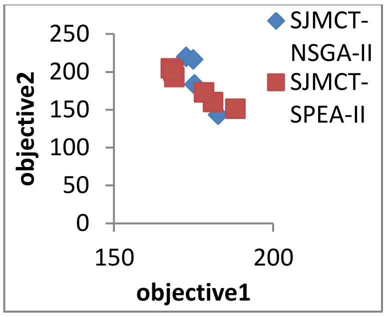
Run	Job	Crossover probability	SJMCT- NSGA-II *Objective1*	SJMCT- NSGA-II *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
4	100	0.6	165.803	186.365	162.885	191.022
			197.053	153.142	169.108	168.015
			183.169	164.118	168.766	190.730
			182.350	178.628	188.067	151.104
			188.575	156.590	186.835	159.354
			175.139	181.452	184.177	166.857
					183.792	167.998
5	100	0.6	173.596	137.552	176.217	140.846
			171.493	217.320	169.939	225.698
			172.729	215.052	171.955	219.415
					173.145	212.482
					176.068	180.736
					174.496	195.294
6	100	0.6	172.674	219.376	168.961	192.934
			182.761	142.934	167.882	204.417
			175.380	183.000	168.692	200.662
			174.961	216.079	188.229	150.820
					181.139	159.952
					178.316	172.434
7	100	0.6	197.962	143.994	173.478	141.029
			172.523	240.573	172.240	259.055
			182.935	149.655	173.442	243.230
			180.414	183.039		
			176.939	201.635		
			175.541	217.850		
			173.421	222.900		
			178.639	188.585		



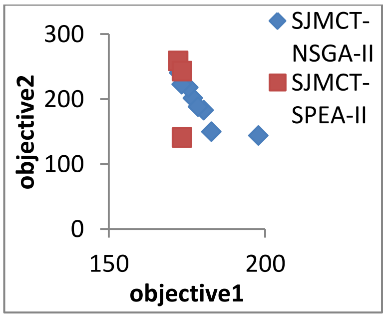
Appendix A.24. Solutions at run 4 for 100 jobs (Crossover prob. 0.6)



Appendix A.25. Solutions at run 5 for 100 jobs (Crossover prob. 0.6)



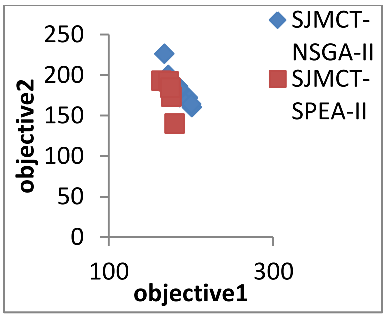
Appendix A.26. Solutions at run 6 for 100 jobs (Crossover prob. 0.6)



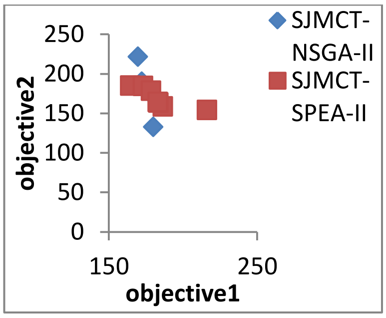
Appendix A.27. Solutions at run7 for 100 jobs (Crossover prob. 0.6)

APPENDIX A. Table 3 (Continue) *The values of the best Non-dominated front for **100 jobs** to each algorithm at crossover probability 0.6*

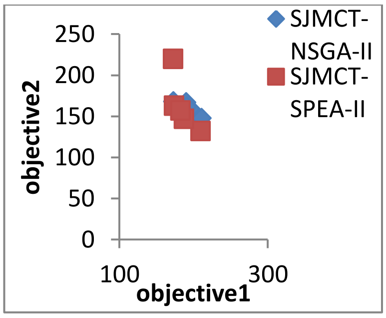
Run	Job	Crossover probability	SJMCT- NSGA-II		SJMCT-SPEA-II	
			Objective1	*Objective2*	*Objective1*	*Objective2*
8	100	0.6	168.075	226.019	164.324	193.095
			201.622	160.239	180.455	139.998
			172.487	200.299	172.984	192.372
			196.882	172.141	176.446	173.452
			186.693	172.449	175.268	184.221
			174.996	187.367		
			200.634	164.065		
			179.674	187.122		
			184.861	185.782		
			186.367	180.661		
			185.507	183.130		
9	100	0.6	169.695	221.387	216.391	154.145
			180.177	132.583	164.668	185.129
			172.349	189.802	173.275	184.790
			177.557	175.920	186.492	158.833
			177.549	181.623	178.630	178.511
					183.535	163.475
					183.046	164.301
10	100	0.6	168.075	226.019	209.848	132.051
			201.622	160.239	173.907	162.895
			172.487	200.299	187.358	146.989
			196.882	172.141	172.535	220.062
			186.693	172.449	182.465	157.111
			174.996	187.367		
			200.634	164.065		
			179.674	187.122		
			184.861	185.782		
			186.367	180.661		
			185.507	183.130		



Appendix A.28. Solutions at run 8 for 100 jobs (Crossover prob. 0.6)



Appendix A.29. Solutions at run 9 for 100 jobs (Crossover prob. 0.6)



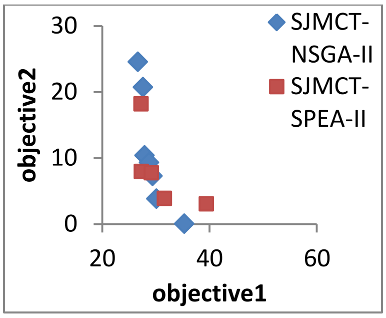
Appendix A.30. Solutions at run 10 for 100 jobs (Crossover prob. 0.6)

APPENDIX B

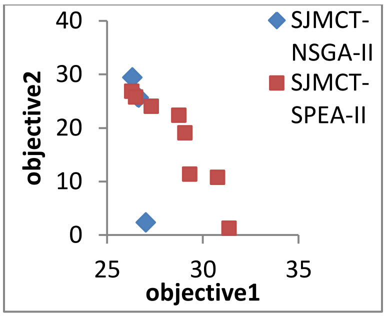
Simulation results for second test problems to each algorithm for 5 machines and **20 jobs**. The values of the best non-dominated front at generation 500 with number of population are 100 and **crossover probability 0.7** as given in APPENDIX B. Table 1.

APPENDIX B. Table 1 *The values of the best non-dominated front for **20 jobs** to each algorithm at crossover probability 0.7*

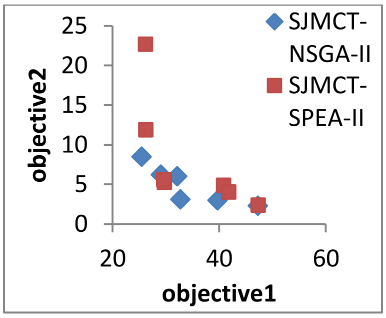
Run	Job	Crossover probability	SJMCT- NSGA-II *Objective1*	SJMCT- NSGA-II *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
1	20	0.7	35.349	0.000	27.316	7.933
			26.694	24.531	29.174	7.720
			30.125	3.813	27.295	18.160
			27.654	20.671	39.462	3.024
			27.912	10.350	31.639	3.831
			29.422	7.226		
			28.787	9.299		
2	20	0.7	26.326	29.389	31.368	1.278
			27.026	2.310	29.314	11.370
			26.662	25.616	26.299	26.816
			26.592	25.750	26.492	25.744
					30.764	10.742
					29.066	19.026
					27.306	23.990
					28.753	22.353
3	20	0.7	47.296	2.279	47.367	2.360
			25.508	8.475	40.858	4.864
			32.780	3.077	41.795	4.030
			39.749	2.954	26.289	11.877
			29.088	6.203	26.208	22.651
			32.164	6.001	29.760	5.225
					29.616	5.606



Appendix B.1. Solutions at run1 for 20 jobs (Crossover prob. 0.7)



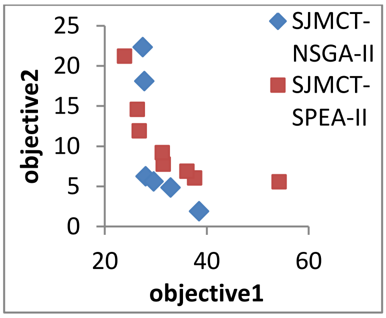
Appendix B.2. Solutions at run 2 for 20 jobs (Crossover prob. 0.7)



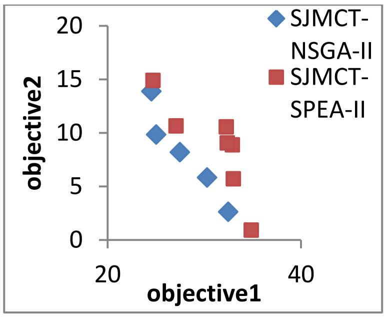
Appendix B.3. Solutions at run 3 for 20 jobs (Crossover prob. 0.7)

APPENDIX B. Table 1 (Continue) *The values of the best non-dominated front for **20 jobs** to each algorithm at crossover probability 0.7*

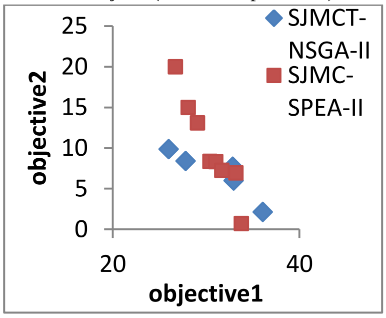
Run	Job	Crossover probability	SJMCT- NSGA-II *Objective1*	*Objective2*	SJMCT-SPEA-II *Objective1*	*Objective2*
4	20	0.7	38.561	1.880	54.238	5.586
			27.460	22.330	23.892	21.201
			32.951	4.852	26.822	11.912
			27.826	18.097	26.394	14.579
			28.081	6.240	31.488	7.755
			29.616	5.606	37.649	6.039
					31.332	9.204
					36.143	6.896
5	20	0.7	24.558	13.880	24.740	14.883
			32.505	2.601	34.890	0.889
			30.314	5.804	27.104	10.639
			27.501	8.173	33.034	5.686
			25.052	9.836	32.915	8.865
					32.400	9.052
					32.298	10.534
6	20	0.7	36.113	2.126	33.788	0.703
			26.035	9.855	26.743	19.964
			32.971	5.980	28.096	14.956
			27.825	8.392	29.110	13.076
			32.871	7.677	30.444	8.328
					31.077	8.325
					31.717	7.229
					33.206	6.929
7	20	0.7	26.620	19.134	25.733	38.543
			32.796	0.236	28.923	0.168
			29.870	8.421	25.860	20.413
			27.746	16.808	26.602	12.800
			28.889	11.814	27.742	6.882
			29.010	10.393		



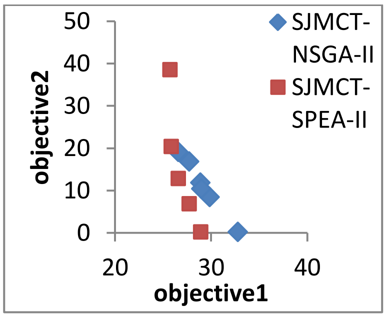
Appendix B.4. Solutions at run 4 for 20 jobs (Crossover prob. 0.7)



Appendix B.5. Solutions at run 5 for 20 jobs (Crossover prob. 0.7)



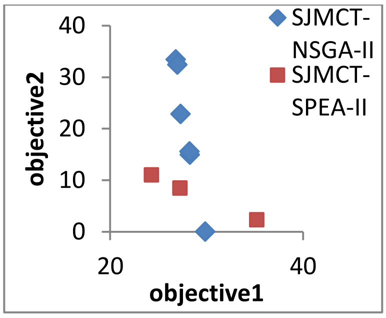
Appendix B.6. Solutions at run 6 for 20 jobs (Crossover prob. 0.7)



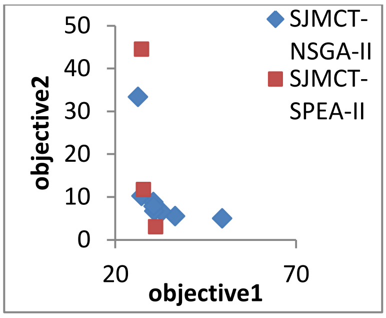
Appendix B.7. Solutions at run 7 for 20 jobs (Crossover prob. 0.7)

APPENDIX B. Table 1 (Continue) *The values of the best non-dominated front for **20 jobs** to each algorithm at crossover probability 0.7*

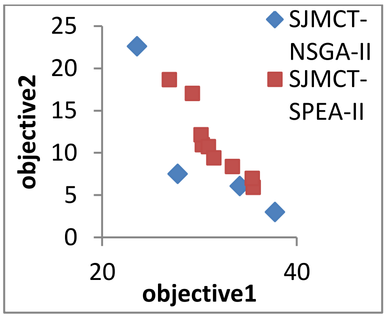
Run	Job	Crossover probability	SJMCT- NSGA-II Objective1	SJMCT- NSGA-II Objective2	SJMCT-SPEA-II Objective1	SJMCT-SPEA-II Objective2
8	20	0.7	26.810	33.407	24.309	11.039
			29.864	0.000	27.249	8.499
			28.289	14.942	35.220	2.341
			27.323	22.863		
			28.255	15.589		
			26.984	32.408		
9	20	0.7	49.599	4.941	27.277	44.478
			26.308	33.355	27.780	11.708
			27.292	10.162	31.207	2.975
			36.636	5.448		
			32.850	6.508		
			30.529	8.759		
			30.846	6.706		
			30.626	7.825		
10	20	0.7	37.783	2.990	26.925	18.643
			23.587	22.591	35.531	5.891
			27.787	7.499	29.294	17.007
			34.152	6.055	30.322	10.937
					30.196	12.107
					35.450	6.979
					30.949	10.711
					31.506	9.408
					33.400	8.355



Appendix B.8. Solutions at run 8 for 20 jobs (Crossover prob. 0.7)



Appendix B.9. Solutions at run 9 for 20 jobs (Crossover prob. 0.7)

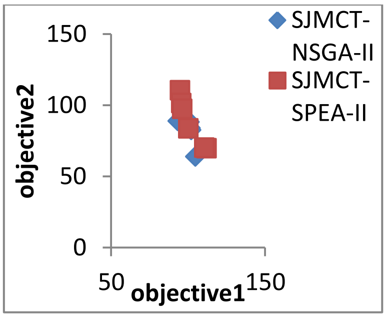


Appendix B.10. Solutions at run 10 for 20 jobs (Crossover prob. 0.7)

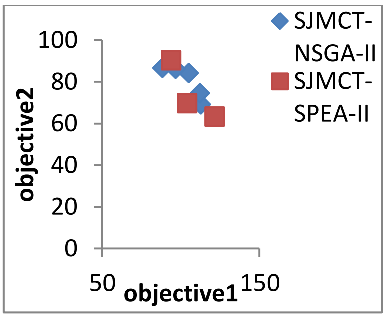
Simulation results for second test problems to each algorithm for 5 machines and **60 jobs**. The values of the best non-dominated front at generation 500 with number of population are 100 and **crossover probability 0.7** as given in APPENDIX B. Table 2.

