YAŞAR UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

PhD THESIS

Metaheuristics for the Permutation Flow Shop Problems

Yavuz İNCE

Thesis Advisor: Assist. Prof. Dr. Korhan KARABULUT
Co-Advisor: Prof. Dr. Mehmet Fatih TAŞGETİREN

Department of Computer Engineering

Presentation Date: 29.01.2016

Bornova-İZMİR
2016

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

Assist. Prof. Dr. Korhan KARABULUT (Supervisor)

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

Prof. Dr. Mehmet Fatih TAŞGETİREN (Co-Supervisor)

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

Prof. Dr. Mehmet Cudi OKUR

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

Assoc. Prof. Dr. Aybars UĞUR

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

Assist. Prof. Dr. Mete EMİNAĞAOĞLU

Prof. Dr. Cüneyt GÜZELİŞ

Director of the Graduate School

ABSTRACT

Metaheuristics for the Permutation Flow Shop Problems

İNCE, Yavuz
PhD in Computer Engineering
Supervisor: Assist. Prof. Dr. Korhan KARABULUT
Co-Supervisor: Prof. Dr. Mehmet Fatih TAŞGETİREN
January 2016, 120 pages

In this study, two variants of permutation flow shop scheduling problem with sequence dependent setup times are considered. The first problem studied in this thesis is the permutation flow shop problem with sequence dependent setup times under makespan criterion. A new iterated greedy algorithm and a new local search algorithm is developed for this problem. The new local search includes insertion neighborhood and swap neighborhood. A new speed up technique is developed to reduce the cost of the swap neighborhood search, which is inspired from Taillard's well-known speed-up method for the insertion neighborhood. The developed speed up technique can save fifty percent CPU time in average. The developed iterated greedy algorithm utilizing the new swap speed-up method is tested on the benchmark instances from the literature and new best-known solutions are found for 250 out of 480 problem instances. The second problem considered is the permutation flow shop scheduling problem with sequence dependent setup times under total flow time criterion. This problem is studied for the first time in the literature to best of our knowledge. NEH_EDD and LR heuristics as well as speed-up methods for problems without the sequence dependent setup times for insertion and swap neighborhoods are adapted to this problem. Several metaheuristics are developed and executed on a benchmark set. The performances of the developed algorithms are compared and the results are presented.

Keywords—Sequence-dependent setup times, flow shop scheduling problem, metaheuristics, iterated greedy algorithm, variable neighborhood search, makespan, total flow time.

ÖZET

Permütasyon Akış Tipi Çizelgeleme Problemleri için Meta-Sezgisel Algoritmalar

Yavuz İNCE
Doktora Tezi, Bilgisayar Mühendisliği Bölümü
Tez Danışmanı: Yrd. Doç. Dr. Korhan KARABULUT
İkinci Danışmanı: Prof. Dr. Mehmet Fatih TAŞGETİREN
Ocak 2016, 120 sayfa

Bu tezde, sıra bağımlı hazırlık süreli permütasyon akış tipi çizelgeleme probleminin iki tane farklı varyasyonu ele alınmıştır. İlk olarak sıra bağımlı hazırlık süreli permütasyon akış tipi çizelgeleme probleminde tamamlanma süresinin en iyilenmesi çalışılmıştır. Bu problem için yeni bir yenilemeli açgözlü algoritma ve yeni yerel arama algoritması geliştirilmiştir. Yeni yerel arama algoritmasında araya sokma ve karşılıklı yer değiştirme komşulukları kullanılmaktadır. Karşılıklı yer değiştirme komşuluğunun hesaplama zamanını azaltabilmek için Taillard'ın araya sokma komşuluğu hesaplama yönteminden esinlenerek bir hızlandırma yöntemi geliştirilmiştir. Yeni geliştirilen bu hızlandırma yöntemi karşılıklı yer değiştirme komşuluğunun hesaplanma süresini ortalama olarak yüzde elli oranında azaltmaktadır. Geliştirilen hızlandırma yöntemini kullanan yenilemeli açgözlü algoritma literatürde kullanılan bir test kümesindeki problemler için çalıştırılmış ve sonuç olarak bilinen en iyi 480 sonuçtan 250 tanesi için yeni en iyi sonuç bulunmuştur. Tez kapsamında ikinci olarak sıra bağımlı hazırlık süreli permütasyon akış tipi çizelgeleme probleminde akış süresi en iyileme çalışılmıştır. Literatürde bu problem ilk defa çalışılmıştır. Sıra bağımlı hazırlık süresi olmayan NEH_DD ve LR sezgisel algoritmaları ve karşılıklı yer değiştirme ve araya sokma komşulukları için hızlandırma yöntemleri bu probleme uyarlanmıştır. Birden fazla sezgi ötesi algoritma geliştirilmiş ve test kümesindeki problemler için çalıştırılmıştır. Tüm algoritmaların başarım sonuçları karşılaştırılmış ve sonuçlar sunulmuştur.

Anahtar Kelimeler - Sıra bağımlı hazırlık süresi, akış tipi çizelgeleme problemi, sezgi ötesi algoritmalar, yenilemeli açgözlü algoritma, değişken komşuluk yapılı arama algoritması, üretim süresi, toplam akış zamanı

ACKNOWLEDGEMENTS

I am grateful to my supervisor, Assist. Prof. Dr. Korhan KARABULUT and co-supervisor, Prof. Dr. Mehmet Fatih TAŞGETİREN, for their endless support. I am thankful for their various ideas, suggestions and for all of their encouraging words in the moments of adversity.

And I would like to thank to Assist. Prof. Dr. Mete EMİNAĞAOĞLU for his support and advices during the thesis progress meetings.

And I want to thank Gökhan AKYOL for his support.

Finally, I thank my family for their constant encouragement and motivation during my studies.

Yavuz İNCE
İzmir, 2016

TEXT OF OATH

I declare and honestly confirm that my study, titled “Metaheuristics for The Permutation Flow Shop Problems” and presented as a Doctorate Thesis, has been written without applying to any assistance inconsistent with scientific ethics and traditions, that all sources from which I have benefited are listed in the bibliography, and that I have benefited from these sources by means of making references.

29/01/2016

Yavuz İNCE

TABLE OF CONTENTS

INDEX OF FIGURES

INDEX OF TABLES

INDEX OF SYMBOLS AND ABBREVIATIONS

Symbols	Explanations
C_{max}	Makespan
π	Permutation of jobs
$F_m\|\, s_{jk}, prmu\, \|C_{max}$	m machine permutation flowshop makespan minimization problem with sequence dependent setup times
j, n	jobs, total number of jobs
k, m	machines, total number of machines
S_k	starting time of jobs on machine k
$e_{i,j}$	the starting time of i^{th} job on j^{th} machine in forward calculation matrix
S_{ijk}	Sequence dependent set up time of k^{th} job on i^{th} machine where j is the previous job completed on i^{th} machine
$F_m\|\, s_{jk}, prmu\, \|\sum C_i$	m machine permutation flowshop total flowtime minimization problem with sequence dependent setup times
$q_{i,j}$	the starting time of i^{th} job on j^{th} machine in backwards calculation

	matrix
$f_{i,j}$	the finishing time i^{th} job on j^{th} machine in final calculation matrix

Abbreviations

ARPD	Average Relative Percentage Deviation
FJSRA	Fictitious Job Setup Ranking Algorithm
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
IG	Iterated Greedy
LIT	Less Idle Times
LS	Local Search
NEH	Nawaz, Enscore, Ham Heuristic
NP	Non-deterministic Polynomial-time
PFSP	Permutation Flowshop Problem
SDST	Sequence Dependent Setup Time

SPD	Smallest Process Distance
SRA	Setup Ranking Algorithm
VLS	Variable Local Search
VND	Variable Neighborhood Descent
VNS	Variable Neighborhood Search
WY	Woo and Yim Algorithm

1. INTRODUCTION

Scheduling is determining the order of the jobs to be handled on machines. A schedule can be considered as a plan for the execution of jobs on machines. Efficient scheduling is very important for production and manufacturing. Benefits of an efficient schedule can be increased resource utilization and production process efficiency, reduced inventory and more accurate handling of due dates. Characteristics of the jobs such as their sizes and routes through machines greatly influence the result of the scheduling process. Also all technological constraints should be considered for a feasible schedule. According to Wight (1984), there are two important decision criteria for manufacturing system scheduling; which are "priorities" and "capacity". These are the answers for two questions: "what will I do next?" and "where will I do it?" Cox et al. (1992) define scheduling as "the actual assignment of starting and/or completion dates to operations or groups of operations to show when these must be done if the manufacturing order is to be completed on time". Henry Gantt proposed a type of bar chart for illustrating job schedules in 1910s, which is called as a Gantt chart. Even today, Gantt charts are widely used for presenting the schedules. In the early years, the term scheduling was only used for scheduling of the manufacturing systems. Now, scheduling is also very important for non-manufacturing areas.

Fundamental structure in scheduling is generally called as jobs and this term is also used for non-manufacturing environments. Jobs may consist of one or more tasks. The main problem in scheduling is to determine the order of the tasks according the priorities and availability of the resources. Scheduling can be considered as decision-making process of ordering the tasks according to some constraints in order to optimize one or more criteria. For manufacturing systems, scheduling plays an important role in production planning. Better scheduling allows to use the production environment more efficiently and to make better resource allocation.