APPENDIX B. Table 2 *The values of the best non-dominated front for **60 jobs** to each algorithm at crossover probability 0.7*

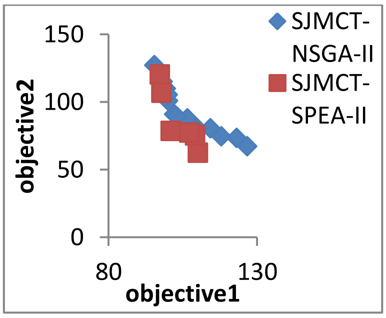
Run	Job	Crossover probability	SJMCT- NSGA-II *Objective1*	SJMCT- NSGA-II *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
1	60	0.7	104.821	63.836	94.736	110.466
			93.275	88.818	95.634	101.489
			102.093	82.724	96.236	97.437
			101.040	88.182	100.119	83.699
			101.583	83.964	111.988	69.860
					110.708	70.256
2	60	0.7	88.232	86.529	93.853	90.203
			112.694	69.091	121.512	63.301
			105.031	84.060	104.070	69.677
			112.155	74.504		
			96.594	86.109		
3	60	0.7	126.743	67.194	110.099	62.376
			95.522	127.240	97.348	120.363
			98.328	115.203	100.938	78.687
			123.127	73.341	97.871	106.802
			117.979	74.563	109.121	75.191
			114.370	80.390	107.278	77.507
			100.138	100.987		
			102.166	90.880		
			109.918	80.521		
			107.388	83.809		
			106.586	87.785		
			99.376	109.938		
			103.177	89.012		
			100.024	105.288		



Appendix B.11. Solutions at run 1 for 60 jobs (Crossover prob. 0.7)



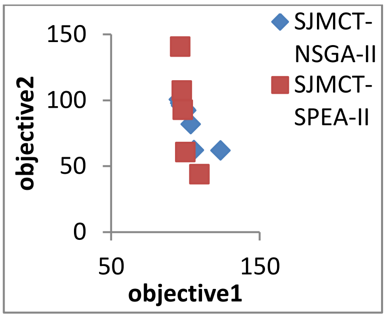
Appendix B.12. Solutions at run 2 for 60 jobs (Crossover prob. 0.7)



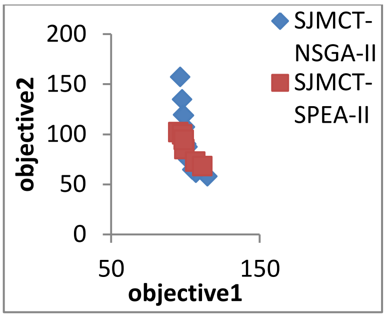
Appendix B.13. Solutions at run 3 for 60 jobs (Crossover prob. 0.7)

APPENDIX B. Table 2 (Continue) *The values of the best non-dominated front for **60 jobs** to each algorithm at crossover probability 0.7*

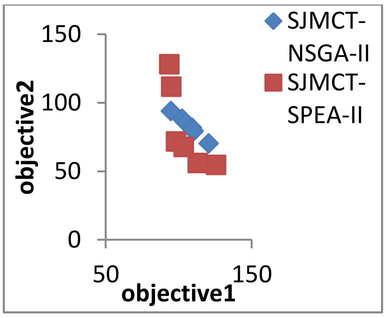
Run	Job	Crossover probability	SJMCT- NSGA-II *Objective1*	SJMCT- NSGA-II *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
4	60	0.7	95.711	100.573	109.535	43.770
			123.767	61.925	100.042	60.636
			105.886	62.106	96.653	140.600
			103.708	81.848	98.387	92.795
			100.808	92.271	97.641	107.435
			97.283	95.851		
			100.124	93.305		
			96.719	97.846		
5	60	0.7	96.585	157.090	95.372	101.970
			114.992	57.673	99.469	84.920
			107.080	61.193	97.835	99.634
			97.920	134.544	98.981	94.320
			104.826	64.605	106.818	72.468
			101.909	74.830	111.545	67.915
			99.853	106.974		
			100.687	90.037		
			98.555	119.295		
			101.667	87.142		
			99.175	118.722		
6	60	0.7	120.788	70.258	125.732	54.494
			94.750	93.850	93.680	127.881
			110.357	79.038	113.180	55.789
			102.415	88.287	98.275	71.530
			104.611	85.555	94.986	111.597
			109.366	81.749	103.471	67.708
			109.831	79.707		



Appendix B.14. Solutions at run 4 for 60 jobs (Crossover prob. 0.7)



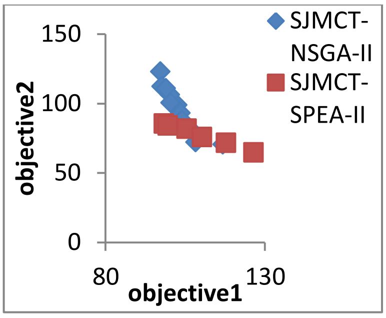
Appendix B.15. Solutions at run 5 for 60 jobs (Crossover prob. 0.7)



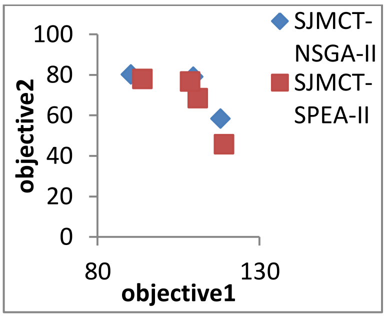
Appendix B.16. Solutions at run 6 for 60 jobs (Crossover prob. 0.7)

APPENDIX B. Table 2 (Continue) *The values of the best non-dominated front for* ***60 jobs*** *to each algorithm at crossover probability 0.7*

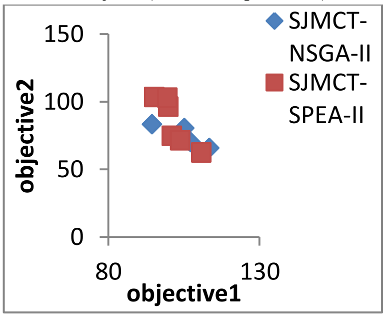
Run	Job	Crossover Probability	SJMCT- NSGA-II *Objective1*	SJMCT- NSGA-II *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
7	60	0.7	116.783	70.433	126.473	64.645
			97.160	122.815	98.318	85.456
			108.297	72.102	117.757	71.607
			104.648	85.178	99.552	84.048
			97.615	112.242	105.423	81.803
			102.500	98.843	110.261	75.910
			107.672	78.981		
			100.181	106.322		
			103.509	93.072		
			100.462	100.422		
			98.910	110.857		
			103.899	88.059		
			107.755	78.559		
8	60	0.7	117.956	58.393	119.014	45.700
			90.333	80.200	93.884	77.963
			109.607	79.081	110.947	68.540
					108.545	76.742
9	60	0.7	94.513	83.359	95.165	103.567
			113.399	65.838	110.813	62.456
			105.140	80.587	101.025	74.916
			107.322	70.116	103.840	71.533
					99.907	96.333
					99.582	102.918
10	60	0.7	114.906	61.809	121.246	62.761
			96.807	139.091	95.657	101.199
			114.228	75.201	98.204	90.054
			99.880	78.033	120.661	75.396
			97.359	128.988	119.514	76.812
			98.828	79.489	111.184	79.298
			98.528	106.423	110.566	80.940
			98.361	110.574	103.144	88.759
					110.255	84.073
					109.743	88.105



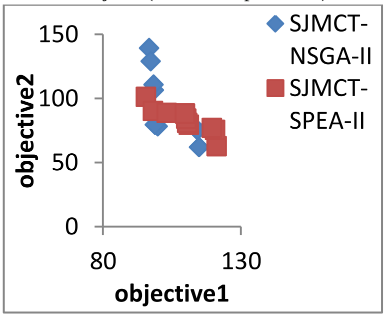
Appendix B.17. Solutions at run 7 for 60 jobs (Crossover prob. 0.7)



Appendix B.18. Solutions at run 8 for 60 jobs (Crossover prob. 0.7)



Appendix B.19. Solutions at run 9 for 60 jobs (Crossover prob. 0.7)

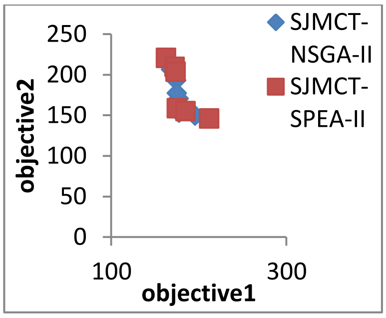


Appendix B.20. Solutions at run 10 for 60 jobs (Crossover prob. 0.7)

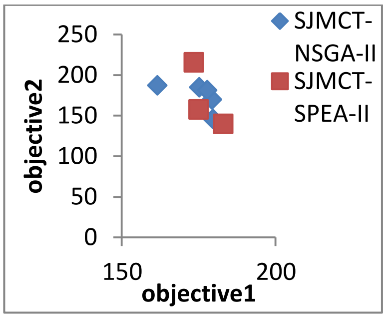
Simulation results for second test problems to each algorithm for 5 machines and **100 jobs**. The values of the best non-dominated front at generation 500 with number of population are 100 and **crossover probability 0.7** as given in APPENDIX B. Table 3.

APPENDIX B. Table 3 *The values of the best non-dominated front for **100 jobs** to each algorithm at crossover probability 0.7*

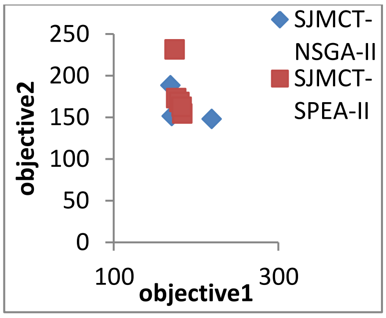
Run	Job	Crossover Probability	SJMCT- NSGA-II *Objective1*	SJMCT- NSGA-II *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
1	100	0.7	195.765	149.480	211.794	146.034
			168.559	206.506	162.763	220.543
			177.587	153.048	175.186	158.695
			174.221	192.986	172.312	209.608
			177.113	170.523	185.070	154.951
			175.205	177.084	173.394	203.838
2	100	0.7	161.546	187.237	183.002	139.518
			179.671	145.442	174.934	157.491
			175.169	184.582	173.422	215.603
			179.369	169.824		
			177.771	181.086		
3	100	0.7	219.205	147.972	175.856	172.820
			168.796	188.493	183.376	154.564
			170.338	151.360	179.779	168.164
					173.768	231.295
					182.464	162.714



Appendix B.21. Solutions at run 1 for 100 jobs (Crossover prob. 0.7)



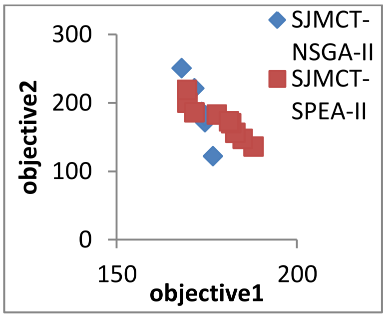
Appendix B.22. Solutions at run 2 for 100 jobs (Crossover prob. 0.7)



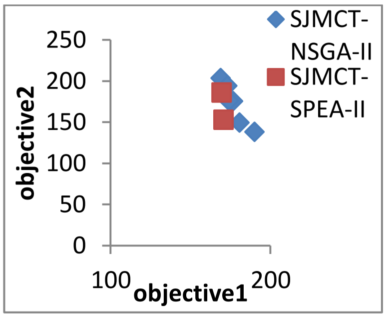
Appendix B.23. Solutions at run 3 for 100 jobs (Crossover prob. 0.7)

APPENDIX B. Table 3 (Continue) *The values of the best non-dominated front for **100 jobs** to each algorithm at crossover probability 0.7*

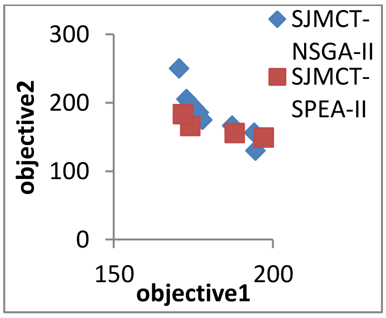
Run	Job	Crossover Probability	SJMCT- NSGA-II *Objective1*	SJMCT- NSGA-II *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
4	100	0.7	168.035	250.868	188.039	136.148
			176.719	121.874	169.642	200.106
			171.584	221.218	169.569	218.936
			174.535	171.715	171.612	186.237
			173.614	186.765	184.999	147.286
			174.114	183.745	182.927	156.032
			174.465	180.301	181.780	170.198
					177.846	182.822
					181.184	173.211
5	100	0.7	168.985	203.541	170.8106	152.9715
			190.198	138.152	169.4504	186.2566
			180.741	149.697		
			176.796	175.736		
			173.119	194.338		
			174.118	175.974		
6	100	0.7	170.483	250.180	174.008	166.189
			194.511	129.883	171.712	183.469
			187.244	166.395	197.149	149.378
			194.115	156.077	187.971	155.872
			177.811	174.764		
			172.877	205.187		
			176.764	185.551		
			175.343	192.837		
			173.716	200.551		
7	100	0.7	180.215	155.091	168.173	125.815
			169.498	227.551		
			175.582	168.273		
			170.997	196.467		



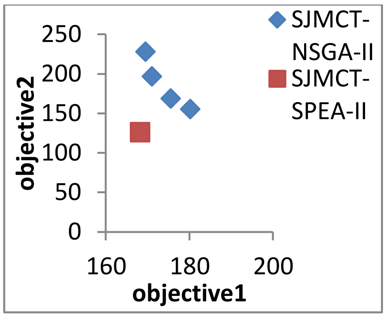
Appendix B.24. Solutions at run 4 for 100 jobs (Crossover prob. 0.7)



Appendix B.25. Solutions at run 5 for 100 jobs (Crossover prob. 0.7)



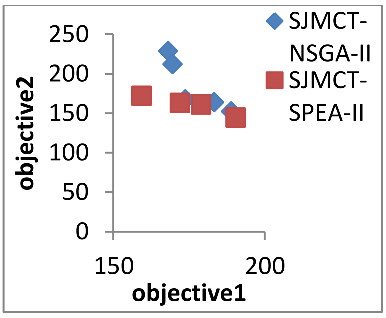
Appendix B.26. Solutions at run 6 for 100 jobs (Crossover prob. 0.7)



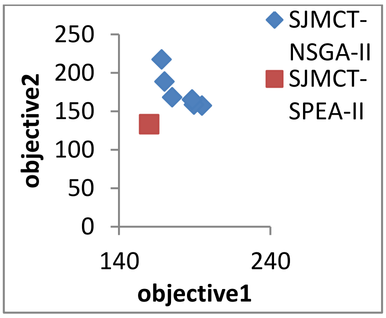
Appendix B.27. Solutions at run 7 for 100 jobs (Crossover prob. 0.7)

APPENDIX B. Table 3 (Continue) *The values of the best non-dominated front for **100 jobs** to each algorithm at crossover probability 0.7*

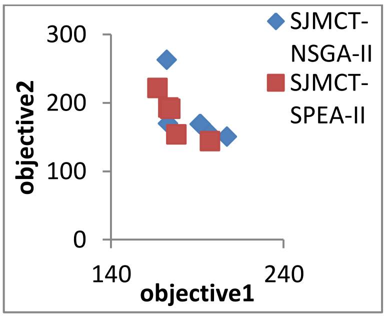
Run	Job	Crossover Probability	SJMCT- NSGA-II *Objective1*	SJMCT- NSGA-II *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
8	100	0.7	168.067	228.949	159.274	171.999
			188.933	152.204	171.977	163.081
			173.849	167.712	190.406	144.374
			169.546	212.066	179.004	160.929
			183.365	164.022		
9	100	0.7	194.866	157.437	159.966	133.284
			168.208	217.341		
			170.058	188.475		
			175.217	168.367		
			188.343	165.171		
			189.615	158.403		
			188.505	164.840		
10	100	0.7	172.373	263.051	197.340	144.029
			207.212	150.433	177.919	153.691
			173.164	169.816	167.037	221.806
			197.062	155.746	173.249	193.417
			191.467	169.086	174.307	191.163
			192.020	168.820		



Appendix B.28. Solutions at run 8 for 100 jobs (Crossover prob. 0.7)



Appendix B.29. Solutions at run 9 for 100 jobs (Crossover prob. 0.7)



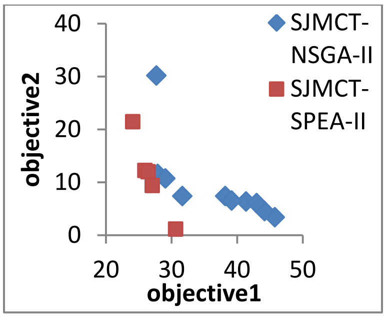
Appendix B.30. Solutions at run 10 for 100 jobs (Crossover prob. 0.7)

APPENDIX C

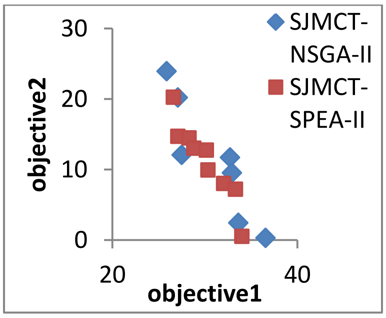
Simulation results for second test problems to each algorithm for 5 machines and **20 jobs**. The values of the best non-dominated front at generation 500 with number of population are 100 and **crossover probability 0.8** as given in APPENDIX C. Table 1.

APPENDIX C. Table 1 *The values of the best non-dominated front for **20 jobs** to each algorithm at crossover probability 0.8*

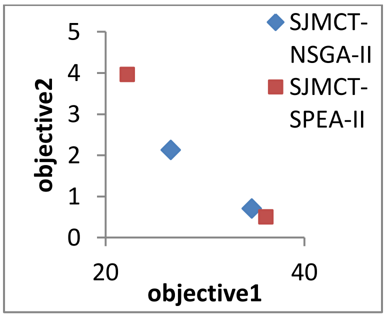
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
1	20	0.8	27.756	30.153	24.148	21.409
			45.806	3.364	25.991	12.191
			27.924	11.564	27.100	9.332
			31.691	7.364	26.480	11.940
			38.241	7.336	30.670	1.102
			29.104	10.719		
			44.235	4.510		
			41.432	6.310		
			43.038	6.021		
			39.202	6.498		
2	20	0.8	25.910	23.951	34.031	0.516
			36.599	0.291	26.616	20.258
			27.522	12.080	27.108	14.707
			33.635	2.444	28.325	14.465
			27.093	20.210	30.348	9.915
			32.751	11.687	28.825	13.012
			32.956	9.528	32.052	8.030
					33.348	7.218
					30.169	12.736
3	20	0.8	26.596	2.127	22.212	3.966
			34.755	0.705	36.142	0.504



Appendix C.1. Solutions at run 1 for 20 jobs (Crossover prob. 0.8)



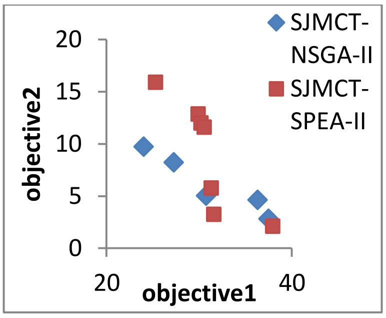
Appendix C.2. Solutions at run 2 for 20 jobs (Crossover prob. 0.8)



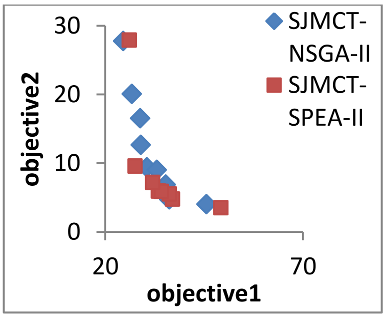
Appendix C.3. Solutions at run 3 for 20 jobs (Crossover prob. 0.8)

APPENDIX C. Table 1 (Continue) *The values of the best non-dominated front for* ***20 jobs*** *to each algorithm at crossover probability 0.8*

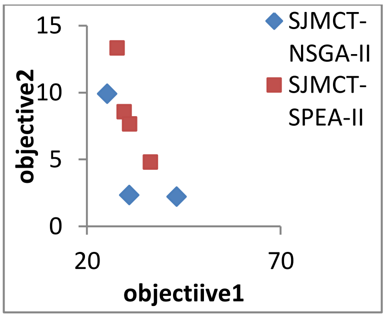
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
4	20	0.8	37.473	2.820	25.305	15.895
			24.031	9.746	31.579	3.252
			30.740	5.042	37.944	2.110
			27.274	8.220	31.299	5.774
			36.310	4.624	29.904	12.865
					30.203	11.994
					30.544	11.610
5	20	0.8	24.553	27.758	49.307	3.528
			45.617	4.014	27.609	9.561
			26.734	20.077	37.012	4.758
			36.204	4.722	26.128	27.909
			28.883	16.537	36.219	5.504
			29.008	12.661	33.458	5.937
			30.555	9.393	34.253	5.901
			33.099	8.995	32.024	7.189
			35.333	6.891		
6	20	0.8	43.204	2.207	29.602	8.560
			25.272	9.910	27.812	13.319
			30.972	2.341	31.042	7.659
					36.369	4.804
7	20	0.8	25.792	12.175	26.244	14.419
			31.934	0.549	28.773	6.242
			30.011	6.292	30.269	3.551
			26.888	9.320	25.981	19.801
			28.877	8.962		
			29.400	6.916		



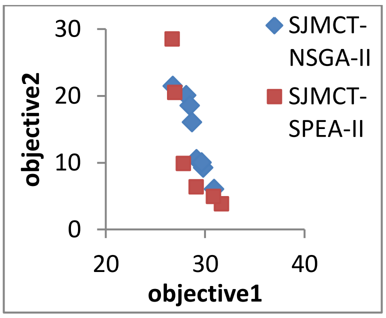
Appendix C.4. Solutions at run 4 for 20 jobs (Crossover prob. 0.8)



Appendix C.5. Solutions at run 5 for 20 jobs (Crossover prob. 0.8)



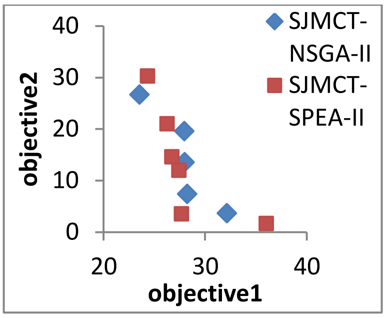
Appendix C.6. Solutions at run 6 for 20 Jobs (Crossover prob. 0.8)



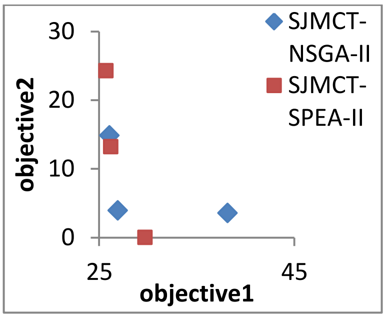
Appendix C.7. Solutions at run 7 for 20 jobs (Crossover prob. 0.8)

APPENDIX C. Table 1 (Continue) *The values of the best non-dominated front for **20 jobs** to each algorithm at crossover probability 0.8*

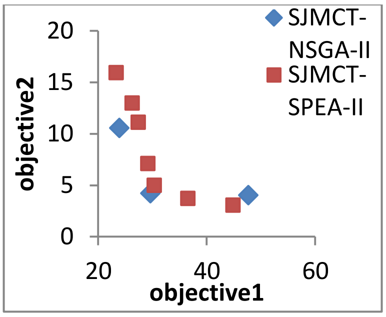
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
8	20	0.8	32.140	3.632	24.321	30.224
			23.547	26.644	27.671	3.505
			27.970	19.526	26.241	20.964
			28.236	7.340	36.013	1.624
			27.990	13.498	27.409	11.932
					26.726	14.570
9	20	0.8	38.148	3.560	29.692	0.000
			26.050	14.827	26.167	13.229
			26.897	3.938	25.690	24.263
10	20	0.8	47.687	4.004	44.876	3.044
			23.893	10.535	23.319	15.931
			29.616	4.191	26.285	12.983
					27.412	11.086
					29.139	7.102
					30.325	4.987
					36.531	3.695



Appendix C.8. Solutions at run 8 for 20 jobs (Crossover prob. 0.8)



Appendix C.9. Solutions at run 9 for 20 jobs (Crossover prob. 0.8)

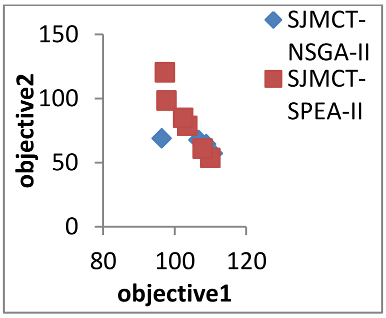


Appendix C.10. Solutions at run 10 for 20 jobs (Crossover prob. 0.8)

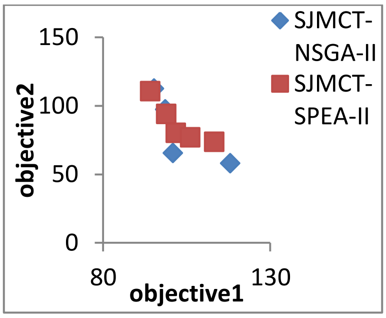
Simulation results for second test problems to each algorithm for 5 machines and **60 jobs**. The values of the best non-dominated front at generation 500 with number of population are 100 and **crossover probability 0.8** as given in APPENDIX C. Table 2.

APPENDIX C. Table 2 *The values of the best non-dominated front for* ***60 jobs*** *to each algorithm at crossover probability 0.8*

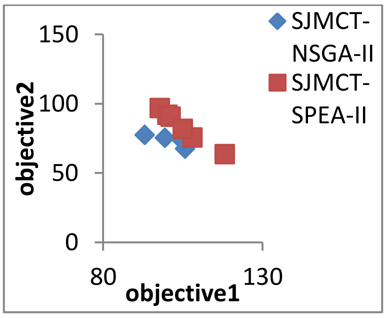
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
1	60	0.8	110.691	57.063	110.019	53.667
			96.414	68.981	107.844	60.935
			106.689	67.534	97.786	98.376
			108.864	64.371	97.304	120.473
					103.522	78.553
					102.406	85.020
2	60	0.8	118.016	58.020	94.131	110.563
			95.292	112.492	101.802	80.294
			100.920	65.539	98.886	93.964
			98.651	97.249	113.262	73.713
					106.106	76.860
3	60	0.8	105.817	67.444	118.229	63.539
			93.143	77.321	97.784	96.745
			99.484	75.261	100.238	91.977
			105.109	73.068	108.106	75.668
					101.277	90.487
					105.060	81.841



Appendix C.11. Solutions at run 1 for 60 jobs (Crossover prob. 0.8)



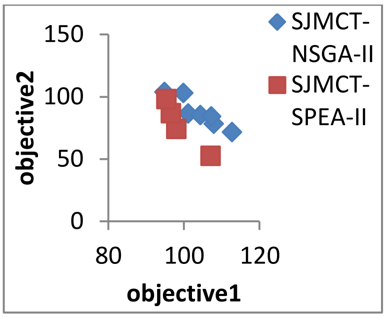
Appendix C.12. Solutions at run 2 for 60 jobs (Crossover prob. 0.8)



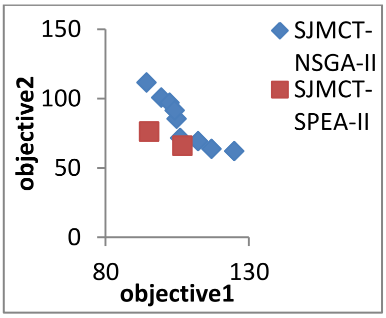
Appendix C.13. Solutions at run 3 for 60 jobs (Crossover prob. 0.8)

APPENDIX C. Table 2 (Continue) *The values of the best non-dominated front for **60 jobs** to each algorithm at crossover probability 0.8*

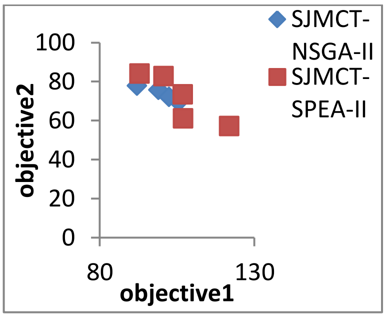
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
4	60	0.8	112.710	71.439	107.054	52.310
			94.917	103.558	97.977	74.004
			99.912	102.954	96.555	86.674
			101.148	86.432	95.371	97.978
			107.942	78.188		
			104.382	85.138		
			107.220	83.997		
5	60	0.8	94.377	111.498	95.207	76.120
			124.948	62.048	106.831	66.101
			106.088	71.659		
			117.021	63.655		
			99.447	100.689		
			112.337	69.268		
			104.858	85.394		
			102.366	96.855		
			104.142	91.383		
6	60	0.8	92.095	77.846	92.873	84.013
			105.490	69.593	121.881	57.242
			99.031	75.675	106.993	61.183
			102.427	72.133	100.664	82.780
					106.909	73.468
7	60	0.8	94.431	101.871	126.818	57.971
			120.545	76.415	97.782	67.947
			107.271	76.997	109.599	67.742
			102.177	93.348	96.947	108.692
			103.358	83.007		
			117.692	76.980		
			104.125	81.518		
			98.149	100.060		
			102.074	99.599		
			102.084	95.992		



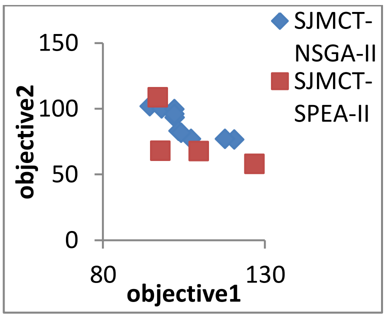
Appendix C.14. Solutions at run 4 for 60 jobs (Crossover prob. 0.8)



Appendix C.15. Solutions at run 5 for 60 Jobs (Crossover prob. 0.8)



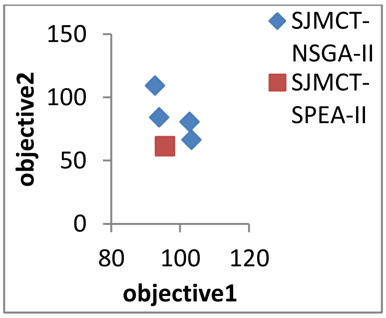
Appendix C.16. Solutions at run 6 for 60 Jobs (Crossover prob. 0.8)



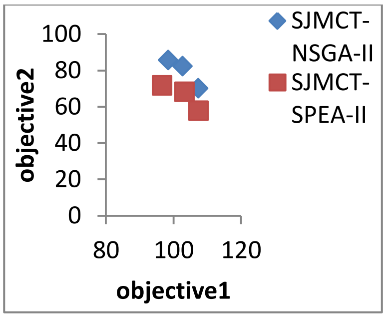
Appendix C.17. Solutions at run 7 for 60 Jobs (Crossover prob. 0.8)

APPENDIX C. Table 2 (Continue) *The values of the best non-dominated front for **60 jobs** to each algorithm at crossover probability 0.8*

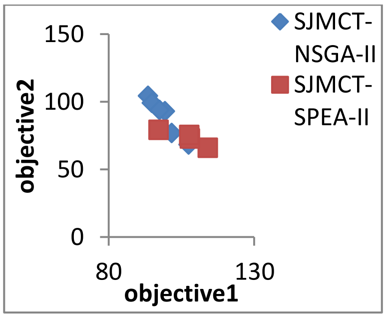
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
8	60	0.8	103.377	66.503	95.593	61.327
			92.764	109.257		
			93.986	84.206		
			102.774	80.638		
9	60	0.8	98.510	85.615	96.719	71.760
			107.322	70.116	107.457	57.846
			102.727	82.378	103.301	68.160
10	60	0.8	107.561	68.425	97.238	79.379
			93.571	104.359	114.118	66.122
			101.662	76.839	108.004	72.850
			99.412	93.131	107.784	75.447
			94.873	99.110		
			97.583	93.989		



Appendix C.18. Solutions at run 8 for 60 jobs (Crossover prob. 0.8)



Appendix C.19. Solutions at run 9 for 60 jobs (Crossover prob. 0.8)

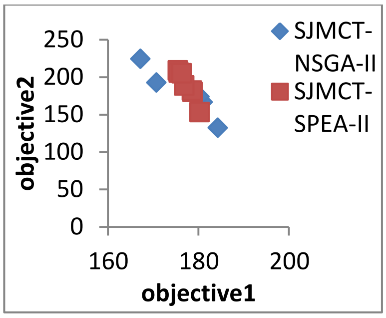


Appendix C.20. Solutions at run 10 for 60 jobs (Crossover prob. 0.8)

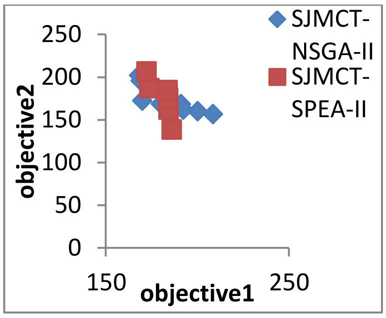
Simulation results for second test problems to each algorithm for 5 machines and **100 jobs**. The values of the best non-dominated front at generation 500 with number of population are 100 and **crossover probability 0.8** as given in APPENDIX C. Table 3.

APPENDIX C. Table 3 *The values of the best non-dominated front for **100 jobs** to each algorithm at crossover probability 0.8*

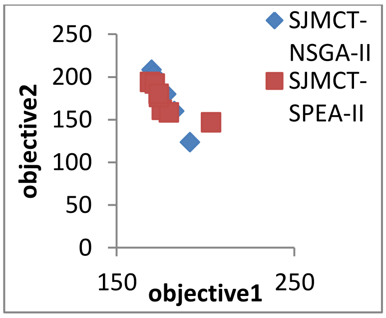
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
1	100	0.8	167.181	223.981	180.245	152.963
			184.241	132.031	175.445	208.761
			170.738	192.467	178.668	179.920
			177.539	181.610	175.671	205.708
			180.946	166.259	178.566	181.728
			180.234	173.797	176.878	188.460
					176.319	204.967
2	100	0.8	208.500	156.305	186.035	138.621
			168.239	201.794	173.733	187.047
			170.015	172.043	172.179	206.551
			169.031	195.586	184.189	161.852
			179.995	168.434	183.717	181.496
			200.213	159.730	184.145	175.399
			191.119	168.081	183.667	184.551
			192.386	161.834		
3	100	0.8	169.683	208.324	203.295	146.663
			191.280	123.328	168.859	193.933
			182.548	159.932	175.436	161.187
			177.989	179.645	179.456	158.189
			174.104	184.732	171.532	192.017
			170.914	202.469	173.896	176.520
					173.690	180.186



Appendix C.21. Solutions at run 1 for 100 jobs (Crossover prob. 0.8)



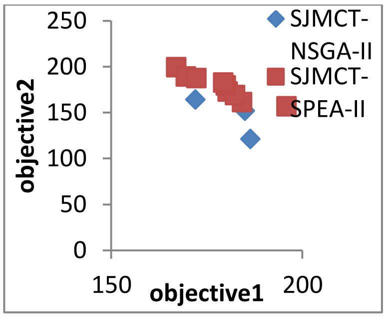
Appendix C.22. Solutions at run 2 for 100 jobs (Crossover prob. 0.8)



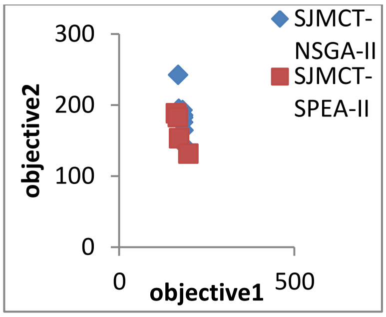
Appendix C.23. Solutions at run 3 for 100 jobs (Crossover prob. 0.8)

APPENDIX C. Table 3 (Continue) *The values of the best non-dominated front for **100 jobs** to each algorithm at crossover probability 0.8*

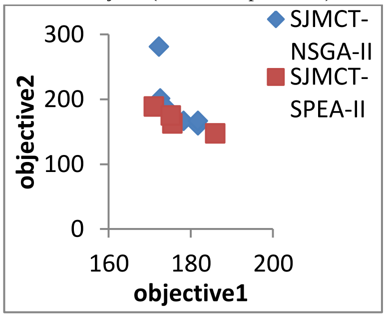
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	*Objective2*	SJMCT-SPEA-II *Objective1*	*Objective2*
4	100	0.8	186.402	121.310	167.042	199.407
			172.032	164.179	169.504	189.299
			185.014	151.948	195.911	156.803
					172.268	187.619
					184.266	161.548
					180.458	172.799
					182.462	169.136
					179.970	179.497
					179.316	182.870
5	100	0.8	168.231	242.052	162.387	188.046
			185.719	142.625	170.811	152.972
			170.388	194.631	167.727	182.930
			180.994	192.664	197.508	131.297
			183.524	164.242		
			183.111	185.356		
			183.243	175.480		
			183.208	182.291		
6	100	0.8	181.743	159.591	185.988	147.358
			172.277	281.347	175.539	162.857
			172.607	201.433	171.060	188.972
			178.290	166.750	175.190	175.216
			174.206	180.669		
			181.728	166.439		
			173.904	188.020		
7	100	0.8	199.247	157.870	171.263	209.356
			170.060	257.022	175.684	187.488
			172.845	216.095	173.356	200.789
			188.565	162.844	187.748	164.369
			181.785	169.867	190.736	160.308
			174.708	183.809	180.270	182.152
			195.685	161.193	186.786	170.837
			174.475	209.376	182.898	180.932
			177.473	172.368	185.438	176.055
			176.677	174.668		