Nowadays, fast growing markets and manufacturing systems deal with higher customer expectations in terms of quality of the product, cost of the product and finally its arrival time. Satisfying these conditions is getting harder each day with the increasing customer demands. To catch up with current situation, manufacturing enterprises focus on two main issues. The first issue is using more technological production lines for increasing the production rate and lowering the production time. Also new technologies are important for producing more reliable products with lower unit costs. Flexibility is another important production requirement for the market in order to make changes fast enough to catch up with customers' needs or demands. Second issue is adopting a more utilized production structure that respects resource utilization, inventory costs, and production and manufacturing times. Under these circumstances, manufacturers spend their efforts on achieving production goals that are best for themselves and customers. As it is mentioned above, these goals are also important for non-manufacturing markets or industries.

Scheduling is considered as a decision-making process regarding the current situation of the system. It tries to optimize the system with respect to one or more objectives considering the state of the resources. Resources can be crew in an airport, nurses in a hospital or production components in production lines. Tasks of the operations may differ with the scheduling environment or industry. Schedules also might have different objectives to satisfy the needs and demands; in order to increase the production utility, the objective can be obtaining minimum make span, while in order to minimize the inventory, the objective can be minimizing total flow time or in order to catch the specific production time, the objective can be minimizing tardiness. Inadequate scheduling causes inefficient utilization of production facilities and employees. Moreover, it increases the idle time in production. As a result, this will increase the costs.

1.1 Classification of Scheduling Problems

Scheduling is the process of finding a feasible order for processing in order to optimize the production or system. In 1981, Graves classified the scheduling problems and put them into three main and two additional categories. In 2012, Dhingra summarized these categories as follows:

- Requirement generation
- Processing complexity
- Scheduling criteria
- Parameter variability
- Scheduling environment

In the first category, jobs are categorized by their stocking attribute. If orders are not stocked and produced with demand of the customer, this kind of jobs are considered as "open shop" and there is no inventory in this kind of scheduling. In closed shops, production is not only going to be determined by customer demands, it will also produce inventory after production. For closed shop type problems, host sequencing problem and lot-sizing decision has to be made for current inventory. Job shop and flow shop problems are considered in closed shop category. Scheduling problems can also be divided into categories by their processing complexity as follows (Graves,1981):

- One-stage, one processor (facility)
- One-stage, parallel processors (facilities)
- Multistage, flow shop
- Multistage, job shop

In single machine problems, there is only one processing step. All jobs have only one non-repeated task to be processed on the single machine. One stage, parallel processors can be described similar to single machine parallel shops. Again, all jobs have a single task but this single task can be processed on parallel machines. This means, two or more jobs can be processed at the same time (on different machines) in parallel machine systems. In multistage problems, each job has more than one task to be processed on different facilities or machines. Flow shop problem is a special case of the multistage problems in which all tasks in all jobs follow the same order through each facility or machine. In the more general case of the multistage problem, named as job shop problem, again all jobs have more than one operation to be processed on the machines but each job has a different operation order (route) through the machines.

The third classification category for the scheduling problems is the scheduling objective according to Graves (1981). In this scheme, scheduling problems are classified according to their schedule cost and schedule performance. Schedule cost includes all expenses for production such as production setup or changeovers in inventory holding cost, etc. Performance of the schedule gives information about optimization criteria, which is the objective for the current state of the system. These performance measures can be utilization of the production lines, in other words minimizing the makespan of the schedule, total completion time of all jobs named as total flow time or average or maximum tardiness with respect to the due dates of the products.

There are two more schemes that can be used for classification of scheduling problems: parameter variability and scheduling environment. In parameter variability, scheduling problems can be divided into two groups as deterministic and stochastic schedules. In scheduling environment, schedules can be categorized as static and dynamic schedules. In static schedules, all requirements are fully specified before the scheduling process and no additional requirements will be added to problem set later. Most of the scheduling problems are deterministic and static. A classification of the scheduling problems is given in Figure 1-1.

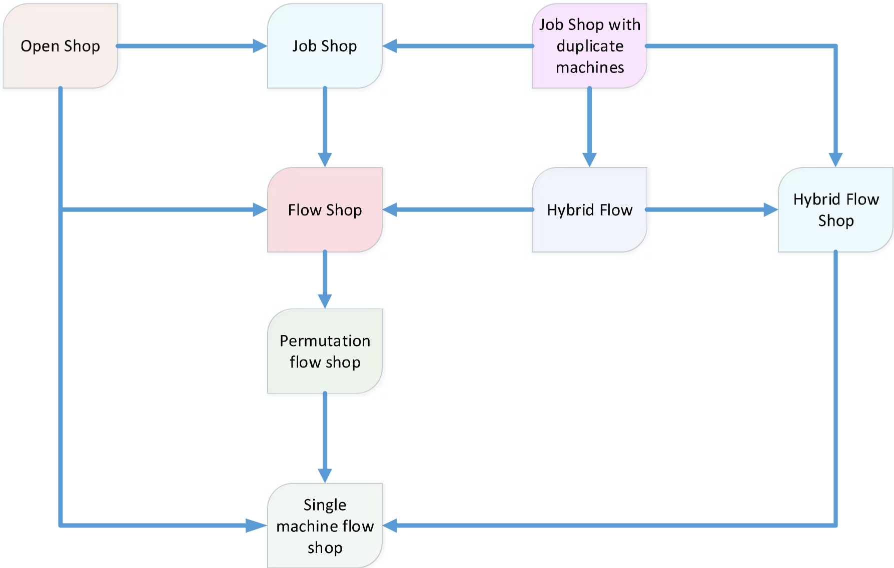

Figure 1-1 A classification of scheduling problems (Dhingra, 2012)

1.2 Permutation Flow Shop Problem

Scheduling problems are keeping their popularity since 1950s when the first seminal publications (Smith, 1956), (Johnson S. , 1954) (Jackson, 1955) began to appear. There are various kinds of scheduling problems that are being studied since. The most general scheduling problem is the job shop scheduling problem. In job shop scheduling problem, there is a finite set of n jobs and these jobs consist of m ordered operations. There are m machines and each can handle at most one operation at a time. Each operation is processed on the machines without interruption. Main purpose is to find a schedule, which optimizes a chosen objective. For job shop scheduling problem there are $(n!)^m$ possible sequences. Gantt chart of a sample schedule for abz06 instance is shown in Figure 1-2.

Figure 1-2 Gantt chart for a sample schedule for sample instance

Flow shop scheduling problem is one of the most popular scheduling problems. Permutation flow shop problem is a special kind of job shop problem in which all jobs visit each machine in the same sequence. Several criteria can be used to consider the performance of the decision making problem for scheduling, such as makespan which deals with maximum completion time of jobs in all machines, total tardiness which deals with tardiness of the jobs in all machines and total flow time which deals with minimizing inventory costs. Makespan criteria is important for machine utilization (Pan & Ruiz, 2013), while flow time criteria focuses on minimizing the in-process in reserves (Dipak & Sarin, 2008), and tardiness criteria satisfies the customer due dates as a hard deadline constraint (VictorFernandez-Viagas, 2015). There are $n!$ possible sequences for jobs in permutation flow shop problem. Gantt chart of a sample schedule for Ta001 instance is shown in Figure 1-3.

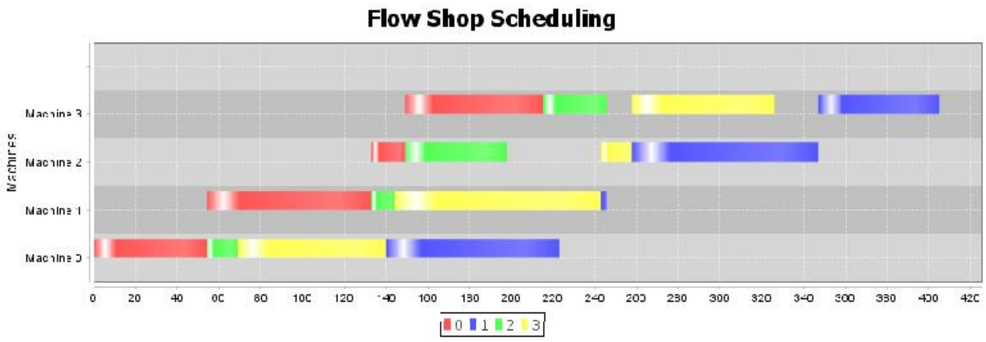

Figure 1-3 Gantt chart for a sample schedule for Ta001 instance

1.3 Set up time and Sequence Dependent Set up Time

In real life, preparation of the production environment (changing the operator controls for new parts, cleaning up the production line for new order, adjusting the production line etc.) takes some time, which is called as set up time. Changing the blades for new paper size or preparing the paint tank for new color production can be considered as real life examples. Actually, most of the production systems that deal with different kinds of products need such setup times in order to make some adjustments for the new piece of product. In some cases, the time spent on adjusting the production line or cleaning up the production environment may show differences with respect to job order to be processed on the machine. For example, in paint production, it takes more time to produce white color paint in a tank which was already used for black color paint production, than producing a dark blue color paint in the same tank. In the former case, in addition to required setup time, more water will be needed to wash the tank. If the setup time changes with respect to the previous task that was executed on the machine and the next task that will be executed, then this kind of setup time is called as sequence dependent set up time.