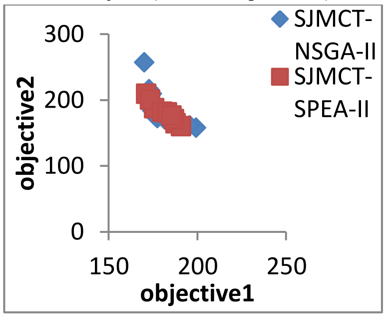
Appendix C.24. Solutions at run 4 for 100 jobs (Crossover prob. 0.8)



Appendix C.25. Solutions at run 5 for 100 jobs (Crossover prob. 0.8)



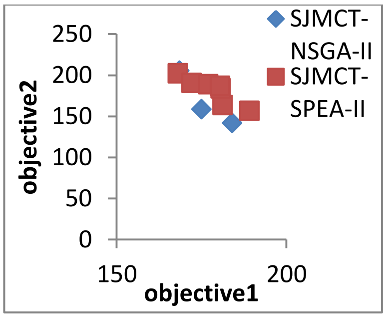
Appendix C.26. Solutions at run 6 for 100 jobs (Crossover prob. 0.8)



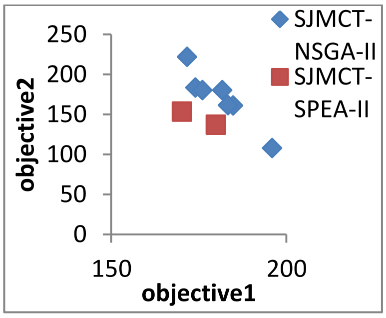
Appendix C.27. Solutions at run 7 for 100 jobs (Crossover prob. 0.8)

APPENDIX C. Table 3 (Continue) *The values of the best non-dominated front for **100 jobs** to each algorithm at crossover probability 0.8*

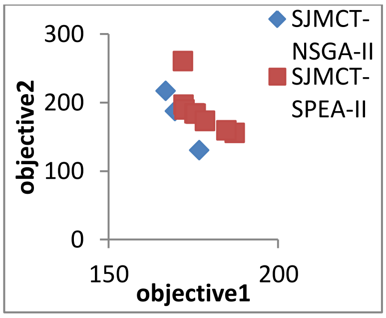
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
8	100	0.8	184.082	142.091	168.038	202.511
			168.553	205.494	172.107	190.587
			175.008	158.749	181.256	163.603
					189.133	156.594
					177.029	189.236
					180.650	184.174
					180.526	187.098
9	100	0.8	171.669	221.829	170.232	153.558
			195.960	107.876	179.857	136.991
			184.813	160.956		
			174.087	183.654		
			181.708	180.284		
			176.094	180.315		
			183.363	161.251		
10	100	0.8	166.852	217.187	171.941	260.636
			176.699	130.593	172.104	197.130
			169.577	187.489	172.366	189.946
					187.219	155.847
					184.768	159.561
					175.336	184.513
					178.478	172.996
					175.719	184.029
					178.439	173.668



Appendix C.28. Solutions at run 8 for 100 jobs (Crossover prob. 0.8)



Appendix C.29. Solutions at run 9 for 100 jobs (Crossover prob. 0.8)



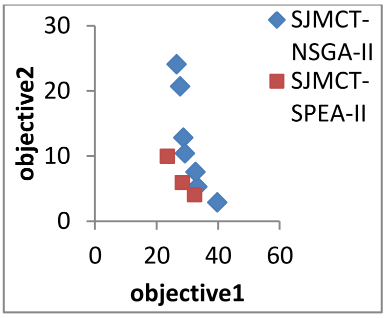
Appendix C.30. Solutions at run 10 for 100 jobs (Crossover prob. 0.8)

APPENDIX D

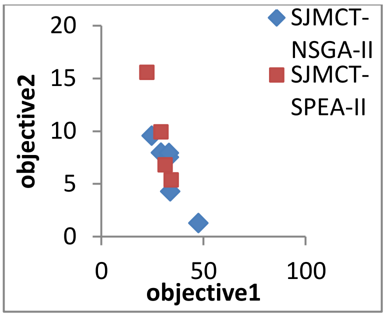
Simulation results for second test problems to each algorithm for 5 machines and **20 jobs**. The values of the best non-dominated front at generation 500 with number of population are 100 and **crossover probability 0.9** as given in APPENDIX D. Table 1.

APPENDIX D. Table 1 *The values of the best non-dominated front for **20 jobs** to each algorithm at crossover probability 0.9*

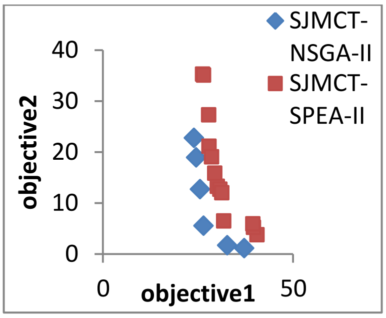
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
1	20	0.9	26.570	24.081	23.485	9.939
			39.718	2.854	28.446	5.894
			33.289	5.249	32.413	4.019
			27.654	20.671		
			28.705	12.818		
			32.686	7.557		
			29.209	10.389		
2	20	0.9	24.772	9.536	22.344	15.560
			47.650	1.268	29.188	9.912
			33.767	4.269	31.331	6.781
			29.281	7.928	34.343	5.371
			33.217	7.524		
			33.134	7.904		
3	20	0.9	23.909	22.752	26.157	35.216
			37.137	1.098	26.479	35.075
			26.485	5.490	40.520	3.771
			32.694	1.669	39.545	5.182
			25.483	12.652	27.786	27.253
			24.485	18.908	39.427	5.858
					27.867	21.083
					31.789	6.422
					30.062	13.221
					28.523	19.016
					30.778	12.669
					29.365	15.828
					31.254	11.997



Appendix D.1. Solutions at run 1 for 20 jobs (Crossover prob. 0.9)



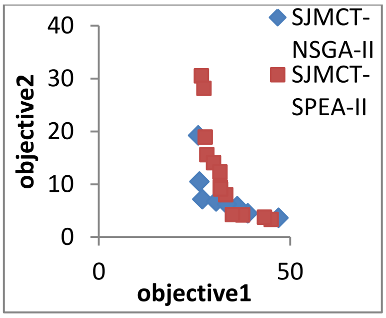
Appendix D.2. Solutions at run 2 for 20 jobs (Crossover prob. 0.9)



Appendix D.3. Solutions at run 3 for 20 jobs (Crossover prob. 0.9)

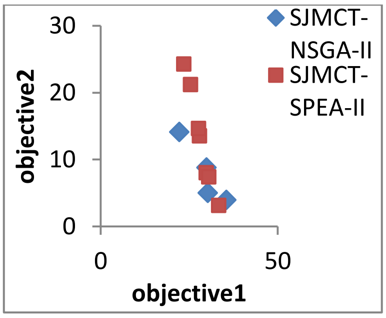
APPENDIX D. Table 1 (Continue) *The values of the best non-dominated front for **20 jobs** to each algorithm at crossover probability 0.9*

Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
4	20	0.9	26.059	19.242	45.139	3.363
			47.080	3.623	43.374	3.770
			26.362	10.519	28.304	15.565
			39.018	4.507	26.910	30.484
			27.124	7.129	37.723	4.176
			30.719	6.710	27.860	18.948
			36.239	5.831	35.038	4.222
			33.568	6.162	27.557	28.116
					30.100	14.075
					31.810	9.471
					33.215	8.002
					31.937	9.073
					31.734	12.306



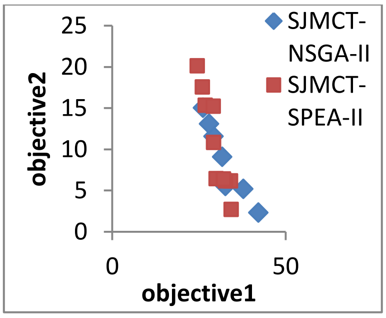
Appendix D.4. Solutions at run 4 for 20 jobs (Crossover prob. 0.9)

Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
5	20	0.9	22.282	14.129	23.544	24.280
			35.610	3.946	25.452	21.184
			29.981	8.813	29.849	8.006
			30.332	4.987	27.945	13.517
					33.444	3.126
					27.717	14.650
					30.592	7.425



Appendix D.5. Solutions at run 5 for 20 jobs (Crossover prob. 0.9)

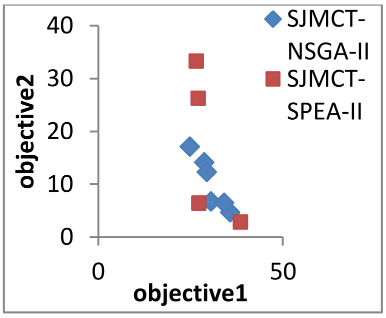
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
6	20	0.9	42.229	2.322	24.585	20.119
			26.202	15.028	26.089	17.564
			37.863	5.197	26.931	15.336
			31.774	9.059	34.377	2.707
			32.715	5.533	30.073	6.436
			29.270	11.556	34.216	6.160
			27.997	13.061	32.708	6.165
					29.268	10.813
					29.205	15.204
					32.125	6.392



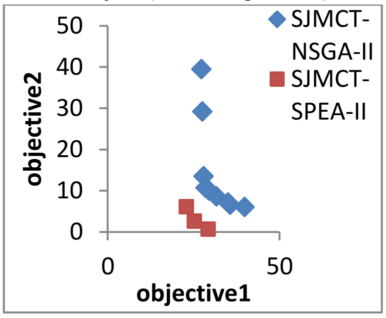
Appendix D.6. Solutions at run 6 for 20 jobs (Crossover prob. 0.9)

APPENDIX D. Table 1 (Continue) *The values of the best non-dominated front for **20 jobs** to each algorithm at crossover probability 0.9*

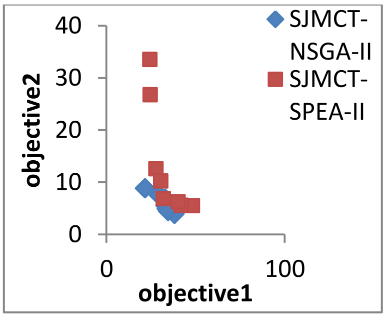
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
7	20	0.9	35.803	4.660	26.590	33.258
			24.860	17.092	38.600	2.857
			30.601	6.696	27.285	6.436
			28.806	14.102	27.150	26.220
			29.485	12.286		
			34.102	6.461		
8	20	0.9	39.910	6.028	22.923	6.057
			27.308	39.478	25.266	2.576
			27.563	29.159	29.303	0.609
			27.990	13.498		
			31.701	8.582		
			35.699	6.535		
			35.070	7.123		
			29.589	9.976		
			28.494	10.683		
9	20	0.9	38.323	3.872	48.362	5.475
			21.732	8.815	42.058	5.573
			29.700	7.471	24.398	33.479
			33.042	6.763	24.692	26.776
			34.601	4.475	40.239	6.253
			33.197	5.502	27.951	12.557
			34.000	4.901	32.012	6.850
					30.587	10.232
10	20	0.9	40.352	3.684	26.176	31.609
			25.667	28.681	33.541	3.544
			27.639	8.317	37.959	2.669
			26.345	26.871	33.393	5.021
			32.802	5.069	29.677	9.382
			30.977	5.399	28.089	13.317
					27.446	19.573
					30.995	8.900
					27.687	17.877



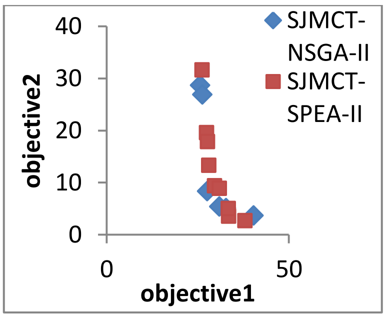
Appendix D.7. Solutions at run 7 for 20 jobs (Crossover prob. 0.9)



Appendix D.8. Solutions at run 8 for 20 jobs (Crossover prob. 0.9)



Appendix D.9. Solutions at run 9 for 20 jobs (Crossover prob. 0.9)

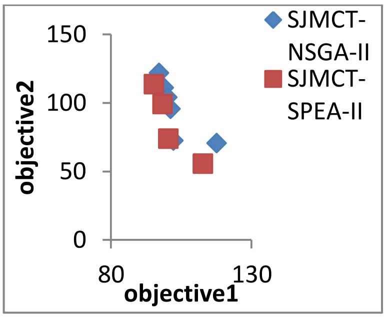


Appendix D.10. Solutions at run 10 for 20 jobs (Crossover prob. 0.9)

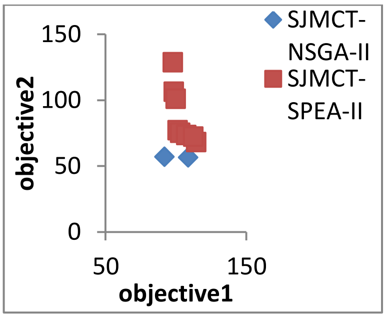
Simulation results for second test problems to each algorithm for 5 machines and **60 jobs**. The values of the best non-dominated front at generation 500 with number of population are 100 and **crossover probability 0.9** as given in APPENDIX D. Table 2.

APPENDIX D. Table 2 *The values of the best non-dominated front for **60 jobs** to each algorithm at crossover probability 0.9*

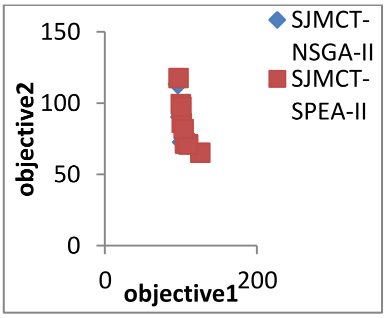
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
1	60	0.9	97.116	121.903	112.674	55.712
			117.669	70.647	100.471	73.837
			102.218	72.547	95.291	113.669
			101.264	95.724	98.359	99.067
			98.870	111.135		
			100.098	103.991		
2	60	0.9	108.787	56.472	101.230	77.220
			92.017	56.968	102.978	75.268
					97.926	128.804
					114.463	68.088
					107.451	73.593
					98.486	106.439
					112.355	72.243
					99.916	100.958
3	60	0.9	95.972	113.360	125.316	65.126
			109.267	70.717	105.268	71.752
			99.287	90.067	109.489	71.034
			102.246	72.364	96.951	117.546
			101.183	83.354	101.093	86.099
			101.512	72.535	103.716	81.783
					99.965	99.425
					100.785	97.526



Appendix D.11. Solutions at run 1 for 60 jobs (Crossover prob. 0.9)



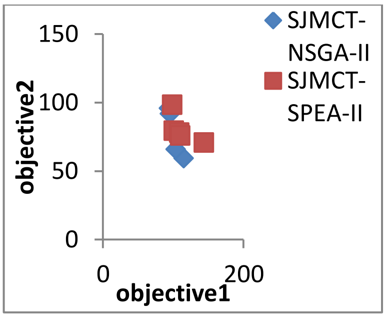
Appendix D.12. Solutions at run 2 for 60 jobs (Crossover prob. 0.9)



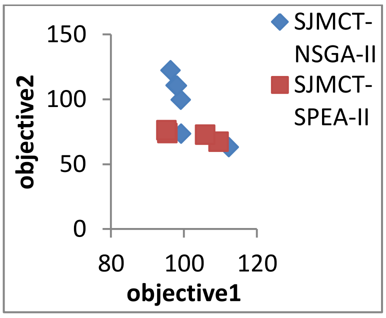
Appendix D.13. Solutions at run 3 for 60 jobs (Crossover prob. 0.9)

APPENDIX D. Table 2 (Continue) *The values of the best non-dominated front for **60 jobs** to each algorithm at crossover probability 0.9*

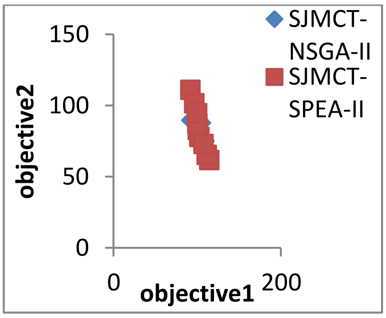
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
4	60	0.9	93.911	95.833	143.796	70.767
			114.805	59.378	100.910	79.636
			103.604	65.993	98.353	98.380
			95.419	91.977	108.753	78.098
					109.729	76.038
5	60	0.9	112.299	63.041	95.506	73.847
			96.345	122.419	95.207	76.120
			99.247	73.423	109.543	67.243
			99.152	99.641	105.807	72.612
			97.784	110.822		
			98.156	110.355		
6	60	0.9	112.906	63.551	91.914	110.917
			93.049	89.641	114.309	61.681
			108.761	74.976	111.617	65.201
			104.648	87.766	96.630	101.504
			106.483	77.776	100.601	82.861
			98.873	88.101	102.510	77.903
					107.506	72.886
					100.353	87.158
					100.067	94.462
7	60	0.9	114.381	55.745	96.475	97.806
			96.304	119.803	96.442	113.969
			100.410	72.050	108.414	70.467
			112.377	67.623	101.478	91.358
			97.683	91.868	104.240	88.332
					108.382	81.154



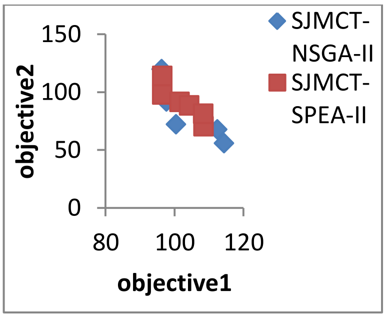
Appendix D.14. Solutions at run 4 for 60 jobs (Crossover prob. 0.9)



Appendix D.15. Solutions at run 5 for 60 jobs (Crossover prob. 0.9)



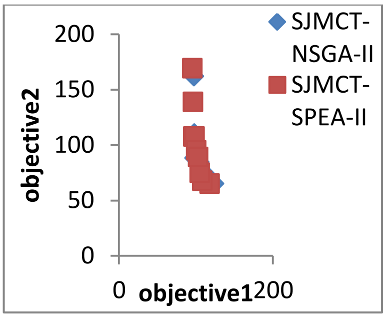
Appendix D.16. Solutions at run 6 for 60 jobs (Crossover prob. 0.9)



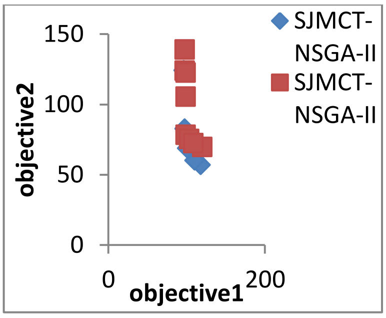
Appendix D.17. Solutions at run 7 for 60 jobs (Crossover prob. 0.9)

APPENDIX D. Table 2 (Continue) *The values of the best non-dominated front for* ***60 jobs*** *to each algorithm at crossover probability 0.9*

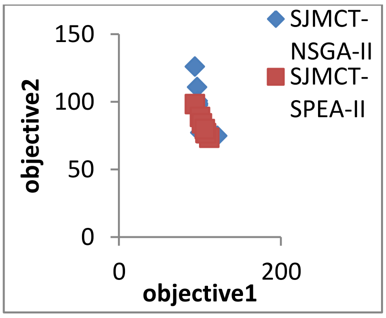
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
8	60	0.9	97.330	161.927	95.576	169.399
			123.597	65.153	117.725	65.147
			98.000	110.167	96.257	138.969
			113.816	72.465	96.803	107.744
			106.484	72.814	108.387	67.545
			98.261	88.331	98.337	107.591
			104.433	84.527	104.832	75.018
			101.458	86.178	100.214	95.240
			102.816	85.286	102.798	89.124
9	60	0.9	96.495	124.232	97.547	138.893
			117.927	56.965	99.002	78.352
			97.526	82.584	120.244	69.739
			110.500	60.050	103.464	75.460
			103.053	68.128	98.818	105.451
			101.134	68.853	97.845	123.927
					108.736	72.484
					98.720	122.473
10	60	0.9	121.585	74.875	93.931	98.276
			94.058	126.058	100.198	88.855
			106.398	75.796	102.593	84.115
			100.520	77.234	111.585	73.364
			96.797	110.927	106.923	77.016
			97.909	97.459	107.551	76.741
			97.268	100.710	106.095	79.775
			97.781	98.680		



Appendix D.18. Solutions at run 8 for 60 jobs (Crossover prob. 0.9)



Appendix D.19. Solutions at run 9 for 60 jobs (Crossover prob. 0.9)

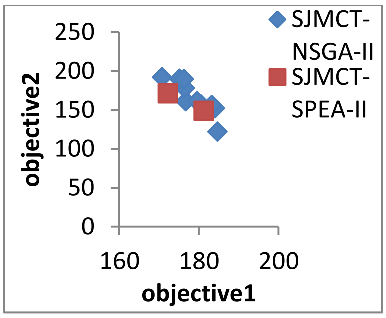


Appendix D.20. Solutions at run 10 for 60 jobs (Crossover prob. 0.9)

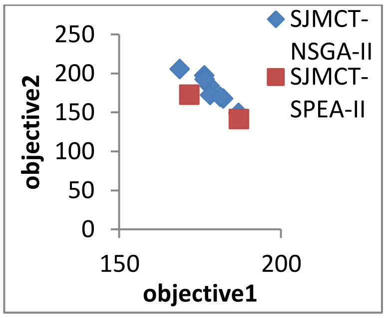
Simulation results for second test problems to each algorithm for 5 machines and **100 jobs**. The values of the best non-dominated front at generation 500 with number of population are 100 and **crossover probability 0.9** as given in APPENDIX D. Table 3.

APPENDIX D. Table 3 *The values of the best non-dominated front for **100 jobs** to each algorithm at crossover probability 0.9*

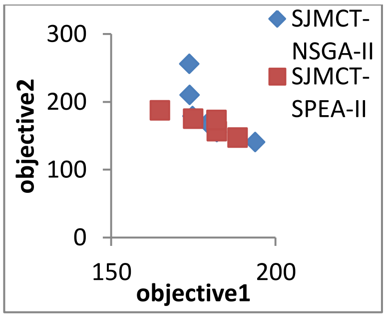
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
1	100	0.9	170.778	191.703	172.250	171.500
			184.749	121.926	181.295	148.558
			184.089	152.129		
			176.746	161.185		
			176.479	177.755		
			175.133	189.760		
			180.493	157.327		
			183.259	156.085		
			179.558	160.923		
			176.240	189.364		
2	100	0.9	186.949	149.363	187.052	141.244
			168.717	205.322	171.709	172.688
			182.186	167.718		
			176.327	197.191		
			177.980	183.813		
			178.099	172.151		
			176.394	192.292		
			181.272	169.741		
3	100	0.9	173.816	256.091	164.814	187.278
			193.809	140.604	188.426	147.073
			182.163	155.198	182.144	156.542
			173.865	210.239	174.941	175.282
			174.768	178.691	182.067	173.360
			179.545	169.216		
			179.919	165.856		



Appendix D.21. Solutions at run 1 for 100 jobs (Crossover prob. 0.9)



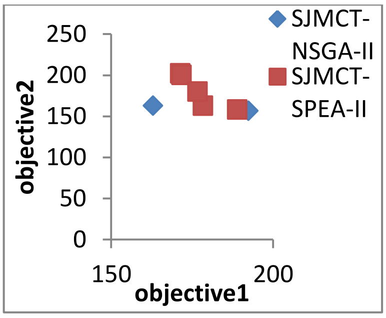
Appendix D.22. Solutions at run 2 for 100 jobs (Crossover prob. 0.9)



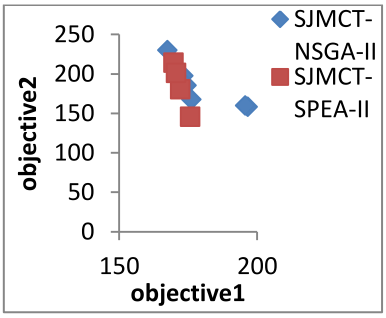
Appendix D.23. Solutions at run 3 for 100 jobs (Crossover prob. 0.9)

APPENDIX D. Table 3 (Continue) *The values of the best non-dominated front for **100 jobs** to each algorithm at crossover probability 0.9*

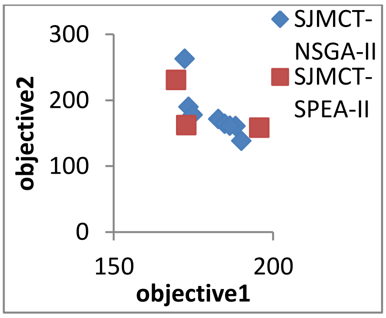
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
4	100	0.9	162.999	162.860	178.392	162.858
			192.422	156.552	171.385	202.420
					171.788	200.296
					188.976	158.258
					176.739	180.191
5	100	0.9	167.534	229.915	175.853	145.479
			196.659	158.116	169.788	214.064
			176.314	167.387	172.214	180.610
			173.415	197.666	170.710	201.173
			195.816	159.771		
			174.507	185.458		
6	100	0.9	190.064	138.109	172.861	162.060
			172.305	262.826	169.645	230.888
			173.534	190.039	195.784	158.112
			174.831	177.505		
			182.896	171.166		
			188.287	160.863		
			184.883	164.274		
			186.534	161.156		
7	100	0.9	187.590	139.767	173.608	151.896
			169.551	204.863	171.660	235.849
			179.816	161.046	172.748	209.060
			177.735	175.145		
			177.572	189.676		
			172.604	191.394		
			171.741	194.180		



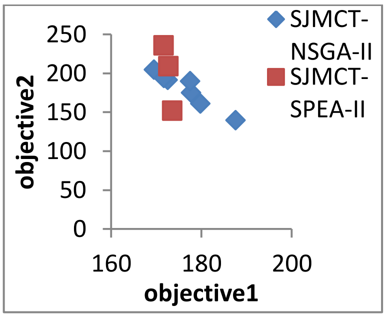
Appendix D.24. Solutions at run 4 for 100 jobs (Crossover prob. 0.9)



Appendix D.25. Solutions at run 5 for 100 jobs (Crossover prob. 0.9)



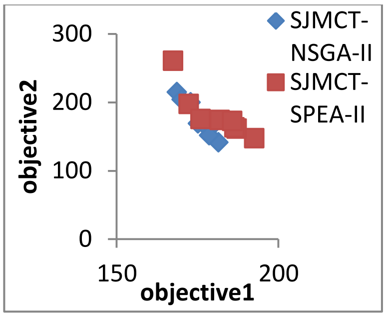
Appendix D.26. Solutions at run 6 for 100 jobs (Crossover prob. 0.9)



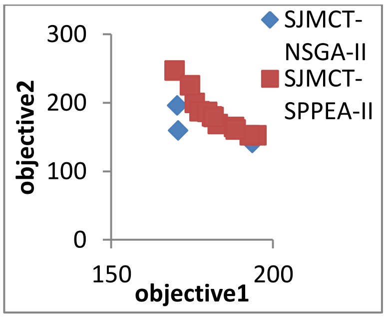
Appendix D.27. Solutions at run 7 for 100 jobs (Crossover prob. 0.9)

APPENDIX D. Table 3 (Continue) *The values of the best non-dominated front for **100 jobs** to each algorithm at crossover probability 0.9*

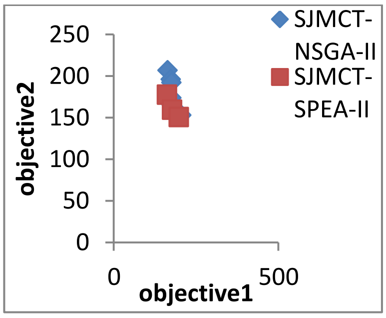
Run	Job	Crossover Probability	SJMCT- NSGAII *Objective1*	SJMCT- NSGAII *Objective2*	SJMCT-SPEA-II *Objective1*	SJMCT-SPEA-II *Objective2*
8	100	0.9	181.558	141.151	167.505	260.778
			168.677	214.798	172.279	197.271
			175.151	168.805	176.008	175.657
			172.892	199.674	192.596	147.523
			170.185	203.970	187.183	161.203
			178.593	160.354	186.418	163.436
			178.602	151.409	181.932	174.137
					185.761	172.573
9	100	0.9	193.647	141.060	169.655	246.799
			170.367	195.866	194.959	152.578
			170.667	159.490	192.856	152.646
					182.935	169.068
					175.831	199.930
					188.043	164.578
					177.315	187.530
					174.377	225.557
					188.548	161.115
					181.165	179.885
					179.716	186.558
					181.646	179.299
10	100	0.9	205.411	152.837	161.982	177.607
			163.364	206.852	176.736	159.277
			176.116	165.169	197.734	150.774
			173.190	195.988		
			175.460	173.614		
			175.171	192.188		



Appendix D.28. Solutions at run 8 for 100 jobs (Crossover prob. 0.9)



Appendix D.29. Solutions at run 9 for 100 jobs (Crossover prob. 0.9)



Appendix D.30. Solutions at run 10 for 100 jobs (Crossover prob. 0.9)

APPENDIX E

GAMS PROGRAMMING FOR BALINS TEST PROBLEM BY USING SJMCT ALGORITHM

```
SETS
     I   machine / 1*4 /
     J job / 1*9 /
```

TABLE p(I,J) processing time to assigning job J to machine I

	1	2	3	4	5	6	7	8	9
1	18	14	24	30	16	20	22	26	14
2	9	7	12	15	8	10	11	13	7
3	4.5	3.5	6	7.5	4	5	5.5	6.5	3.5
4	3.6	2.8	4.8	6	3.2	4	4.4	5.2	2.8 ;

```
VARIABLES
Z,Z1,Z2,X15,X25,X35,X45,X16,X26,X36,X46,X17,X27,X37,X47,X18,X28,X38,X4
8,X19,X29,X39,X49;
 EQUATIONS
OBJ,kisit1,kisit2,kisit3,kisit4,kisit5,kisit6,kisit7,kisit8,kisit9,kis
it10,kisit11,kisit12,kisit13,kisit14,kisit15,kisit16,kisit17,kisit18,k
isit19,kisit20;
parameter  X(I,J),C(I,J),C11,C22,C33,C44
,C15,C25,C35,C45,C16,C26,C36,C46,C17,C27,C37,C47,C18,C28,C38,C48,C19,C
29,C39,C49;
X('1','1') = 1;
X('2','2') = 1;
X('3','3') =1;
X('4','4') =1;
C11=p('1','1')*X('1','1');
C22=p('2','2')*X('2','2');
C33=p('3','3')*X('3','3');
C44=p('4','4')*X('4','4');
*************The  first iteration  J=5:
     if (C11 <=C22 and C11 <=C33 and C11 <=C44 ,
display C11;
X('1','5')=1;
else
X('1','5')=0;
 if (C22 <= C11  and C22 <=C33 and C22 <=C44,
display C22;
X('2','5')=1;
else
X('2','5')=0;
if (C33 <= C11 and C33 <=C22 and C33 <=C44,
display C33;
X('3','5')=1;
else
X('3','5')=0;
if (C44 <= C11 and C44 <=C22 and C44 <=C33 ,
display C44;
 X('4','5')=1;
else
```

```
X('4','5')=0;
);
);
);
);
kisit1.. X('1','5') =E= X15;
kisit2.. X('2','5') =E= X25;
kisit3.. X('3','5') =E= X35;
kisit4.. X('4','5') =E= X45;
C15=p('1','1')*X('1','1')+p('1','5')*X('1','5');
C25=p('2','2')*X('2','2')+p('2','5')*X('2','5');
C35=p('3','3')*X('3','3')+p('3','5')*X('3','5');
C45=p('4','4')*X('4','4')+p('4','5')*X('4','5');
********************************************
*************The  second iteration  J=6:
 if (C15 <=C25 and C15 <=C35 and  C15 <=C45,
display C15;
X('1','6')=1;
else
X('1','6')=0;
     if (C25 <=C15 and C25 <=C35 and C25 <=C45,
display C25;
X('2','6')=1;
else
X('2','6')=0;
     if (C35 <=C15 and C35 <=C25 and C35 <=C45,
display C35;
X('3','6')=1;
else
X('3','6')=0;
     if (C45 <=C15 and C45 <=C25 and C45 <=C35,
display C45;
X('4','6')=1;
else
X('4','6')=0;
);
);
);
);
C16=p('1','1')*X('1','1')+p('1','5')*X('1','5')+p('1','6')*X('1','6');
C26=p('2','2')*X('2','2')+p('2','5')*X('2','5')+p('2','6')*X('2','6');
C36=p('3','3')*X('3','3')+p('3','5')*X('3','5')+p('3','6')*X('3','6');
C46=p('4','4')*X('4','4')+p('4','5')*X('4','5')+p('4','6')*X('4','6');
kisit5.. X('1','6') =E= X16;
kisit6.. X('2','6') =E= X26;
kisit7.. X('3','6') =E= X36;
kisit8.. X('4','6') =E= X46;
*************The third iteration  J=7:
 if (C16 <=C26 and C16 <=C36 and  C16 <=C46,
display C16;
X('1','7')=1;
else
```

```
X('1','7')=0;
     if (C26 <=C16 and C26 <=C36 and C26 <=C46,
display C26;
X('2','7')=1;
else
X('2','7')=0;
     if (C36 <=C16 and C36 <=C26 and C36 <=C46,
display C36;

X('3','7')=1;
else
X('3','7')=0;
     if (C46 <=C16 and C46 <=C26 and C46 <=C36,
display C46;
X('4','7')=1;
else
X('4','7')=0;
);
);
);
);
C17=p('1','1')*X('1','1')+p('1','5')*X('1','5')+p('1','6')*X('1','6')+
p('1','7')*X('1','7');
C27=p('2','2')*X('2','2')+p('2','5')*X('2','5')+p('2','6')*X('2','6')+
p('2','7')*X('2','7');
C37=p('3','3')*X('3','3')+p('3','5')*X('3','5')+p('3','6')*X('3','6')+
p('3','7')*X('3','7');
C47=p('4','4')*X('4','4')+p('4','5')*X('4','5')+p('4','6')*X('4','6')+
p('4','7')*X('4','7');
kisit9..  X('1','7') =E= X17;
kisit10.. X('2','7') =E= X27;
kisit11.. X('3','7') =E= X37;
kisit12.. X('4','7') =E= X47;
*************The forth iteration  J=8:
 if (C17 <=C27 and C17 <=C37 and  C17 <=C47,
display C17;
X('1','8')=1;
else
X('1','8')=0;
     if (C27 <=C17 and C27 <=C37 and C27 <=C47,
display C27;
X('2','8')=1;
else
X('2','8')=0;
     if (C37 <=C17 and C37 <=C27 and C37 <=C47,
display C37;
X('3','8')=1;
else
X('3','8')=0;
     if (C47 <=C17 and C47 <=C27 and C47 <=C37,
display C47;
```

```
X('4','8')=1;
else
X('4','8')=0;
);
);
);
);
C18=p('1','1')*X('1','1')+p('1','5')*X('1','5')+p('1','6')*X('1','6')+
p('1','7')*X('1','7')+p('1','8')*X('1','8');
C28=p('2','2')*X('2','2')+p('2','5')*X('2','5')+p('2','6')*X('2','6')+
p('2','7')*X('2','7')+p('2','8')*X('2','8');
C38=p('3','3')*X('3','3')+p('3','5')*X('3','5')+p('3','6')*X('3','6')+
p('3','7')*X('3','7')+p('3','8')*X('3','8');
C48=p('4','4')*X('4','4')+p('4','5')*X('4','5')+p('4','6')*X('4','6')+
p('4','7')*X('4','7')+p('4','8')*X('4','8');
kisit13..  X('1','8') =E= X18;
kisit14.. X('2','8') =E= X28;
kisit15.. X('3','8') =E= X38;
kisit16.. X('4','8') =E= X48;
*************The fifth iteration  J=9:
 if (C18 <=C28 and C18 <=C38 and  C18 <=C48,
display C18;
X('1','9')=1;
else
X('1','9')=0;
     if (C28 <=C18 and C28 <=C38 and C28 <=C48,
display C28;
X('2','9')=1;
else
X('2','9')=0;
     if (C38 <=C18 and C38 <=C28 and C38 <=C48,
display C38;
X('3','9')=1;
else
X('3','9')=0;
     if (C48 <=C18 and C48 <=C28 and C48 <=C38,
display C48;
X('4','9')=1;
else
X('4','9')=0;
);
);
);
);
C19=p('1','1')*X('1','1')+p('1','5')*X('1','5')+p('1','6')*X('1','6')+
p('1','7')*X('1','7')+p('1','8')*X('1','8')+p('1','9')*X('1','9');
C29=p('2','2')*X('2','2')+p('2','5')*X('2','5')+p('2','6')*X('2','6')+
p('2','7')*X('2','7')+p('2','8')*X('2','8')+p('2','9')*X('2','9');
C39=p('3','3')*X('3','3')+p('3','5')*X('3','5')+p('3','6')*X('3','6')+
p('3','7')*X('3','7')+p('3','8')*X('3','8')+p('3','9')*X('3','9');
C49=p('4','4')*X('4','4')+p('4','5')*X('4','5')+p('4','6')*X('4','6')+
p('4','7')*X('4','7')+p('4','8')*X('4','8')+p('4','9')*X('4','9');
```

```
kisit17.. X('1','9') =E= X19;
kisit18.. X('2','9') =E= X29;
kisit19.. X('3','9') =E= X39;
kisit20.. X('4','9') =E= X49;
 if (C19> C29 and C19> C39 and C19> C49,
display C19;
 else
C19=0;
 );
   if (C29> C19 and C29> C39 and C29> C49,
display C29;
 else
C29=0;
 );
 if (C39> C19 and C39> C29 and C39> C49,
display C39;
 else
C39=0;
 );
 if (C49> C19 and C49> C29 and C49> C39,
display C49;
 else
C49=0;
);
**********************************************************
OBJ..    Z=E=C19+C29+C39+C49;
 MODEL SCHEDUALING / ALL /;
 SOLVE SCHEDUALING USING MIP MINIMIZING Z ;
```