Sequence dependent setup times have significant importance for the production systems. Luh et al. (1998) designed a system for Toshiba's gas insulated switchgears (GIS). In this study, it is reported that model performance in handling of the sequence dependent setup time has a critical effect. As Pinedo (2008) mentions, machine efficiency can be improved up to 20% with correctly handling the sequence dependent set up times in flow shop problems. There is more research on the effect of the sequence set up times in production such as Yi and Wang (2003) and Gendreaua et al. (2001) that showed the significance (impact) of sequence dependent set up times in different cases.

Setup time means preparation of the machine in order to start production and it includes the time needed for setting up the environment, adjusting the system etc. (Allahverdi et.al, 1999). In some applications, this setup time is added to the processing time and neglected. When dealing with separate set up times, two kinds of setup times can be seen in the literature. In the first kind, the job type determines the setup time, so it can be named as sequence-independent setup time. In the second kind, both job and machine determines the setup time, so this can be named as sequence dependent set-up (Allahverdi et.al, 1999).

Permutation flow shop scheduling problem with sequence dependent setup time (PFSP - SDST), is also be named as the sequence-dependent setup time flow shop scheduling problem (SDST - FSP) in the literature. In SDST - FSP, setup times (costs) are processed separately instead of being added to processing times of the jobs. While the permutation flow shop problem is popular among researchers, sequence dependent setup time version of the problem has not been studied much. However, set up times have significant effects as shown by many researchers. Wilbrecht and Prescott (1969) showed that SDST has a reasonably large amount of effect when the system operates near the limits.

Maximizing throughput is one of the important goals of scheduling. High utilization and high throughput can be achieved by achieving the optimum makespan schedule for the machines. So, machines will have less idle time and this will lead to higher equipment efficiency.

Flow time can be described as the time consumed by the processes (jobs) on machines. Minimizing the flow time as a scheduling criteria makes fewer inventories for the system and minimizes the mean number of processes (jobs) in the system (Baker & Trietsch, 2009) . In addition, minimum flow time values lead to less cycle time for manufacturing (Ciavotta, Minella, & Ruiz, 2010).

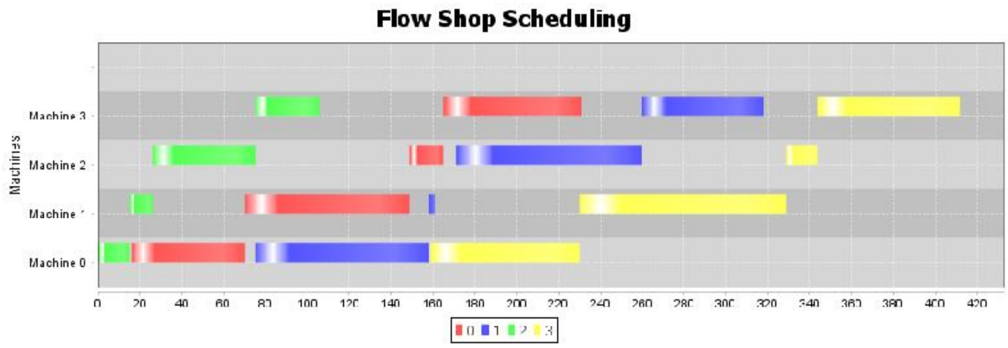

Figure 1-4 Gantt chart of a sample schedule for Ta001 instance with SDST

Gantt chart of a sample schedule for Taillard's instance Ta001 with large sequence dependent setup times is shown in Figure 1-4. The effect of the sequence dependent setup time is obvious when Figure 1-3 and Figure 1-4 are compared. Figure 1-3 shows the Gantt chart of the same instance without sequence dependent setup times. As observed from Figure 1-4, the total completion of all of the operations has been delayed from 1300 to 2050 when the setup times are considered.

1.4 Scope of the Work

In this thesis; metaheuristic approaches for permutation flow shop problem with sequence dependent set up times have been studied. Two different optimization criteria are considered. The first optimization criterion is the minimization of make span. A novel speedup method for the swap neighborhood is developed for this problem. The proposed speedup method is inspired from the well-known Taillard's speedup method for the insertion neighborhood. A new iterated greedy (IG) algorithm with a local search procedure that utilizes the developed speed up calculation technique for swap neighborhood in addition to insertion neighborhood is developed. The developed IG algorithm is compared to other metaheuristics from the literature.

The second optimization objective considered in the thesis is total flow time minimization for permutation flow shop problem with sequence dependent set up times. For this problem, Li's (Li, Wang, & Wu, 2009) speed-up calculation method as well as NEH_EDD and LR heuristics are adapted to consider the sequence dependent set up times and new local search algorithms are proposed. Experiments are carried out in order to tune the parameters of the implemented metaheuristics. The results of the performances of all implemented algorithms are presented in computational results section and new local minimum values that are obtained by proposed methods are given in appendices.

1.5 Organization of the Thesis

The rest of this thesis is organized as follows: in the second chapter, formal problem descriptions and literature review for the permutation flow shop problem for make span and total flow time criteria are presented. Details of the developed speed-up method for swap neighborhood under makespan objective and adaption of Li's speed-up method (Li, Wang, & Wu, 2009) are given in chapter 3 along with examples. In chapter 4, heuristic and metaheuristic methods that are used in this thesis are explained. Chapter 5 gives brief information about the design of experiments method. Chapter 6 gives the details of the algorithms that are developed in this thesis along with the details of the design of experiment approach used for tuning the algorithm parameters. In chapter 7, computational results of the proposed algorithms are given and compared to state of the art algorithms from the literature. Conclusions and future suggestions about the problems are presented in chapter 8.

2 PROBLEM DEFINITIONS AND LITERATURE REVIEW

2.1 SDST Permutation Flow shop Problem under MakeSpan Optimization Criteria

2.1.1 Problem Definition

SDST permutation flowshop scheduling problem under minimum makespan criterion, which is denoted as $F_m \Big| s_{jk}, prmu \Big| C_{max}$ (Pinedo, 2008) is shown to be NP - hard (Gupta & Darrow, 1986). It is assumed that the job order in each machine is same. So, this problem can be considered as a feasible subset set of the general flowshop shop problem in which job order does not have to be same on all machines. Objective of this problem is to find minimum completion time for all jobs, known as makespan or C_{max}. Minimizing the makespan leads to maximum machine utilization.

In SDST - PFSP, there are n jobs to be processed on m machines. All jobs have non-negative processing times on each machine which are denoted as p_{ij} $(i = 1, \dots, m, j = 1, \dots, n)$. All jobs has to visit all machines and a machine is able to handle at most one job at a time. Operations do not have priorities. Processing sequence of the jobs is the same for all machines and a sequence is represented by a permutation of jobs, i.e. $\pi = \{\pi_1, \dots, \pi_n\}$ where π_1 is the first job to be processed, and so on.

Each job consists of same number of operations. When a machine starts handling a job, no other job can interrupt the processing of the job. For all jobs and machines, ready times are required to be zero at start of processing. Sequence-dependent setup time is the machine preparation time for the next task to be handled and is denoted by S_{ijk} where i is the machine, j is the previous job that was processed on machine i and k is the current job to be handled. This setup time is required for setting up the processing environment for the next task in real life. Switching production environment or preparation cost for the next job will be different for different jobs in sequence dependent set up time problems. Cleaning a paint tank in which black paint was produced takes more time than cleaning a tank in which white color paint was produced if the next job is producing white color paint. C_{ij} denotes the completion time of job j on machine i. Total completion time or makespan is the finishing time of the last job on the last machine. For simplicity, total completion time is denoted as C_{mn} or C_{max}. Completion time of job j on machine i can be calculated as:

$$C_{ij} = \max\{C_{i,j-1} + S_{i,j-1,j}, C_{i-1,j}\} + p_{ij} \tag{1}$$

where $C_{0j} = C_{i0} = S_{i0j} = 0$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$.

2.1.2 Previous Works

In the literature, flow shop is one of the most studied scheduling problems since it was introduced by Johnson (1954). The literature on SDST flow shop scheduling problems has been extensively summarized in Allahverdi et al. (1999), Yang and Liao (1999), Cheng et al. (2000) and Potts and Kovalyov (2000). More recently Allahverdi et al. (2008) published a survey.

Even though exact algorithms are proposed in Corwin and Esogbue (1974), Rios-Mercado and Bard (1998a), Rios-Mercado and Bard (1999a), Rios-Mercado and Bard (2003), Tseng and Stafford (2001), Stafford and Tseng (2002), they are able to optimally solve problems up to 10 jobs and 6 machines or 9 jobs and 9 machines. For this reason, efforts have been devoted to heuristic and metaheuristic algorithms for larger problems that involve more jobs and machines.

Regarding heuristic algorithms, in Simons (1992), two general heuristics called TOTAL and SETUP have been developed. In Das et al. (1995), a heuristic based on a saving index is proposed. A well-known heuristic for the permutation flowshop scheduling problem without SDST is the NEH heuristic proposed by Nawaz et al. (1983). In Rios-Mercado and Bard (1998b), the NEH heuristic is modified in order to consider sequence dependent setup times and the new heuristic is called NEH_RMB.

In Rios-Mercado and Bard (1999b), some modifications are proposed to the heuristics in Simons (1992) and a new hybrid heuristic is developed and is called HYBRID.

As to metaheuristic approaches, in Rios-Mercado and Bard (1998b), a greedy randomized adaptive search procedure (GRASP) and another modification of the NEH algorithm called NEHT-RB was proposed. Ruiz et al. (Ruiz, Maroto, & Alcaraz, 2005) proposed genetic and memetic algorithms for the SDST flowshop scheduling problem under makespan criterion. Genetic algorithms and hybrid versions with new constructive population algorithms were tested. Performance of the proposed algorithms were compared to Osman and Potts' simulated annealing (Osman & Potts, 1989), Widmer and Hertz's tabu search, Rios-Mercado and Bard's GRASP.

Gajpal et al. (2006) proposed a new algorithm based on ant colony optimization. Artificial ants are used to initialize solutions and three different local search procedures are used to improve the initial solution. Result of the proposed algorithm is compared to SI (Das, Gupta, & Khumawala, 1995), GRASP (Rios-Mercado and Bard, 1998b) and MMAS (Stuetzle, 1998). The proposed algorithm showed better performance by reducing mean and relative percentage deviation.

Ruiz and Stützle (Ruiz & Stützle, 2007) proposed an iterated greedy algorithm (IG) with excellent results for the flowshop scheduling problem. In 2008, Ruiz and Stützle proposed another IG algorithm for PFSP with SDST (Ruiz & Stützle, 2008). A new test set was constructed by adding sequence dependent set up times with different distributions changing from 10 to 125 to Taillard's instances. Total weighted tardiness criterion was also considered in the same paper. They also extended the IG by adding a local search. The proposed IG_RS_{LS} algorithm has a simple structure and is easy to implement. IG_RS_{LS} was compared against 5 different algorithms including PACO, MA, IG_RS, GA and MA_{LS}. Statistically IG_RS_{LS} shows better results than the other tested algorithms with respect to ARPD for makespan minimization criterion.

In 2011, Mirabi proposed a new ant colony optimization technique for solving flow shop problem with sequence dependent setup times. The proposed local search algorithm was a combination of three techniques; forward insertion, backward insertion and pairwise interchange neighborhood. Results are compared to GA and HGA by Ruiz et al. (2005) and Tabu Searh (TS) algorithm by Eksioğlu (2008). In 2014, Mirabi published another paper that proposed a new hybrid genetic algorithm for PFSP with SDST problem.

R. Vanchipura and R. Sridharan (2013) proposed two constructive heuristics for PFSP with SDST that were named as setup ranking algorithm (SRA) and fictitious job setup ranking algorithm (FJSRA) and compared them to NEH_RMB. The proposed algorithms were based on ordering the jobs according to their setup times. NEH_RB order the jobs by their total processing times before sequencing and using them to construct partial schedules. Computational results showed that SRA algorithm did not show better performance than NEH_RB algorithm. However, FJSRA outperformed NEH_RB for smaller number of machines, but the performance of the proposed algorithm decreases for larger number of machines.

Li and Zhang (2012) developed three adaptive hybrid genetic algorithms and three local search methods are used in the proposed algorithms. These local search methods are based on hybrid neighborhood, insertion neighborhood and swap neighborhood. In addition, results were compared to IG_RS (Ruiz & Stützle, 2008). Proposed algorithms achieved varying performance for different distribution of setup times. AHA_1's performance decreased as the distribution range of the setup times increase. In contrast, AHA_3's performance increased as the distribution range of the setup times increase. No new best results were reported.

Victor Fernandez-Viagas and Jose M. Framinan (2014) proposed a new tie breaking mechanism for NEH and IG algorithms. NEH and IG algorithms were reported as notably efficient algorithms (Ruiz & Stützle, 2008) for flowshop problem with makespan optimization. The original proposed algorithm did not suggest any mechanism for solving ties in construction phase; so the first position which makes the makespan minimum is accepted as the insertion point of the job. Kalczynski and Kamburowski showed the importance of the tie-breaking mechanism for NEH heuristic (Kalczynski & Kamburowski, 2007). A new tie-breaking mechanism based on estimation of idle times was embedded into IG_RS_{LS}, NEH and IG_{RIS}. The results are compared to Dong's (Dong, Huang, & Chen, 2008) and Kalczynski & Kamburowski's (2007) tie-breaking mechanism. Performance of the proposed tie-breaking mechanism showed better performance.

Most recently, A. Allahverdi (2015) published a survey which puts together the recent studies that deal with sequence dependent setup times. This paper is the third survey for problems with sequence dependent set up times by the same author. Previous ones were in (Allahverdi, Gupta, & Aldowaisan, 1999) and (Allahverdi, Ng, Cheng, & Kovalyov, 2008).

2.2 SDST Permutation Flow shop Problem under Total Flowtime Criterion

2.2.1 Problem Definition

SDST permutation flowshop problem under total flow time minimization criterion is denoted as $F_m \big| \, s_{jk}, prmu \, \big| \sum C_i$ (Pinedo, 2008) and is proven to be $\mathcal{NP}$ -hard (Gray, Johnson, & Sethi, 1976) for the case without SDST. The typical flowshop structure is same with this version of the problem. Given jobs will be processed on each machine with the same order. Jobs cannot be interrupted during the process, i.e., there is no preemption. One job can be processed only on one machine and a machine can process only one job at a time. The objective of this problem is to find an optimum schedule which makes the total flow time minimum. The version of the problem studied in this thesis has sequence dependent set up times.

There are n jobs to be processed on m machines. All jobs have non-negative processing times on each machine, denoted as p_{ij} $(i = 1, \dots, m, j = 1, \dots, n)$. A solution (schedule) is represented by a permutation of jobs. i.e. $\pi = \{\pi_1, \dots, \pi_n\}$.

Each operation has sequence – dependent setup time shown as S_{ijk} where i is the machine, k is the next job that will be processed on machine i and j is the last job processed on machine i. C_{ij} denotes the completion time of job j on machine i. The goal is to find a sequence, which make the total flow time minimum. Completion times on each machine are calculated as follows:

$$C_{ij} = \max\{ C_{i,j-1} + S_{i,j-1} + C_{i-1,j}\} + p_{ij} \qquad (2)$$

where $C_{0j} = C_{i0} = S_{i0j} = 0$ for $i = 1, \dots, m$ and $j = 1, \dots, n$

Total flow time is calculated as follows:

$$TFT = C_{sum} = \sum_{j=1}^{n} C_{im} \qquad (3)$$

2.2.2 Previous works

Garey et al. (1979) proved that the mean flowtime (total flowtime) problem is NP-complete. Since then, many researchers developed heuristics for the problem. Some of the pioneers of these researchers are Gupta (1972) and Miyazaki et al. (1978).

Ho (1995) proposed a heuristic algorithm based on sorting, to minimize the total flow time. In this paper, Ho states that SPT (smallest processing time) rule gives better results for single machine problem. So, it is better to place jobs having smaller total processing time into early slots of the schedule. At the initial step of the heuristics, all jobs are sequenced in ascending order and an index is calculated and assigned to each job. In the second phase of the heuristic, indexes assigned to the jobs are used to obtain optimal schedules by using exchange sort and bubble sort. The heuristic shows a good performance for large instances.

Wang et al. (1997)'s heuristics are also based on indexing and sorting of these indexes. In their first heuristic, researchers tried to keep idle times small. The idea is; if completion time of the current machine to be scheduled is smaller than the arrival time of the scheduled job, then an idle time will occur on this machine. Idle times lead to delays in job completion times and also increase total flow time. In the first heuristic to keep idle times low, the job with earliest starting time is chosen to be scheduled. This heuristic is named as LIT (less idle times). Second heuristic aims to minimize machine idle time and job queuing times. Two metrics were calculated using Euclidean distance and linear distance. According to calculated values, an index is assigned to each job. By choosing the smallest distance, an optimal schedule was constructed. This heuristic is named as smallest process distance (SPD) rule. Their computational experiments showed that the results of the proposed methods are very close to optimal values.

In 1997, C. Rajendran and H. Ziegler proposed an algorithm for minimizing the total weighted flowtime for flowshop problems. They stated that, the objective of minimizing the total weighted flowtime of jobs is the same as minimizing the mean weighted flow time of jobs objective. The proposed algorithm has two steps. A seed sequence is constructed according to shortest weighted total processing time in the first step. Second step is the improvement step. First job is taken from the sorted job list and it is inserted to all possible slots in a fashion similar to NEH. After placing the first job, next job is taken from the list and inserted to all possible slots and it is inserted to position where the total flow time is minimum. In literature, this algorithm is called the RZ (Rajendran & Ziegler, 1997) algorithm.

In 1998, Woo and Yim developed an algorithm for total flow time minimization problem. The algorithm starts with calculating the total flow time of the jobs and puts jobs into the sequence which has the minimum flow time value. The algorithm then picks a job from the unscheduled job list and inserts it to all possible slots. This process continues until all jobs are scheduled. Framinan and Leisten (2003) stated that, RZ algorithm outperforms WY for small instances. However, WY algorithm outperforms RZ algorithm for larger instances.

J. Liu and C. Reeves's algorithm (Liu & Reeves, 2001), which is named as LR algorithm was developed in 2001. This heuristic proposed a new initial solution construction method for total flow time minimization problem. The constructive algorithm is combined with a local search procedure. Construction of an initial solution consists of two phases. In the first phase, the weighted total machine idle time index is calculated for all unscheduled jobs. In the second phase, artificial total flow time is calculated for these jobs and an index value is assigned to them. Finally, calculated index from the first phase and second phase is added in order to find the final index values of the jobs. These calculations are repeated for all unsorted jobs. The job with the lowest index value is added to the schedule. In the local search procedure, pairwise exchanges are applied to the schedule that is obtained in the constructive phase.