MATLAB PROGRAMMING (FIRST TEST PROBLEM) TO SOLVE SJMCT-NSGA-II AND SJMCT-SPEA-II ALGORITHM WITH SELECTED PAREMTERS 60 JOBS AND GENERATION 40

COMPUTE THE FITNESS FUNCTION Z=MP60(x)

```
m=5
n=60
p=unifrnd(1,20,[m n]);
t=unifrnd(1,20,[m n]);
for i= 1:m
s(i)=p(i,i)
d1=s
end
 for i= 1:m
 r(i)=t(i,i)
 r1=r
end
for i=1:m
if s(i)==min(s)
 s(i)=s(i)+p(i,m+1)
 a6=t(i,m+1);
 break
end
end
for j=1:m
```

```
d2(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d2)
 s(i)=min(d2)+p(i,m+2)
 a7=t(i,m+2);
 break
end
end
for j=1:m
d3(j)=[s(1,j)]
end
for i=1:m
if s(i)==min(d3)
s(i)=min(d3)+p(i,m+3)
a8=t(i,m+3);
break
end
end
for j=1:m
d4(j)=[s(1,j)]
end
for i=1:m
if s(i)==min(d4)
s(i)=min(d4)+p(i,m+4)
a9=t(i,m+4);
break
end
end
for j=1:m
d5(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d5)
s(i)=min(d5)+p(i,m+5)
a10=t(i,m+5);
 break
end
end
for j=1:m
d6(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d6)
s(i)=min(d6)+p(i,m+6)
a11=t(i,m+6);
 break
end
end
for j=1:m
d7(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d7)
s(i)=min(d7)+p(i,m+7)
a12=t(i,m+7);
 break
end
end
for j=1:m
```

```
d8(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d8)
s(i)=min(d8)+p(i,m+8)
a13=t(i,m+8);
 break
end
end
for j=1:m
d9(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d9)
s(i)=min(d9)+p(i,m+9)
a14=t(i,m+9);
 break
end
end
for j=1:m
d10(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d10)
s(i)=min(d10)+p(i,m+10)
a15=t(i,m+10);
 break
end
end
for j=1:m
d11(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d11)
s(i)=min(d11)+p(i,m+11)
a16=t(i,m+11);
 break
end
end
for j=1:m
d12(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d12)
s(i)=min(d12)+p(i,m+12)
a17=t(i,m+12);
 break
end
end
for j=1:m
d13(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d13)
s(i)=min(d13)+p(i,m+13)
a18=t(i,m+13);
 break
end
end
for j=1:m
```

```
d14(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d14)
s(i)=min(d14)+p(i,m+14)
a19=t(i,m+14);
 break
end
end
for j=1:m
d15(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d15)
s(i)=min(d15)+p(i,m+15)
a20=t(i,m+15);
 break
end
end
for j=1:m
d16(j)=[s(1,j)]
end
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 40 job
for i= 1:m
if s(i)==min(d16)
 s(i)=min(d16)+p(i,m+16)
 a21=t(i,m+16);
 break
end
end
for j=1:m
d17(j)=[s(1,j)]
end
for i=1:m
if s(i)==min(d17)
s(i)=min(d17)+p(i,m+17)
a22=t(i,m+17);
break
end
end
for j=1:m
d18(j)=[s(1,j)]
end
for i=1:m
if s(i)==min(d18)
s(i)=min(d18)+p(i,m+18);
a23=t(i,m+18);
break
end
end
for j=1:m
d19(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d19)
s(i)=min(d19)+p(i,m+19)
a24=t(i,m+19);
 break
end
end
```

```
for j=1:m
d20(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d20)
s(i)=min(d20)+p(i,m+20)
a25=t(i,m+20);
 break
end
end
for j=1:m
d21(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d21)
s(i)=min(d21)+p(i,m+21)
a26=t(i,m+21);
 break
end
end
for j=1:m
d22(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d22)
s(i)=min(d22)+p(i,m+22)
a27=t(i,m+22);
 break
end
end
for j=1:m
d23(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d23)
s(i)=min(d23)+p(i,m+23)
a28=t(i,m+23);
 break
end
end
for j=1:m
d24(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d24)
s(i)=min(d24)+p(i,m+24)
a29=t(i,m+24);
 break
end
end
for j=1:m
d25(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d25)
s(i)=min(d25)+p(i,m+25)
a30=t(i,m+25);
 break
end
end
```

```
for j=1:m
d26(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d26)
s(i)=min(d26)+p(i,m+26)
a31=t(i,m+26);
 break
end
end
for j=1:m
d27(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d27)
s(i)=min(d27)+p(i,m+27)
a32=t(i,m+27);
 break
end
end
for j=1:m
d28(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d28)
s(i)=min(d28)+p(i,m+28)
a33=t(i,m+28);
 break
end
end
for j=1:m
d29(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d29)
s(i)=min(d29)+p(i,m+29)
a34=t(i,m+29);
 break
end
end
for j=1:m
d30(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d30)
s(i)=min(d30)+p(i,m+30)
a35=t(i,m+30);
 break
end
end
for j=1:m
d31(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d31)
s(i)=min(d31)+p(i,m+31)
a36=t(i,m+31);
 break
end
end
```

```
for j=1:m
d32(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d32)
s(i)=min(d32)+p(i,m+32)
a37=t(i,m+32);
 break
end
end
for j=1:m
d33(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d33)
s(i)=min(d33)+p(i,m+33)
a38=t(i,m+33);
 break
end
end
for j=1:m
d34(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d34)
s(i)=min(d34)+p(i,m+34)
a39=t(i,m+34);
 break
end
end
for j=1:m
d35(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d35)
s(i)=min(d35)+p(i,m+35)
a40=t(i,m+35);
 break
end
end
for j=1:m
d36(j)=[s(1,j)]
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%60 job
for i= 1:m
if s(i)==min(d36)
 s(i)=min(d36)+p(i,m+36)
 a41=t(i,m+36);
 break
end
end
for j=1:m
d37(j)=[s(1,j)]
end
for i=1:m
if s(i)==min(d37)
s(i)=min(d37)+p(i,m+37)
a42=t(i,m+37);
break
end
```

```
end
for j=1:m
d38(j)=[s(1,j)]
end
for i=1:m
if s(i)==min(d38)
s(i)=min(d38)+p(i,m+38)
a43=t(i,m+38);
break
end
end
for j=1:m
d39(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d39)
s(i)=min(d39)+p(i,m+39)
a44=t(i,m+39);
 break
end
end
for j=1:m
d40(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d40)
s(i)=min(d40)+p(i,m+40)
a45=t(i,m+40);
 break
end
end
for j=1:m
d41(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d41)
s(i)=min(d41)+p(i,m+41)
a46=t(i,m+41);
 break
end
end
for j=1:m
d42(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d42)
s(i)=min(d42)+p(i,m+42)
a47=t(i,m+42);
 break
end
end
for j=1:m
d43(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d43)
s(i)=min(d43)+p(i,m+43)
a48=t(i,m+43);
 break
end
```

```
end
for j=1:m
d44(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d44)
s(i)=min(d44)+p(i,m+44)
a49=t(i,m+44);
 break
end
end
for j=1:m
d45(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d45)
s(i)=min(d45)+p(i,m+45)
a50=t(i,m+45);
 break
end
end
for j=1:m
d46(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d46)
s(i)=min(d46)+p(i,m+46)
a51=t(i,m+46);
 break
end
end
 for j=1:m
d47(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d47)
s(i)=min(d47)+p(i,m+47)
a52=t(i,m+47);
 break
end
end
for j=1:m
d48(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d48)
s(i)=min(d48)+p(i,m+48)
a53=t(i,m+48);
 break
end
end
for j=1:m
d49(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d49)
s(i)=min(d49)+p(i,m+49)
a54=t(i,m+49);
 break
end
```

```
end
for j=1:m
d50(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d50)
s(i)=min(d50)+p(i,m+50)
a55=t(i,m+50);
 break
end
end
for j=1:m
d51(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d51)
s(i)=min(d51)+p(i,m+51)
a56=t(i,m+51);
 break
end
end
for j=1:m
d52(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d52)
s(i)=min(d52)+p(i,m+52)
a57=t(i,m+52);
 break
end
end
for j=1:m
d53(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d53)
s(i)=min(d53)+p(i,m+53)
a58=t(i,m+53);
 break
end
end
for j=1:m
d54(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d54)
s(i)=min(d54)+p(i,m+54)
a59=t(i,m+54);
 break
end
end
for j=1:m
d55(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d55)
s(i)=min(d55)+p(i,m+55)
a60=t(i,m+55);
 break
end
```

```
end
for j=1:m
d56(j)=[s(1,j)]
end

%%*****************************************

J6=d2-d1
J7=d3-d2
J8=d4-d3
J9=d5-d4
J10=d6-d5
J11=d7-d6
J12=d8-d7
J13=d9-d8
J14=d10-d9
J15=d11-d10
J16=d12-d11
J17=d13-d12
J18=d14-d13
J19=d15-d14
J20=d16-d15
J21=d17-d16
J22=d18-d17
J23=d19-d18
J24=d20-d19
J25=d21-d20
J26=d22-d21
J27=d23-d22
J28=d24-d23
J29=d25-d24
J30=d26-d25
J31=d27-d26
J32=d28-d27
J33=d29-d28
J34=d30-d29
J35=d31-d30
J36=d32-d31
J37=d33-d32
J38=d34-d33
J39=d35-d34
J40=d36-d35
J41=d37-d36
J42=d38-d37
J43=d39-d38
J44=d40-d39
J45=d41-d40
J46=d42-d41
J47=d43-d42;
J48=d44-d43
J49=d45-d44
J50=d46-d45
J51=d47-d46
J52=d48-d47
J53=d49-d48
J54=d50-d49
J55=d51-d50
J56=d52-d51
J57=d53-d52
J58=d54-d53
```

```
J59=d55-d54
J60=d56-d55
TAR1=d1-r1
TAR6=max(J6)-a6
TAR7=max(J7)-a7
TAR8=max(J8)-a8
TAR9=max(J9)-a9
TAR10=max(J10)-a10
TAR11=max(J11)-a11
TAR12=max(J12)-a12
TAR13=max(J13)-a13
TAR14=max(J14)-a14
TAR15=max(J15)-a15
TAR16=max(J16)-a16
TAR17=max(J17)-a17
TAR18=max(J18)-a18
TAR19=max(J19)-a19
TAR20=max(J20)-a20
TAR21=max(J21)-a21
TAR22=max(J22)-a22
TAR23=max(J23)-a23
TAR24=max(J24)-a24
TAR25=max(J25)-a25
TAR26=max(J26)-a26
TAR27=max(J27)-a27
TAR28=max(J28)-a28
TAR29=max(J29)-a29
TAR30=max(J30)-a30
TAR31=max(J31)-a31
TAR32=max(J32)-a32
TAR33=max(J33)-a33
TAR34=max(J34)-a34
TAR35=max(J35)-a35
TAR36=max(J36)-a36
TAR37=max(J37)-a37
TAR38=max(J38)-a38
TAR39=max(J39)-a39
TAR40=max(J40)-a40
TAR41=max(J41)-a41
TAR42=max(J42)-a42
TAR43=max(J43)-a43
TAR44=max(J44)-a44
TAR45=max(J45)-a45
TAR46=max(J46)-a46
TAR47=max(J47)-a47
TAR48=max(J48)-a48
TAR49=max(J49)-a49
TAR50=max(J50)-a50
TAR51=max(J51)-a51
TAR52=max(J52)-a52
TAR53=max(J53)-a53
TAR54=max(J54)-a54
TAR55=max(J55)-a55
TAR56=max(J56)-a56
TAR57=max(J57)-a57
TAR58=max(J58)-a58
TAR59=max(J59)-a59
TAR60=max(J60)-a60
T=[TAR1,TAR6,TAR7,TAR8,TAR9,TAR10,TAR11,TAR12,TAR13,TAR14,TAR15,TAR16,
TAR17,TAR18,TAR19,TAR20,TAR21,TAR22,TAR23,TAR24,TAR25,TAR26,TAR27,TAR2
```

```
8,TAR29,TAR30,TAR31,TAR32,TAR33,TAR34,TAR35,TAR36,TAR37,TAR38,TAR39,TA
R40,TAR41,TAR42,TAR43,TAR44,TAR45,TAR46,TAR47,TAR48,TAR49,TAR50,TAR51,
TAR52,TAR53,TAR54,TAR55,TAR56,TAR57,TAR58,TAR59,TAR60]
for j=1:n
if T(j) >0
DD(j)=T(j);
else
DD(j)=0;
end
CMAX=max(s)
TARD=sum(DD)
%%
optjobs=[d1;J6;J7;J8;J9;J10;J11;J12;J13;J14;J15;J16;J17;J18;J19;J20;J2
1;J22;J23;J24;J25;J26;J27;J28;J29;J30;J31;J32;J33;J34;J35;J36;J37;J38;
J39;J40;J41;J42;J43;J44;J45;J46;J47;J48;J49;J50;J51;J52;J53;J54;J55;J5
6;J57;J58;J59;J60]';
%figure(1);
%title 'parallel machine';
%barh(optjobs ,'stack');
%xlabel('JOBS')
%ylabel('MACHINE')
%TARD=sum(DD)
%CMAX=max(s)
z1=CMAX;
z2=TARD;
z=[z1 z2]';
end
end
```