J.M.Framinan et al. (2002) proposed an improved version of the NEH algorithm which extends the current algorithm for total flow time objective. Their work focus on both makespan and flow time minimization. The proposed algorithm basically changes the first phase of the NEH algorithm. In the original NEH algorithm, jobs are ordered with respect to descending sum of their total processing times on machines, while the new algorithm changes this ordering to ascending sum of total processing times on machines. Computational results show that the algorithm gives superior results for multi objective version of the problem.

Allahverdi and Aldowaisan (2002) proposed seven heuristics for this problem. These seven heuristic were developed by combining previously proposed heuristics such as WY (Woo & Yim, 1998), NEH (Nawaz, Enscore, & Ham, 1983) and RZ (Rajendran & Ziegler, 1997). The results obtained by the proposed algorithms were compared to results of earlier heuristics. One of the proposed algorithms, IH6, which is the combination of the RZ and WY algorithm, gave superior results with respect to the compared algorithms. Adding a local search procedure (pair-wise exchange) for NEH, WY, IH2, RZ and IH6 yielded significant improvements. They claimed that poor performance of RZ algorithm in large instances became better with their proposed new algorithms.

Framinan and Leisten (2003) proposed a heuristic based on NEH algorithm for total flowtime minimization objective. In this paper, Framinan and Leisten recalculated that complexity of WY algorithm (Woo & Yim, 1998), claiming that it was miscalculated. Woo & Yim (1998) calculated the complexity of their algorithm as $O(n^3)$. Framinan and Leisten (2003) calculated the complexity as $O(n^4)$. In WY algorithm, after sequencing the jobs in ascending order with respect to their total process time, a job is added to the partial sequence and to find its final position, the inserted job is pair-wised exchanged with the other jobs which are already sequenced. The new algorithm proposed by Framinan and Leisten (2003) only differs from the WY algorithm in insertion procedure of the newly inserted job to the partial sequence. In Framinan and Leisten's algorithm, when a new job is added to partial sequence, the position of the new job is determined by interchanging the all possible sequences for partial schedule. This constructive step also increases the complexity of the NEH algorithm from $O(n^3)$ to $O(n^4)$**.** Computational results of that research show that, the proposed new algorithm outperforms WY (Woo & Yim, 1998) and RZ (Rajendran & Ziegler, 1997) algorithms. Besides FL construction heuristic, a new algorithm was proposed in this paper. The new algorithm is named as "IH7-proposed". The new algorithm is inspired from IH7 (Allahverdi & Aldowaisan, 2002). IH7-proposed gave better results than IH7.

Li et al (2009) proposed an algorithm which speeds up the calculation of total flow time. General Flowtime Computing (GFC) algorithm presented in this paper basically divides the sequence into two parts as changed and unchanged part. For next calculation steps, only the jobs which are in the changed part of the sequence are considered in computations. The proposed algorithm does not reduce the complexity of the procedure, but it reduces the CPU time required for calculations. A minimum of 33.3% of CPU time is saved with the new calculation method. The speed-up technique is applied to LR (Framinan & Leisten, 2003) and IH7 (Allahverdi & Aldowaisan, 2002) heuristics and the proposed algorithm achieved better results. In this study, this acceleration technique (Li et al. 2009) is adapted and used to speed-up the calculation of total flow time.

In D. Laha and S.C. Sarin (2009)'s paper, a modified FL (Framinan & Leisten, 2003) heuristic was proposed. The proposed algorithm modified the iterative step of adding a new job to the partial schedule. The modified algorithm did not change computational complexity of FL algorithm. The modified algorithm outperformed the previous methods for large and small size problems.

Quan-Ke Pan and Ruben Ruiz's paper (2013) has a compressive review of the flow shop problems and proposed heuristics. Authors implemented the major heuristics and evaluated their computational results with respect to relative percentage increase (RPI) performance criterion and CPU time in detail. Five new algorithms were proposed and compared against existing heuristics. The proposed algorithms were mostly combinations of the previously proposed heuristics. A total number of 22 heuristics were implemented and their results are compared statistically. A simple heuristic (Laha and Sarin, 2009) has a good performance. As a composite heuristic, authors proposed an algorithm called LR_NEH(x) which represents a better trade-off in CPU time and quality than other heuristics.

V. Fernandez and J. Framinan (2015) recently proposed a new heuristic based on Liu and Reeves heuristic (LR). The newly proposed constructive heuristic is reported to decrease the computational complexity by one. The new algorithm has better results in terms of ARPD and CPU time than LR algorithm.

3 SPEED-UP METHODS

In this chapter, the speed-up methods developed and adapted for the SDST-PFSP under makespan and total flow time minimization criteria are explained in detail. The novel speed-up method for the swap neighbourhood is given in section 3.1. The details of the how the speed-up methods for the total flow time calculations without sequence-dependent setup times are adapted to sequence dependent setup times version of the problem are explained in section 3.2.

3.1 Speed-up Methods for Makespan Calculation

Nawaz et al. (1983)'s heuristic, known as the NEH heuristic, is recognized to be the best performing heuristic for the regular PFSP under makespan criterion (Ruiz & Stützle, 2007). In the NEH heuristic, jobs are arranged by a descending order of their total processing times on machines and the first two jobs are considered for insertion into an empty permutation in order to minimize the partial makespan. Then, the remaining jobs are inserted into each available position in the partial solution and the position that minimizes the partial makespan is selected as the insertion position. All jobs are considered in order such that each job is inserted in the position with a minimum partial makespan. Time complexity of the NEH algorithm is $O(n^3m)$. However, Taillard (1990) proposed a well-known speed-up method for the NEH heuristic, which reduces the time complexity of the NEH algorithm from $O(n^3m)$ to $O(n^2m)$.

The details of the Taillard speedup method for the NEH heuristic for the permutation flowshop scheduling problem (PFSP) without sequence-dependent setup times under the makespan minimization criterion is well described in Taillard (1990) and Fernandez-Viagas and Framinan (2014). The speed-up method of Taillard for insertion of a job into a position in the partial permutation can be adapted to the SDST permutation flowshop scheduling problem using a notation similar to Fernandez-Viagas and Framinan (2014) as follows:

Assume that a partial schedule of $k-1$ jobs has been established and an unscheduled job r with processing times p_{ir} will be inserted in position $l\ (l = 1,\dots,k)$. The earliest completion time of j^{th} job on i^{th} machine before inserting the unscheduled job can be calculated as follows:

$$e_{ij} = \max\{ e_{i,j-1} + S_{i,j-1,j}, e_{i-1,j}\} + p_{ij} \qquad i = \{1,\dots,m\}; j = \{1,\dots,k-1\} \qquad (4)$$

where $e_{0j} = e_{i0} = S_{i,0,1} = 0$ (i.e., the starting time of the first job on the first machine is 0 and the setup time for the first job is 0 on all machines).

The duration between the starting time of the j^{th} job on the i^{th} machine and the end of all operations (also known as the tail) before insertion can be calculated as follows:

$$q_{ij} = \max\{q_{i+1,j}, q_{i,j+1} + S_{i,j-1,j}\} + p_{ij} \qquad i = \{m,\dots,1\}; j = \{n,\dots,k+1\} \quad (5)$$

where $q_{m+1,j} = q_{i,k} = 0$

The earliest relative completion time $f_{i,j}$, which is the completion time of job r on machine i that will be inserted into position l can be calculated as follows:

$$f_{il} = \max\{ f_{i-1,l}, e_{i,l-1} + S_{i,l-1,l}\} + p_{i,r} \qquad i = \{1,\dots,m\} \quad (6)$$

where $f_{0l} = 0$

The makespan value of the new permutation after inserting job r to position l can be calculated as follows:

$$C_{max}(l) = \max_{i=1,\dots,m}\{ f_{il} + q_{il} + S_{i,l,l+1}\} \qquad (7)$$

The above procedure for the insertion neighborhood reduces the computational complexity of calculating the makespan by using the Eq. (1) from $O(n^3m)$ to (n^2m) .

It is possible to extend the above speed-up method to swap neighborhood, which follows:

Suppose that two jobs in positions s and t will be exchanged. In order to calculate the new makespan value, first calculate the earliest completion time e_{ij} of j^{th} job on i^{th} machine before the first swapping position as:

$$e_{ij} = \max\{ e_{i,j-1} + S_{i,j-1,j}, e_{i-1,j}\} + p_{ij} \qquad i = \{1, \dots, m\},\ j = \{1, \dots, s-1\} \quad (8)$$

where $e_{0j} = e_{i0} = S_{i,0,1} = 0$ (i.e., the starting time of the first job on the first machine is 0 and the setup time for the first job is 0 on all machines).

Before swapping two jobs, calculate the tail q_{ij}, which is the duration between the starting time of job j on machine i and the end of the operations:

$$q_{ij} = \max\{ q_{i+1,j}, q_{i,j+1} + S_{i,j,j+1}\} + p_{ij} \qquad i = \{m, \dots, 1\},\ j = \{ n, \dots, t+1\} \quad (9)$$

where $q_{m+1,j} = 0, q_{i,k} = 0.$

Then, calculate the earliest relative computation times of the jobs starting from s prior to position t (the changed part of the permutation after exchanging the jobs in positions s and t):

$$f_{ij} = \max\{ f_{i,j-1} + S_{i,j-1,j}, e_{i-1,j} \} + p_{ij} \qquad i = \{ 1, \dots, m\},\ j = \{s, \dots, t\} \quad (10)$$

where $f_{i,s-1} = e_{i,s-1}$

Finally, the new makespan value after exchange of jobs in positions s and t can be calculated as:

$$C_{max} = \max_{i=1,\dots,m}\{ f_{it} + q_{i,t+1} + S_{i,t,t+1}\} \qquad (11)$$

In order to explain the proposed new speed-up method more clearly, an example with an 8 job-2 machine instance is given in Figure 3-1, where job 3 and job 5 will be interchanged in an identity permutation. Assume that, the size of jobs between two swap positions is denoted by δ. First, the earliest completion times e_{12} and e_{22} is calculated. Then, q_{16} and q_{26} can be easily calculated up to position $t+1$. It is clear that, after swapping jobs at positions 3 and 5, the earliest completion times of all jobs prior to 3 will not be changing. Note that the third step is provided in Figure 3-1 in order to explain how f_{ij} values are updated in the fourth step. Briefly, the completion times δ_{15} and δ_{25} from position 3 to position 5 should be re-calculated in the fourth step by using Eq. (8). This step, which is not present in Taillard's speed-up for the insertion neighborhood, is required for the swap neighborhood since the completion times of the jobs within the positions s and t will be changed. In the fourth step, since we have e_{12} and e_{22} already calculated before, f_{13} and f_{23} can be calculated starting from position 3 up to position 5. Finally, the makespan value after interchanging job 3 with job 5 can be obtained by taking the maximum of additions as follows:

$$C_{max} = \max\{f_{13} + q_{16} + S_{1,3,6}\ , f_{23} + q_{26} + S_{2,3,6}\ \}.$$

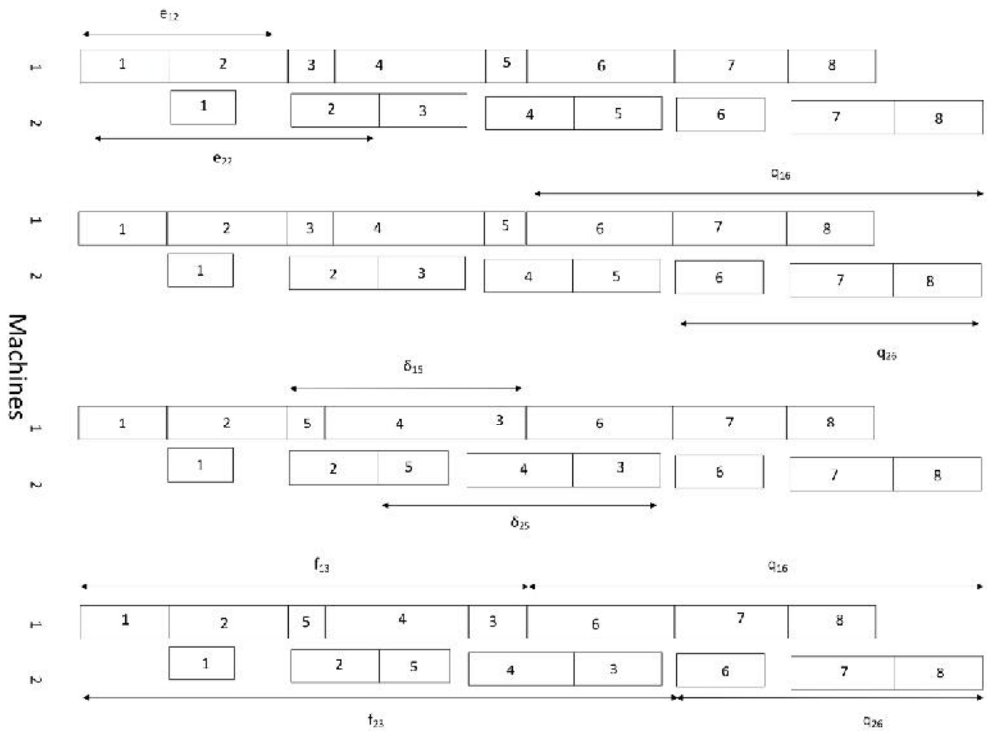

Figure 3-1 : Swapping job 3 with job 5

As known, the size of interchange neighborhood structure is $n(n-1)/2$. Since each objective function evaluation takes $O(nm)$ time, the computational complexity of interchange neighborhood structure is $O(n^3m)$. The proposed speed-up method can provide 53% decrease in CPU time in average as shown experimentally in Table 7-1. However, it should be noted that the proposed speed-up method cannot decrease the time complexity of swap neighborhood structure from $O(n^3m)$ to $O(n^2m)$.

A numerical example for swap speed-up procedure is presented below. SDST_TA001 instance is used in this example. For simplicity, only the first 8 jobs with 5 machines are considered. The processing times and setup times matrices for each machine are given in Table 3-1 to 3-6.

n/m	1	2	3	4	5
1	54	79	16	66	58
2	83	3	89	58	56
3	15	11	49	31	20
4	71	99	15	68	85
5	77	56	89	78	53
6	36	70	45	91	35
7	53	99	60	13	53
8	38	60	23	59	41

Table 3-1. The processing times matrix

$S_{1,j,k}$	1	2	3	4	5	6	7	8
1	0	27	41	8	36	39	18	27
2	47	0	39	2	6	49	28	35
3	2	38	0	8	44	24	8	44
4	37	38	20	0	33	29	15	34
5	27	38	25	15	0	29	28	10
6	44	29	9	34	14	0	11	4
7	43	36	23	48	4	43	0	34
8	31	21	18	22	4	42	4	0

Table 3-2. The setup times matrix for machine 1

$S_{2,j,k}$	1	2	3	4	5	6	7	8
1	0	11	41	49	23	28	8	49
2	34	0	26	14	37	49	4	42
3	17	7	0	10	34	13	47	6
4	45	35	9	0	28	19	2	19
5	41	14	25	8	0	17	25	21
6	11	38	27	28	45	0	37	41
7	30	40	19	21	2	9	0	13
8	20	19	27	17	29	22	25	0

Table 3-3. The setup times matrix for machine 2

$S_{3,j,k}$	1	2	3	4	5	6	7	8
1	0	30	43	8	26	14	39	3
2	32	0	40	6	36	17	32	9
3	25	35	0	33	10	48	26	41
4	14	16	30	0	42	22	9	45
5	15	22	27	13	0	19	47	18
6	32	6	28	1	6	0	17	1
7	16	31	14	46	13	2	0	8
8	34	2	3	49	37	24	41	0

Table 3-4. The setup times matrix for machine 3

$S_{4,j,k}$	1	2	3	4	5	6	7	8
1	0	37	3	42	3	22	32	34
2	28	0	5	25	44	31	46	12
3	32	19	0	34	19	3	37	37
4	1	20	27	0	40	17	22	49
5	17	6	13	45	0	23	42	42
6	40	8	40	11	27	0	1	36
7	4	15	6	17	14	15	0	17
8	46	43	7	11	24	20	31	0

Table 3-5. The setup times matrix for machine 4

$S_{5,j,k}$	1	2	3	4	5	6	7	8
1	0	23	12	21	18	45	1	7
2	13	0	40	17	1	12	2	6
3	2	29	0	5	17	47	10	4
4	34	35	22	0	32	15	17	17
5	32	33	17	30	0	44	23	32
6	28	39	46	3	46	0	49	15
7	39	47	1	48	21	16	0	44
8	2	33	9	25	45	16	20	0

Table 3-6. The setup times matrix for machine 5

Suppose that, the current permutation is $= \{6, 3, 4, 2, 8, 5, 1, 7\}$, and jobs 4 and 5 are to be interchanged to obtain the new permutation $\pi^* = \{6, 3, 5, 2, 8, 4, 1, 7\}$.

In order to calculate the new makespan, first e_{ij} and q_{ij} matrices are constructed for permutation π which are given in Table 3-7 and 3-8.

e_{ij}	1	2	3	4	5	6	7	8
1	36	60	-	-	-	-	-	-
2	106	144	-	-	-	-	-	-
3	151	228	-	-	-	-	-	-
4	242	313	-	-	-	-	-	-
5	277	343	-	-	-	-	-	-

Table 3-7. e_{ij} matrix for permutation π

q_{ij}	1	2	3	4	5	6	7	8
1	-	-	-	-	-	-	366	278
2	-	-	-	-	-	-	312	225
3	-	-	-	-	-	-	194	126
4	-	-	-	-	-	-	178	66
5	-	-	-	-	-	-	112	53

Table 3-8. q_{ij} matrix for permutation π

Since $\pi_2^* = 3$, $\pi_3^* = 5$, $e_{03} = 0$ and $f_{12} = e_{1,2} = 60$.