USING THE FITNESS FUNCTION Z=MP60(x) WITH CROSSOVER PROBABILITY 0.6 AND THE FOLLOWING ASSUMPTIONS TO SOLVE SJMCT-NSGA-II ALGORITHM

```
clc;
clear;
close all;
%% Problem Definition
CostFunction=@(x)MP60(x);
nVar=[5 60];              % Number of Decision Variables
VarSize=[nVar  1];    % Decision Variables Matrix Size
VarMin=-15;            % Decision Variables Lower Bound
VarMax=15;            % Decision Variables Upper Bound
% Number of Objective Functions
nObj=numel(CostFunction(unifrnd(VarMin,VarMax,VarSize)));
%% NSGA-II Parameters
MaxIt=40;       % Maximum Number of Iterations
nPop=100;        % Population Size
Crossover=0.6;                            % Crossover Percentage
nCrossover=2*round(pCrossover*nPop/2); %Number of Parnets (Offsprings)
pMutation=0.4;                             % Mutation Percentage
nMutation=round(pMutation*nPop);          % Number of Mutants
mu=0.02;                     % Mutation Rate
sigma=0.1*(VarMax-VarMin);  % Mutation Step Size
%% Initialization
empty_individual.Position=[];
empty_individual.Cost=[];
empty_individual.Rank=[];
empty_individual.DominationSet=[];
empty_individual.DominatedCount=[];
```

```
empty_individual.CrowdingDistance=[];
pop=repmat(empty_individual,nPop,1);
for i=1:nPop
    pop(i).Position=unifrnd(VarMin,VarMax,VarSize);
    pop(i).Cost=CostFunction(pop(i).Position);
end
% Non-Dominated Sorting
[pop, F]=NonDominatedSorting(pop);
% Calculate Crowding Distance
pop=CalcCrowdingDistance(pop,F);
% Sort Population
[pop, F]=SortPopulation(pop);
%% NSGA-II Main Loop
for it=1:MaxIt
    % Crossover
    popc=repmat(empty_individual,nCrossover/2,2);
    for k=1:nCrossover/2
        i1=randi([1 nPop]);
        p1=pop(i1);
        i2=randi([1 nPop]);
        p2=pop(i2);
        [popc(k,1).Position,
popc(k,2).Position]=Crossover(p1.Position,p2.Position);
        popc(k,1).Cost=CostFunction(popc(k,1).Position);
        popc(k,2).Cost=CostFunction(popc(k,2).Position);
        end
    popc=popc(:);
    % Mutation
    popm=repmat(empty_individual,nMutation,1);
    for k=1:nMutation
        i=randi([1 nPop]);
        p=pop(i);
        popm(k).Position=Mutate(p.Position,mu,sigma);
        popm(k).Cost=CostFunction(popm(k).Position);
    end
    % Merge
    pop=[pop
         popc
         popm]; %#ok
    % Non-Dominated Sorting
    [pop, F]=NonDominatedSorting(pop);
    % Calculate Crowding Distance
    pop=CalcCrowdingDistance(pop,F);
    % Sort Population
    pop=SortPopulation(pop);
    % Truncate
    pop=pop(1:nPop);
    % Non-Dominated Sorting
    [pop, F]=NonDominatedSorting(pop);
    % Calculate Crowding Distance
    pop=CalcCrowdingDistance(pop,F);
    % Sort Population
    [pop, F]=SortPopulation(pop);
    % Store F1
    F1=pop(F{1});
     % Show Iteration Information
    disp(['Iteration ' num2str(it) ': Number of F1 Members = '
num2str(numel(F1))]);
    % Plot F1 Costs
    figure(1);
```

```
    PlotCosts(F1);
    pause(0.3);
end
%% Results
CF1 = [F1.Cost];
for j=1:size(CF1,1)
    disp(['Objective #' num2str(j) ':']);
    disp(['      Min = ' num2str(min(CF1(j,:)))]);
    disp(['      Max = ' num2str(max(CF1(j,:)))]);
    disp(['    Range = ' num2str(max(CF1(j,:))-min(CF1(j,:)))]);
    disp(['    St.D. = ' num2str(std(CF1(j,:)))]);
    disp(['     Mean = ' num2str(mean(CF1(j,:)))]);
    disp(' ');
end
```

USING THE FITNESS FUNCTION Z=MP60(x) WITH CROSSOVER PROBABILITY 0.6 AND THE FOLLOWING ASSUMPTIONS TO SOLVE SJMCT-SPEA-II ALGORITHM

```
clc;
clear;
close all;
%% Problem Definition
CostFunction=@(x)MP60(x);
nVar=[5 60];             % Number of Decision Variables
VarSize=[nVar  1];    % Decision Variables Matrix Size
VarMin=-15;            % Decision Variables Lower Bound
VarMax=15;            % Decision Variables Upper Bound
%% SPEA2 Settings
MaxIt=40;           % Maximum Number of Iterations
nPop=100;             % Population Size
nArchive=60;         % Archive Size
K=round(sqrt(nPop+nArchive));  % KNN Parameter
pCrossover=0.6;
nCrossover=round(pCrossover*nPop/2)*2;
pMutation=1-pCrossover;
nMutation=nPop-nCrossover;
crossover_params.gamma=0.1;
crossover_params.VarMin=VarMin;
crossover_params.VarMax=VarMax;
mutation_params.h=0.2;
mutation_params.VarMin=VarMin;
mutation_params.VarMax=VarMax;
%% Initialization
empty_individual.Position=[];
empty_individual.Cost=[];
empty_individual.S=[];
empty_individual.R=[];
empty_individual.sigma=[];
empty_individual.sigmaK=[];
empty_individual.D=[];
empty_individual.F=[];
pop=repmat(empty_individual,nPop,1);
for i=1:nPop
    pop(i).Position=unifrnd(VarMin,VarMax,VarSize);
    pop(i).Cost=CostFunction(pop(i).Position);
end
archive=[];
%% Main Loop
```

```
for it=1:MaxIt
    Q=[pop
       archive];
    nQ=numel(Q);
    dom=false(nQ,nQ);
    for i=1:nQ
        Q(i).S=0;
    end
    for i=1:nQ
        for j=i+1:nQ
            if Dominates(Q(i),Q(j))
                Q(i).S=Q(i).S+1;
                dom(i,j)=true;
            elseif Dominates(Q(j),Q(i))
                Q(j).S=Q(j).S+1;
                dom(j,i)=true;
            end

        end
    end
    S=[Q.S];
    for i=1:nQ
        Q(i).R=sum(S(dom(:,i)));
    end
    Z=[Q.Cost]';
    SIGMA=pdist2(Z,Z,'seuclidean');
    SIGMA=sort(SIGMA);
    for i=1:nQ
        Q(i).sigma=SIGMA(:,i);
        Q(i).sigmaK=Q(i).sigma(K);
        Q(i).D=1/(Q(i).sigmaK+2);
        Q(i).F=Q(i).R+Q(i).D;
    end
    nND=sum([Q.R]==0);
    if nND<=nArchive
        F=[Q.F];
        [F, SO]=sort(F);
        Q=Q(SO);
        archive=Q(1:min(nArchive,nQ));
    else
        SIGMA=SIGMA(:,[Q.R]==0);
        archive=Q([Q.R]==0);
        k=2;
        while numel(archive)>nArchive
            while min(SIGMA(k,:))==max(SIGMA(k,:)) && k<size(SIGMA,1)
                k=k+1;
            end
            [~, j]=min(SIGMA(k,:));
            archive(j)=[];
            SIGMA(:,j)=[];
        end
    end
    PF=archive([archive.R]==0); % Approximate Pareto Front
    % Plot Pareto Front
    figure(1);
    PlotCosts(PF);
    pause(0.01);
    % Display Iteration Information
    disp(['Iteration ' num2str(it) ': Number of PF members = '
num2str(numel(PF))]);
```

```
    if it>=MaxIt
        break;
    end
    % Crossover
    popc=repmat(empty_individual,nCrossover/2,2);
    for c=1:nCrossover/2
        p1=BinaryTournamentSelection(archive,[archive.F]);
        p2=BinaryTournamentSelection(archive,[archive.F]);
        [popc(c,1).Position,
popc(c,2).Position]=Crossover(p1.Position,p2.Position,crossover_params
);
        popc(c,1).Cost=CostFunction(popc(c,1).Position);
        popc(c,2).Cost=CostFunction(popc(c,2).Position);
    end
    popc=popc(:);
    % Mutation
    popm=repmat(empty_individual,nMutation,1);
    for m=1:nMutation
        p=BinaryTournamentSelection(archive,[archive.F]);
        popm(m).Position=Mutate(p.Position,mutation_params);
        popm(m).Cost=CostFunction(popm(m).Position);
    end
    % Create New Population
    pop=[popc
         popm];
end
%% Results
 disp(' ');
 PFC = [PF.Cost];
for j=1:size(PFC,1)
    disp(['Objective #' num2str(j) ':']);
    disp(['      Min = ' num2str(min(PFC(j,:)))]);
    disp(['      Max = ' num2str(max(PFC(j,:)))]);
    disp(['    Range = ' num2str(max(PFC(j,:))-min(PFC(j,:)))]);
    disp(['    St.D. = ' num2str(std(PFC(j,:)))]);
    disp(['     Mean = ' num2str(mean(PFC(j,:)))]);
    disp(' ');
end
```

MATLAB PROGRAMMING (SECOND TEST PROBLEM) TO SOLVE SJMCT-NSGA-II AND SJMCT-SPEA-II ALGORITHM WITH DIFFERENT NUMBER OF JOBS ANDGENERATION 500

COMPUTETING THE FITNESS FUNCTION FOR 20 JOBS Z=MP20(x)

```
m=5
n=20
p=unifrnd(1,20,[m n]);
t=unifrnd(1,20,[m n]);
 for i= 1:m
   s(i)=p(i,i)
     d1=s
end
 for i= 1:m
   r(i)=t(i,i)
     r1=r
end
 for i=1:m
if s(i)==min(s)
 s(i)=s(i)+p(i,m+1)
```

```
 a6=t(i,m+1);
 break
 end
end
for j=1:m
d2(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d2)
 s(i)=min(d2)+p(i,m+2)
 a7=t(i,m+2);
 break
end
end
for j=1:m
d3(j)=[s(1,j)]
end
for i=1:m
if s(i)==min(d3)
s(i)=min(d3)+p(i,m+3)
a8=t(i,m+3);
break
end
end
 for j=1:m
d4(j)=[s(1,j)]
end
 for i=1:m
if s(i)==min(d4)
s(i)=min(d4)+p(i,m+4)
a9=t(i,m+4);
break
end
end
for j=1:m
d5(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d5)
s(i)=min(d5)+p(i,m+5)
a10=t(i,m+5);
 break
end
end
for j=1:m
d6(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d6)
s(i)=min(d6)+p(i,m+6)
a11=t(i,m+6);
 break
end
end
 for j=1:m
d7(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d7)
s(i)=min(d7)+p(i,m+7)
```

```
a12=t(i,m+7);
 break
end
end
for j=1:m
d8(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d8)
s(i)=min(d8)+p(i,m+8)
a13=t(i,m+8);
 break
end
end
for j=1:m
d9(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d9)
s(i)=min(d9)+p(i,m+9)
a14=t(i,m+9);
 break
end
end
for j=1:m
d10(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d10)
s(i)=min(d10)+p(i,m+10)
a15=t(i,m+10);
 break
end
end
for j=1:m
d11(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d11)
s(i)=min(d11)+p(i,m+11)
a16=t(i,m+11);
 break
end
end
for j=1:m
d12(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d12)
s(i)=min(d12)+p(i,m+12)
a17=t(i,m+12);
 break
end
end
 for j=1:m
d13(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d13)
s(i)=min(d13)+p(i,m+13)
```

```
a18=t(i,m+13);
 break
end
end
 for j=1:m
d14(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d14)
s(i)=min(d14)+p(i,m+14)
a19=t(i,m+14);
 break
end
end
for j=1:m
d15(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d15)
s(i)=min(d15)+p(i,m+15)
a20=t(i,m+15);
 break
end
end
 for j=1:m
d16(j)=[s(1,j)]
end
%%*****************************************
J6=d2-d1
J7=d3-d2
J8=d4-d3
J9=d5-d4
J10=d6-d5
J11=d7-d6
J12=d8-d7
J13=d9-d8
J14=d10-d9
J15=d11-d10
J16=d12-d11
J17=d13-d12
J18=d14-d13
J19=d15-d14
J20=d16-d15
TAR1=d1-r1
TAR6=max(J6)-a6
TAR7=max(J7)-a7
TAR8=max(J8)-a8
TAR9=max(J9)-a9
TAR10=max(J10)-a10
TAR11=max(J11)-a11
TAR12=max(J12)-a12
TAR13=max(J13)-a13
TAR14=max(J14)-a14
TAR15=max(J15)-a15
TAR16=max(J16)-a16
TAR17=max(J17)-a17
TAR18=max(J18)-a18
TAR19=max(J19)-a19
TAR20=max(J20)-a20
```

```
T=[TAR1,TAR6,TAR7,TAR8,TAR9,TAR10,TAR11,TAR12,TAR13,TAR14,TAR15,TAR16,
TAR17,TAR18,TAR19,TAR20]
for j=1:n
if T(j) >0
DD(j)=T(j);
else
DD(j)=0;
end
CMAX=max(s)
TARD=sum(DD)
%%
optjobs=[d1;J6;J7;J8;J9;J10;J11;J12;J13;J14;J15;J16;J17;J18;J19;J20]';
%figure(1);
%title 'parallel machine';
%barh(optjobs ,'stack');
%xlabel('JOBS')
%ylabel('MACHINE')
z1=CMAX;
z2=TARD;
z=[z1 z2]';
end
```