Then, f_{13} can be calculated as:

$$f_{13} = \max\{ f_{12} + S_{1,3,5} , e_{0,3}\} + p_{1,5} = \max\ \{ 60 + 44 , 0 \} + 77 = 181.$$

The remaining f_{ij} matrix can be calculated similarly by using equation 8. The f_{ij} matrix for π^* is shown in Table 3-9.

f_{ij}	1	2	3	4	5	6	7	8
1	-	-	181	302	375	468	-	-
2	-	-	237	305	435	567	-	-
3	-	-	327	438	470	582	-	-
4	-	-	410	496	567	650	-	-
5	-	-	463	552	608	735	-	-

Table 3-9. f_{ij} matrix for permutation π^*

Now, the makespan on each machine can be calculated by using Eq. (9) and the results are given in Table 3-10.

i	$f_{1,6}$	$q_{1,7}$	$S_{i,4,1}$	$C_{max}(\pi^*)$
1	468	366	37	871
2	567	312	45	**924**
3	582	194	14	790
4	650	178	1	829
5	735	112	34	881

Table 3-10. Calculation of final makespan value

Finally, the new make span value for π^* is the maximum of $C_{max}(\pi^*)$ values, which is 924.

3.2 Speed-up Methods for Total Flow Time Calculation

Li et al. (2009) proposed General Flow time Computing (GFC) speed-up method for total flowtime calculation. In order to calculate the objective function, finishing times of each job on the last machine must be calculated. While applying the search operations like swap and insertion, new permutations carry similar sub sequences from their parent permutations. As an outcome of this fact, the proposed GFC suggests dividing the resulting schedule into changed and unchanged parts. Partial fitness values for the unchanged part of the permutation values do not have to be calculated again; they can be used directly in further calculations.

Total flow time calculation starts from the first job of the permutation. In each perturbation, all completion times of the jobs are calculated for the new permutation using Eq. (2). If completion times of all jobs on each machine are calculated initially, the new fitness value after a perturbation is applied to current schedule can be calculated in a quicker way. Consider the example given in Figure 3.2 where job 5 is removed from the permutation and inserted after job 3. Completion times of the first three jobs remain unchanged; there is no need to calculate them again.

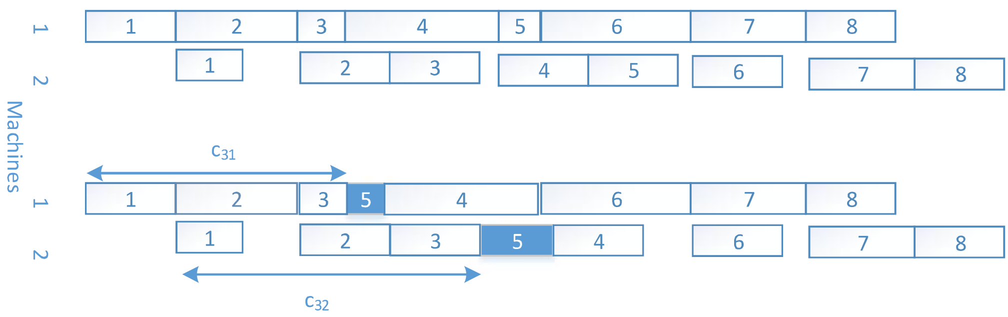

Figure 3-2 Total Flowtime Insertion operation of job 5

The total flow time calculation procedure can be modified as follows in order to speed up the fitness computation:

The fifth job is inserted between third and fourth jobs (Figure 3-2). Finishing times up to third job are already available from the previous calculations and they are stored in $\boldsymbol{e_{ij}}$ matrix. Completion times of the new sequence can be calculated as:

$c_{ij} = \max\{e_{i,j-1} + S_{i,j-1,j}, e_{i-1,j}\} + p_{ij}$ where $i = \{1,..,m\}$ and $j = \{4, \ldots, n\}$, i.e., fitness calculation starts from position 4 instead of position 1.

Finally, the total flowtime will be calculated using Eq. (2) using new completion times. As a result, completion times of the unchanged part are not calculated again, the number of calculation steps is decreased.

Li et al. (2009) showed that the fitness calculation times can be reduced up to 50%. Li's speed-up algorithm can be adapted for both swap and insertion neighborhoods. However, the computational time complexity remains the same.

An example for swap neighborhood is presented below. SDST_TA001 instance is used in this example. For simplicity, only the first 8 jobs with 5 machines are considered. This is the same problem instance which is used for SDST PFSP under C_{max} optimization criterion example. Suppose that, the current permutation is $\pi = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and the job 4 and job 7 are to be interchanged as shown in Figure 3-3, to obtain new permutation $\pi^* = \{1, 2, 3, 7, 5, 6, 4, 8\}$.

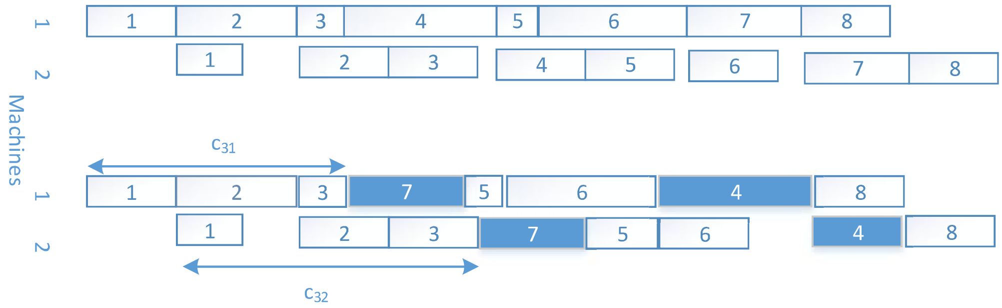

Figure 3-3 Total Flowtime swap of job 4 and job 7

In order to calculate the new total flowtime, previously calculated machine finishing time (e_{ij}) matrix for permutation π shown in Table 3-11 will be used.

e_{ij}	1	2	3	4	5	6	7	8
1	54	164	218	297	407	472	536	608
2	133	167	229	396	480	567	703	776
3	149	268	357	411	569	633	763	799
4	215	326	388	490	647	761	776	858
5	273	382	442	575	700	796	898	983

Table 3-11 Completion times matrix for π

There is no need to recalculate the finishing times of the first three jobs in order to calculate the new fitness permutation for π^*. So, they will simply be reused as shown in Table 3-12.

e_{ij}	1	2	3	4	5	6	7	8
1	54	164	218	-	-	-	-	-
2	133	167	229	-	-	-	-	-
3	149	268	357	-	-	-	-	-
4	215	326	388	-	-	-	-	-
5	273	382	442	-	-	-	-	-

Table 3-12 Completion times matrix for first three jobs

New completion times for the jobs after the third position will be calculated using Eq. (3), starting from position 4.

$c_{4,1} = \max(\ 218 + 8\ ,0\ ) + 53 = \ \ 279$

$c_{4,2} = \max(\ 229 + 47\ ,279\ ) + 99 = 378$

$c_{4,3} = \max(\ 357 + 26\ ,378\ ) + 60 = 443$

$c_{4,4} = \max(\ 388 + 37\ ,443\ ) + 13 = 456$

$c_{4,5} = \max(\ 442 + 10\ ,456\ ) + 53 = \ \ 509$

Jobs in the remaining part of the permutation can calculated similarly and total flow time will be calculated using these new values. The results are shown in Table 3-13.

f_{ij}	1	2	3	4	5	6	7	8
1	54	164	218	279	360	425	530	602
2	133	167	229	378	436	523	650	729
3	149	268	357	443	545	609	665	752
4	215	326	388	456	623	737	816	924
5	273	382	442	509	676	772	901	965

Table 3-13 The new finishing times matrix

Total flow time can be calculated by using equation 2 as follows

$$TFT(\pi^*) = \ \sum_{j=1}^{5} C_{8,j} = (602 + 729 + 752 + 924 + 965) = \ 3972.$$

The new fitness value is calculated using 25 operations instead of 40 operations, hence saving 15 calculations, resulting in 15/40 = 37.5% less CPU time.

4 HEURISTIC AND METAHEURISTIC ALGORITHMS USED IN THESIS

Optimization problems can be divided in two classes. A solution can be developed in polynomial time for the first class of problems. So, an optimum solution can be found efficiently. On the other hand, for the second class of optimization problems, finding an optimum solution in polynomial time is considered impossible and these kinds of problems are named as NP (nondeterministic polynomial time) problems. For these types of problems, brute force or exhaustive search can be infeasible for even moderate problem sizes. Thus, alternative algorithms are needed for finding optimum or near optimum solutions for problems in reasonable amount of time. Stochastic optimization is general category of algorithms and techniques which use some level of randomness to achieve optimal solutions (Luke, 2015).

Heuristics are alternative ways to find solutions for hard problems. Some problems can be solved by using heuristic algorithms that are tailored for the problem in hand. Heuristic approaches do not guarantee to find optimum solutions but they can generally find near optimal solutions in reasonable time. However, it may not be straightforward to develop a heuristic for a given problem. Besides not guaranteeing the optimal solution, heuristics have more tradeoffs such as incompleteness, loss in accuracy and precision and long execution times.

Other important tradeoff of heuristics approaches is being problem specific. It means that, a good heuristics for a given problem cannot be applied to the other problems in most cases. At this point, metaheuristic approaches can offer more generic solution methods for different kinds of problems. Metaheuristic is a term for representing an extensive subfield of stochastic optimization (Korst, Aarts, & Michiels, 2005). Metaheuristics are derived by abstracting the heuristic methods for different problems (Johnson C. G., 2008). Another way to come up with new metaheuristics is observations from nature. Some metaheuristics, such as Genetic

Algorithms (Mitchell, 1996) and Ant Colony Optimization (Dorigo & Gambardella, 1997) are inspired from nature.