COMPUTETING THE FITNESS FUNCTION FOR 100 JOBS Z=MP100(x)

```
m=5
n=100
p=unifrnd(1,20,[m n]);
t=unifrnd(1,20,[m n]);
for i= 1:m
    s(i)=p(i,i)
     d1=s
end
 for i= 1:m
   r(i)=t(i,i)
     r1=r
end
 for i=1:m
if s(i)==min(s)
 s(i)=s(i)+p(i,m+1)
 a6=t(i,m+1);
 break
 end
end
for j=1:m
d2(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d2)
 s(i)=min(d2)+p(i,m+2)
 a7=t(i,m+2);
 break
end
end
 for j=1:m
d3(j)=[s(1,j)]
end
 for i=1:m
if s(i)==min(d3)
s(i)=min(d3)+p(i,m+3)
a8=t(i,m+3);
```

```
 break
end
end
 for j=1:m
d4(j)=[s(1,j)]
 end
 for i=1:m
if s(i)==min(d4)
s(i)=min(d4)+p(i,m+4)
a9=t(i,m+4);
break
end
end
for j=1:m
d5(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d5)
s(i)=min(d5)+p(i,m+5)
a10=t(i,m+5);
 break
end
end
for j=1:m
d6(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d6)
s(i)=min(d6)+p(i,m+6)
a11=t(i,m+6);
 break
end
end
 for j=1:m
d7(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d7)
s(i)=min(d7)+p(i,m+7)
a12=t(i,m+7);
 break
end
end
 for j=1:m
d8(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d8)
s(i)=min(d8)+p(i,m+8)
a13=t(i,m+8);
 break
end
end
 for j=1:m
d9(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d9)
s(i)=min(d9)+p(i,m+9)
a14=t(i,m+9);
```

```
 break
end
end
 for j=1:m
d10(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d10)
s(i)=min(d10)+p(i,m+10)
a15=t(i,m+10);
 break
end
end
for j=1:m
d11(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d11)
s(i)=min(d11)+p(i,m+11)
a16=t(i,m+11);
 break
end
end
 for j=1:m
d12(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d12)
s(i)=min(d12)+p(i,m+12)
a17=t(i,m+12);
 break
end
end
 for j=1:m
d13(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d13)
s(i)=min(d13)+p(i,m+13)
a18=t(i,m+13);
 break
end
end
 for j=1:m
d14(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d14)
s(i)=min(d14)+p(i,m+14)
a19=t(i,m+14);
 break
end
end
 for j=1:m
d15(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d15)
s(i)=min(d15)+p(i,m+15)
a20=t(i,m+15);
```

```
 break
end
end
 for j=1:m
d16(j)=[s(1,j)]
end
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 40 job
 for i= 1:m
if s(i)==min(d16)
 s(i)=min(d16)+p(i,m+16)
 a21=t(i,m+16);
 break
end
end
 for j=1:m
d17(j)=[s(1,j)]
end
 for i=1:m
if s(i)==min(d17)
s(i)=min(d17)+p(i,m+17)
a22=t(i,m+17);
break
end
end
 for j=1:m
d18(j)=[s(1,j)]
end
 for i=1:m
if s(i)==min(d18)
s(i)=min(d18)+p(i,m+18);
a23=t(i,m+18);
break
end
end
for j=1:m
d19(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d19)
s(i)=min(d19)+p(i,m+19)
a24=t(i,m+19);
 break
end
end
for j=1:m
d20(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d20)
s(i)=min(d20)+p(i,m+20)
a25=t(i,m+20);
 break
end
end
 for j=1:m
d21(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d21)
s(i)=min(d21)+p(i,m+21)
```

```
a26=t(i,m+21);
 break
end
end
 for j=1:m
d22(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d22)
s(i)=min(d22)+p(i,m+22)
a27=t(i,m+22);
 break
end
end
 for j=1:m
d23(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d23)
s(i)=min(d23)+p(i,m+23)
a28=t(i,m+23);
 break
end
end
 for j=1:m
d24(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d24)
s(i)=min(d24)+p(i,m+24)
a29=t(i,m+24);
 break
end
end
 for j=1:m
d25(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d25)
s(i)=min(d25)+p(i,m+25)
a30=t(i,m+25);
 break
end
end
 for j=1:m
d26(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d26)
s(i)=min(d26)+p(i,m+26)
a31=t(i,m+26);
 break
end
end
 for j=1:m
d27(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d27)
s(i)=min(d27)+p(i,m+27)
```

```
a32=t(i,m+27);
 break
end
end
 for j=1:m
d28(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d28)
s(i)=min(d28)+p(i,m+28)
a33=t(i,m+28);
 break
end
end
 for j=1:m
d29(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d29)
s(i)=min(d29)+p(i,m+29)
a34=t(i,m+29);
 break
end
end
 for j=1:m
d30(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d30)
s(i)=min(d30)+p(i,m+30)
a35=t(i,m+30);
 break
end
end
 for j=1:m
d31(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d31)
s(i)=min(d31)+p(i,m+31)
a36=t(i,m+31);
 break
end
end
 for j=1:m
d32(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d32)
s(i)=min(d32)+p(i,m+32)
a37=t(i,m+32);
 break
end
end
 for j=1:m
d33(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d33)
s(i)=min(d33)+p(i,m+33)
```

```
a38=t(i,m+33);
 break
end
end
 for j=1:m
d34(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d34)
s(i)=min(d34)+p(i,m+34)
a39=t(i,m+34);
 break
end
end
 for j=1:m
d35(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d35)
s(i)=min(d35)+p(i,m+35)
a40=t(i,m+35);
 break
end
end
 for j=1:m
d36(j)=[s(1,j)]
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%60 job
for i= 1:m
if s(i)==min(d36)
 s(i)=min(d36)+p(i,m+36)
 a41=t(i,m+36);
 break
end
end
 for j=1:m
d37(j)=[s(1,j)]
end
for i=1:m
if s(i)==min(d37)
s(i)=min(d37)+p(i,m+37)
a42=t(i,m+37);
break
end
end
 for j=1:m
d38(j)=[s(1,j)]
end
 for i=1:m
if s(i)==min(d38)
s(i)=min(d38)+p(i,m+38)
a43=t(i,m+38);
break
end
end
for j=1:m
d39(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d39)
```

```
s(i)=min(d39)+p(i,m+39)
a44=t(i,m+39);
 break
end
end
for j=1:m
d40(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d40)
s(i)=min(d40)+p(i,m+40)
a45=t(i,m+40);
 break
end
end
 for j=1:m
d41(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d41)
s(i)=min(d41)+p(i,m+41)
a46=t(i,m+41);
 break
end
end
 for j=1:m
d42(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d42)
s(i)=min(d42)+p(i,m+42)
a47=t(i,m+42);
 break
end
end
 for j=1:m
d43(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d43)
s(i)=min(d43)+p(i,m+43)
a48=t(i,m+43);
 break
end
end
 for j=1:m
d44(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d44)
s(i)=min(d44)+p(i,m+44)
a49=t(i,m+44);
 break
end
end
 for j=1:m
d45(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d45)
```

```
s(i)=min(d45)+p(i,m+45)
a50=t(i,m+45);
 break
end
end
 for j=1:m
d46(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d46)
s(i)=min(d46)+p(i,m+46)
a51=t(i,m+46);
 break
end
end
 for j=1:m
d47(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d47)
s(i)=min(d47)+p(i,m+47)
a52=t(i,m+47);
 break
end
end
 for j=1:m
d48(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d48)
s(i)=min(d48)+p(i,m+48)
a53=t(i,m+48);
 break
end
end
 for j=1:m
d49(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d49)
s(i)=min(d49)+p(i,m+49)
a54=t(i,m+49);
 break
end
end
 for j=1:m
d50(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d50)
s(i)=min(d50)+p(i,m+50)
a55=t(i,m+50);
 break
end
end
 for j=1:m
d51(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d51)
```

```
s(i)=min(d51)+p(i,m+51)
a56=t(i,m+51);
 break
end
end
 for j=1:m
d52(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d52)
s(i)=min(d52)+p(i,m+52)
a57=t(i,m+52);
 break
end
end
for j=1:m
d53(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d53)
s(i)=min(d53)+p(i,m+53)
a58=t(i,m+53);
 break
end
end
for j=1:m
d54(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d54)
s(i)=min(d54)+p(i,m+54)
a59=t(i,m+54);
 break
end
end
for j=1:m
d55(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d55)
s(i)=min(d55)+p(i,m+55)
a60=t(i,m+55);
 break
end
end
 for j=1:m
d56(j)=[s(1,j)]
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%80 job
for i= 1:m
if s(i)==min(d56)
 s(i)=min(d56)+p(i,m+56)
 a61=t(i,m+56);
 break
end
end
 for j=1:m
d57(j)=[s(1,j)]
end
 for i=1:m
```

```
if s(i)==min(d57)
s(i)=min(d57)+p(i,m+57)
a62=t(i,m+57);
break
end
end
 for j=1:m
d58(j)=[s(1,j)]
end
 for i=1:m
if s(i)==min(d58)
s(i)=min(d58)+p(i,m+58)
a63=t(i,m+58);
break
end
end
for j=1:m
d59(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d59)
s(i)=min(d59)+p(i,m+59)
a64=t(i,m+59);
 break
end
end
for j=1:m
d60(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d60)
s(i)=min(d60)+p(i,m+60)
a65=t(i,m+60);
 break
end
end
 for j=1:m
d61(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d61)
s(i)=min(d61)+p(i,m+61)
a66=t(i,m+61);
 break
end
end
 for j=1:m
d62(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d62)
s(i)=min(d62)+p(i,m+62)
a67=t(i,m+62);
 break
end
end
 for j=1:m
d63(j)=[s(1,j)]
end
for i= 1:m
```

```
if s(i)==min(d63)
s(i)=min(d63)+p(i,m+63)
a68=t(i,m+63);
 break
end
end
 for j=1:m
d64(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d64)
s(i)=min(d64)+p(i,m+64)
a69=t(i,m+64);
 break
end
end
 for j=1:m
d65(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d65)
s(i)=min(d65)+p(i,m+65)
a70=t(i,m+65);
 break
end
end
 for j=1:m
d66(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d66)
s(i)=min(d66)+p(i,m+66)
a71=t(i,m+66);
 break
end
end
 for j=1:m
d67(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d67)
s(i)=min(d67)+p(i,m+67)
a72=t(i,m+67);
 break
end
end
 for j=1:m
d68(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d68)
s(i)=min(d68)+p(i,m+68)
a73=t(i,m+68);
 break
end
end
 for j=1:m
d69(j)=[s(1,j)]
end
 for i= 1:m
```

```
if s(i)==min(d69)
s(i)=min(d69)+p(i,m+69)
a74=t(i,m+69);
 break
end
end
 for j=1:m
d70(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d70)
s(i)=min(d70)+p(i,m+70)
a75=t(i,m+70);
 break
end
end
 for j=1:m
d71(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d71)
s(i)=min(d71)+p(i,m+71)
a76=t(i,m+71);
 break
end
end
 for j=1:m
d72(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d72)
s(i)=min(d72)+p(i,m+72)
a77=t(i,m+72);
 break
end
end
for j=1:m
d73(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d73)
s(i)=min(d73)+p(i,m+73)
a78=t(i,m+73);
 break
end
end
 for j=1:m
d74(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d74)
s(i)=min(d74)+p(i,m+74)
a79=t(i,m+74);
 break
end
end
 for j=1:m
d75(j)=[s(1,j)]
end
for i= 1:m
```

```
if s(i)==min(d75)
s(i)=min(d75)+p(i,m+75)
a80=t(i,m+75);
 break
end
end
 for j=1:m
d76(j)=[s(1,j)]
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%100 job
for i= 1:m
if s(i)==min(d76)
 s(i)=min(d76)+p(i,m+76)
 a81=t(i,m+76);
 break
end
end
 for j=1:m
d77(j)=[s(1,j)]
end
for i=1:m
if s(i)==min(d77)
s(i)=min(d77)+p(i,m+77)
a82=t(i,m+77);
break
end
end
for j=1:m
d78(j)=[s(1,j)]
end
for i=1:m
if s(i)==min(d78)
s(i)=min(d78)+p(i,m+78)
a83=t(i,m+78);
break
end
end
for j=1:m
d79(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d79)
s(i)=min(d79)+p(i,m+79)
a84=t(i,m+79);
 break
end
end
for j=1:m
d80(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d80)
s(i)=min(d80)+p(i,m+80)
a85=t(i,m+80);
 break
end
end
for j=1:m
d81(j)=[s(1,j)]
end
```

```
for i= 1:m
if s(i)==min(d81)
s(i)=min(d81)+p(i,m+81)
a86=t(i,m+81);
 break
end
end
 for j=1:m
d82(j)=[s(1,j)]
end
 for i= 1:m
if s(i)==min(d82)
s(i)=min(d82)+p(i,m+82)
a87=t(i,m+82);
 break
end
end
for j=1:m
d83(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d83)
s(i)=min(d83)+p(i,m+83)
a88=t(i,m+83);
 break
end
end
for j=1:m
d84(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d84)
s(i)=min(d84)+p(i,m+84)
a89=t(i,m+84);
 break
end
end
for j=1:m
d85(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d85)
s(i)=min(d85)+p(i,m+85)
a90=t(i,m+85);
 break
end
end
for j=1:m
d86(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d86)
s(i)=min(d86)+p(i,m+86)
a91=t(i,m+86);
 break
end
end
for j=1:m
d87(j)=[s(1,j)]
end
```

```
for i= 1:m
if s(i)==min(d87)
s(i)=min(d87)+p(i,m+87)
a92=t(i,m+87);
 break
end
end
for j=1:m
d88(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d88)
s(i)=min(d88)+p(i,m+88)
a93=t(i,m+88);
 break
end
end
for j=1:m
d89(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d89)
s(i)=min(d89)+p(i,m+89)
a94=t(i,m+89);
 break
end
end
for j=1:m
d90(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d90)
s(i)=min(d90)+p(i,m+90)
a95=t(i,m+90);
 break
end
end
for j=1:m
d91(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d91)
s(i)=min(d91)+p(i,m+91)
a96=t(i,m+91);
 break
end
end
for j=1:m
d92(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d92)
s(i)=min(d92)+p(i,m+92)
a97=t(i,m+92);
 break
end
end
for j=1:m
d93(j)=[s(1,j)]
end
```

```
for i= 1:m
if s(i)==min(d93)
s(i)=min(d93)+p(i,m+93)
a98=t(i,m+93);
 break
end
end
for j=1:m
d94(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d94)
s(i)=min(d94)+p(i,m+94)
a99=t(i,m+94);
 break
end
end
 for j=1:m
d95(j)=[s(1,j)]
end
for i= 1:m
if s(i)==min(d95)
s(i)=min(d95)+p(i,m+95)
a100=t(i,m+95);
 break
end
end
for j=1:m
d96(j)=[s(1,j)]
end
%%******************************************
J6=d2-d1
J7=d3-d2
J8=d4-d3
J9=d5-d4
J10=d6-d5
J11=d7-d6
J12=d8-d7
J13=d9-d8
J14=d10-d9
J15=d11-d10
J16=d12-d11
J17=d13-d12
J18=d14-d13
J19=d15-d14
J20=d16-d15
J21=d17-d16
J22=d18-d17
J23=d19-d18
J24=d20-d19
J25=d21-d20
J26=d22-d21
J27=d23-d22
J28=d24-d23
J29=d25-d24
J30=d26-d25
J31=d27-d26
J32=d28-d27
J33=d29-d28
J34=d30-d29
```

J35=d31-d30
J36=d32-d31
J37=d33-d32
J38=d34-d33
J39=d35-d34
J40=d36-d35
J41=d37-d36
J42=d38-d37
J43=d39-d38
J44=d40-d39
J45=d41-d40
J46=d42-d41
J47=d43-d42;
J48=d44-d43
J49=d45-d44
J50=d46-d45
J51=d47-d46
J52=d48-d47
J53=d49-d48
J54=d50-d49
J55=d51-d50
J56=d52-d51
J57=d53-d52
J58=d54-d53
J59=d55-d54
J60=d56-d55
J61=d57-d56
J62=d58-d57
J63=d59-d58
J64=d60-d59
J65=d61-d60
J66=d62-d61
J67=d63-d62
J68=d64-d63
J69=d65-d64
J70=d66-d65
J71=d67-d66
J72=d68-d67
J73=d69-d68
J74=d70-d69
J75=d71-d70
J76=d72-d71
J77=d73-d72
J78=d74-d73
J79=d75-d74
J80=d76-d75
J81=d77-d76
J82=d78-d77
J83=d79-d78
J84=d80-d79
J85= d81-d80
J86= d82-d81
J87= d83-d82
J88= d84-d83
J89= d85-d84
J90=d86-d85
J91=d87-d86
J92=d88-d87
J93=d89-d88
J94=d90-d89

J95=d91-d90
J96=d92-d91
J97=d93-d92
J98=d94-d93
J99=d95-d94
J100=d96-d95
TAR1=d1-r1
TAR6=max(J6)-a6
TAR7=max(J7)-a7
TAR8=max(J8)-a8
TAR9=max(J9)-a9
TAR10=max(J10)-a10
TAR11=max(J11)-a11
TAR12=max(J12)-a12
TAR13=max(J13)-a13
TAR14=max(J14)-a14
TAR15=max(J15)-a15
TAR16=max(J16)-a16
TAR17=max(J17)-a17
TAR18=max(J18)-a18
TAR19=max(J19)-a19
TAR20=max(J20)-a20
TAR21=max(J21)-a21
TAR22=max(J22)-a22
TAR23=max(J23)-a23
TAR24=max(J24)-a24
TAR25=max(J25)-a25
TAR26=max(J26)-a26
TAR27=max(J27)-a27
TAR28=max(J28)-a28
TAR29=max(J29)-a29
TAR30=max(J30)-a30
TAR31=max(J31)-a31
TAR32=max(J32)-a32
TAR33=max(J33)-a33
TAR34=max(J34)-a34
TAR35=max(J35)-a35
TAR36=max(J36)-a36
TAR37=max(J37)-a37
TAR38=max(J38)-a38
TAR39=max(J39)-a39
TAR40=max(J40)-a40
TAR41=max(J41)-a41
TAR42=max(J42)-a42
TAR43=max(J43)-a43
TAR44=max(J44)-a44
TAR45=max(J45)-a45
TAR46=max(J46)-a46
TAR47=max(J47)-a47
TAR48=max(J48)-a48
TAR49=max(J49)-a49
TAR50=max(J50)-a50
TAR51=max(J51)-a51
TAR52=max(J52)-a52
TAR53=max(J53)-a53
TAR54=max(J54)-a54
TAR55=max(J55)-a55
TAR56=max(J56)-a56
TAR57=max(J57)-a57
TAR58=max(J58)-a58

```
TAR59=max(J59)-a59
TAR60=max(J60)-a60
TAR61=max(J61)-a61
TAR62=max(J62)-a62
TAR63=max(J63)-a63
TAR64=max(J64)-a64
TAR65=max(J65)-a65
TAR66=max(J66)-a66
TAR67=max(J67)-a67
TAR68=max(J68)-a68
TAR69=max(J69)-a69
TAR70=max(J70)-a70

TAR71=max(J71)-a71
TAR72=max(J72)-a72
TAR73=max(J73)-a73
TAR74=max(J74)-a74
TAR75=max(J75)-a75
TAR76=max(J76)-a76
TAR77=max(J77)-a77
TAR78=max(J78)-a78
TAR79=max(J79)-a79
TAR80=max(J80)-a80
TAR81=max(J81)-a81
TAR82=max(J82)-a82
TAR83=max(J83)-a83
TAR84=max(J84)-a84
TAR85=max(J85)-a85
TAR86=max(J86)-a86
TAR87=max(J87)-a87
TAR88=max(J88)-a88
TAR89=max(J89)-a89
TAR90=max(J90)-a90
TAR91=max(J91)-a91
TAR92=max(J92)-a92
TAR93=max(J93)-a93
TAR94=max(J94)-a94
TAR95=max(J95)-a95
TAR96=max(J96)-a96
TAR97=max(J97)-a97
TAR98=max(J98)-a98
TAR99=max(J99)-a99
TAR100=max(J100)-a100
T=[TAR1,TAR6,TAR7,TAR8,TAR9,TAR10,TAR11,TAR12,TAR13,TAR14,TAR15,TAR16,
TAR17,TAR18,TAR19,TAR20,TAR21,TAR22,TAR23,TAR24,TAR25,TAR26,TAR27,TAR2
8,TAR29,TAR30,TAR31,TAR32,TAR33,TAR34,TAR35,TAR36,TAR37,TAR38,TAR39,TA
R40,TAR41,TAR42,TAR43,TAR44,TAR45,TAR46,TAR47,TAR48,TAR49,TAR50,TAR51,
TAR52,TAR53,TAR54,TAR55,TAR56,TAR57,TAR58,TAR59,TAR60,TAR61,TAR62,TAR6
3,TAR64,TAR65,TAR66,TAR67,TAR68,TAR69,TAR70,TAR71,TAR72,TAR73,TAR74,TA
R75,TAR76,TAR77,TAR78,TAR79,TAR80,TAR81,TAR82,TAR83,TAR84,TAR85,TAR86,
TAR87,TAR88,TAR89,TAR90,TAR91,TAR92,TAR93,TAR94,TAR95,TAR96,TAR97,TAR9
8,TAR99,TAR100]
for j=1:n
if T(j) >0
DD(j)=T(j);
else
  DD(j)=0;
end
CMAX=max(s)
TARD=sum(DD)
```

```
%%%%%
optjobs=[d1;J6;J7;J8;J9;J10;J11;J12;J13;J14;J15;J16;J17;J18;J19;J20;J2
1;J22;J23;J24;J25;J26;J27;J28;J29;J30;J31;J32;J33;J34;J35;J36;J37;J38;
J39;J40;J41;J42;J43;J44;J45;J46;J47;J48;J49;J50;J51;J52;J53;J54;J55;J5
6;J57;J58;J59;J60;J61;J62;J63;J64;J65;J66;J67;J68;J69;J70;J71;J72;J73;
J74;J75;J76;J77;J78;J79;J80;J81;J82;J83;J84;J85;J86;J87;J88;J89;J90;J9
1;J92;J93;J94;J95;J96;J97;J98;J99;J100]';
%figure(1);
%title 'parallel machine';
%barh(optjobs ,'stack');
%xlabel('JOBS')
%ylabel('MACHINE')

z1=CMAX;
z2=TARD;
z=[z1 z2]';
end
```