Metaheuristic methods become very useful when they are hybridized them with local search methods (Osman & Laporte, 1996). Neighborhood search algorithms can be a good option for optimization problems for finding local optimums. The current best value can be taken as initial starting point for the local search algorithm. In each iteration, neighborhood of the current solution is searched. If a neighboring solution having better fitness value is found, the current best solution is replaced with the neighbor. This process can continue until a local optimum value is found. This operation is called as local search. This neighborhood search continues until some finishing criteria is met or a local optimum is reached. However, this local optimum value can be far away from the optimum value as stated by Osman and Laporte (1996). Other techniques, such as picking a good starting point (solution), using learning systems like tabu search or an adaptive acceptance criterion as in simulated annealing may be needed in order to get better results.

A hypothetical state space landscape is demonstrated in Figure 4-1 (Russell & Norvig, 2010). In the figure, current state represents the current best solution and elevation is value of the objective function. Objective function can aim to find the highest peak (global maximum) or lowest valley (global minimum). Local search algorithms aim to search through the solution space to find a local optimum value.

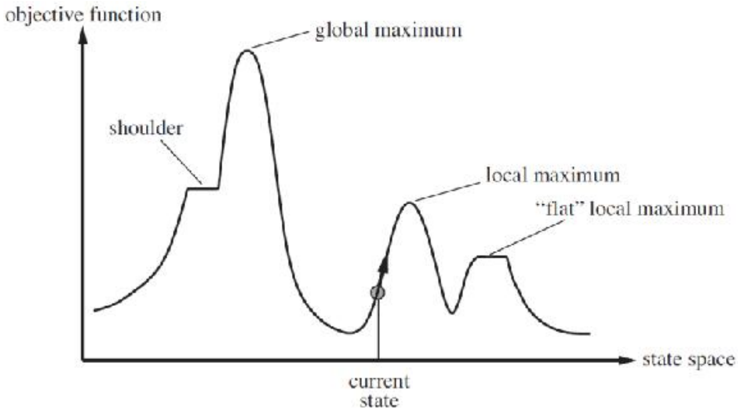

Figure 4-1 A one-dimensional state-space landscape in which elevation corresponds to the objective function (Russell & Norvig, 2010)

Some algorithms are greedy and this may cause them to become stuck in local minimum or maximum. A pure random move can be useful to avoid such circumstances, but this may be very expensive and sometimes it may not be useful. Various techniques to avoid being stuck in local hills and valleys are proposed. One of such techniques is used in the simulated annealing algorithm, which has been applied to many problems successfully. Annealing is the process of heating up metals to high temperatures, and then leaving them to cooling for a while and heating them up again. While increasing the temperature of the metal for a small period of time, the process allows metal to move in opposite direction. Stuart and Norvig (2010) gave a ball example to explain the simulated annealing approach. If a ball is thrown to the search space, ball will get stuck in the first local optimum. If ground (search space) is shaken, ball can move out of its current state and advance to the next valley. In most simulated annealing implementations, the size of the shake is bigger at the beginning of the search. The size of the shake decreases gradually as the algorithm runs.

4.1 NEH Algorithm

NEH (Nawaz, Enscore, & Ham, 1983) is the best known and most efficient heuristic for flowshop scheduling problem. NEH algorithm aims to insert new jobs into best position in the permutation of partially scheduled jobs. The best position is the position that results in minimum partial fitness function value. The worst-case complexity of the algorithm is $O(n^3m)$. In 1990, Taillard (Taillard, 1990) proposed a new calculation method for makespan calculation in flowshop problems that decreases the computational complexity of the algorithm from $O(n^3m)$ to $O(n^2m)$. NEH algorithm became more efficient and popular in flowshop problems with this improvement for makespan minimization criterion.

Basic NEH algorithm consists of two stages. In the first stage, jobs are sorted according to their total processing times on all machines. In the second stage, the jobs are considered one by one in order for insertion into the partial schedule. The jobs are inserted in all possible positions in the partial schedule and the position that minimizes the partial fitness value is selected for insertion. The pseudo-code of the NEH algorithm is given in Figure 4-2.

$Step\ 1:\ Order\ the\ jobs\ according\ to\ their\ total\ processing\ times\ wrt\ LPT$
$Step\ 2:\ Take\ the\ first\ two\ jobs, schedule\ them\ to\ minimize\ the\ makespan$
$\quad and\ remove\ them\ from\ the\ list$
$Step\ 3:\ Repeat\ the\ following\ while\ the\ job\ list\ is\ not\ empty$
$\quad Step\ 3.1:\ Take\ the\ next\ job\ from\ the\ list$
$\quad Step\ 3.2:\ Insert\ the\ job\ in\ all\ possible\ slots\ and\ calculate\ partial\ makespan$
$\quad Step\ 3.3:\ Fix\ the\ job\ to\ the\ positon\ which\ makes\ the\ partial\ makespan\ minimum$

Figure 4-2 Pseudocode of the NEH algorithm

Taillard's speedup (Taillard, 1990) constructs earliest completion times (e_{ij}) and tail (q_{ij}) matrices before executing Step 3.2 in order to calculate partial makespan values in one operation instead of using $n \cdot m$ operations.

Adaptations and modifications to NEH algorithm has been proposed for different versions of the flowshop problem. For example, Rios-Mercado and Bard (Rios-Mercado & Bard , 1998b) extended the algorithm to consider sequence dependent setup times (SDST) in the calculation and named their algorithm as NEHT-RB (Nawaz-Enscore-Ham, Taillard, Rios-Mercado and Bard). e_{ij} and q_{ij} matrixes are calculated by applying the Taillard's speed up again before inserting the new job in all positions in the partial sequence and by considering SDST as follows:

$$e_{i0} = 0\ ,\ \ e_{oj} = 0$$
$$e_{ij} = \max\{\ e_{i-1,j}\ ,\ e_{i,j-1} + S_{i,j-1,j}\} + p_{ij}$$

and

$$q_{ik} = 0, q_{m+1,j} = 0$$
$$q_{ij} = \max\{q_{i+1,j}\ ,\ q_{i,j+1} + S_{i,j,j+1}\} + p_{ij}$$

Computational complexity of NEHT-RB is calculated as $O(mn^2)$ (Rios-Mercado & Bard, 1998b).

4.2 Iterated Greedy (IG) Algorithm

Ruiz and Stützle (2007) presented Iterated Greedy (IG) algorithm for permutation flowshop scheduling problem and later extended the algorithm for the same problem to include sequence dependent set up times (Ruiz & Stützle, 2008). IG can produce good solutions in a short time limit. IG algorithm consists of two phases: destruction and construction. In the first phase (destruction phase), a previously determined number of elements are removed from the current incumbent solution. After this phase, construction phase is started. In the construction phase, the elements (jobs) that were removed in the previous phase are inserted into the solution again, by considering them in the order they were removed. The position that minimizes the partial makespan for the current considered job is selected. After all removed jobs are reinserted and a new full solution is generated, the new solution is considered for acceptance to be used in the next iteration. As its name suggests, IG algorithm is greedy; the new solution is always accepted if it is better than the current solution. However, since greedy algorithms can get stuck at local minima, a Metropolis type acceptance criterion (Metropolis, Rosenbluth, Rosenbluth, & Teller, 1953) similar to simulated annealing can be used. IG algorithm continues to iterate until a stopping criteria such as time limit or predetermined number of iterations, is met. IG algorithm has been successfully applied to many other problems, such as set covering problem (Jacobs & Brusco, 1995) and airline crew scheduling (Marchiori & Steenbeek, 2000). The pseudo code of the IG algorithm is given in Figure 4-3.

$\boldsymbol{Procedure}\ IG(\pi, k)$

$\pi = NEH$

$\pi = IterativeImprovementInsertion(\pi)$

$\pi_{best} = \pi$

$\boldsymbol{while}(\ termination\ criterion\ is\ not\ satisfied)\ \boldsymbol{do}$

$\pi' \coloneqq \pi;$

$\boldsymbol{for}\ i \coloneqq\ 1\ \boldsymbol{to}\ k\ \boldsymbol{do}$

$\pi' \coloneqq remove\ a\ job\ at\ random\ from\ \pi' and\ insert\ it\ into\ \pi'_R$

$\boldsymbol{endfor}$

$\boldsymbol{for}\ i \coloneqq 1\ \boldsymbol{to}\ k\ \boldsymbol{do}$

$\pi' \coloneqq best\ permutation\ obtained\ by\ inserting\ \pi'_{R_i}\ in\ all\ positions\ in\ \pi'$

$\boldsymbol{endfor}$

$\pi'' = IterativeImprovementInsertion(\pi')$

$\boldsymbol{if}\ C_{max}\ (\pi'') < C_{max}\ (\pi)$

$\pi \coloneqq \pi''$

$\boldsymbol{if}\ C_{max}\ (\pi) < C_{max}\ (\pi_{best})$

$\pi_{best} = \pi$

$\boldsymbol{endif}$

$\boldsymbol{else\ if}\ random() < \exp\{-(C_{max}\ (\pi'') - C_{max}\ (\pi))/Temperature$

$\pi \coloneqq \pi''$

$\boldsymbol{endif}$

$\boldsymbol{endwhile}$

$\boldsymbol{end\ Procedure}$

Figure 4-3 Iterated Greedy Algorithm pseudo code

Initial solution for IG is obtained by using the NEH algorithm. In destruction phase of IG, k randomly selected jobs are removed from π. k is a very important performance parameter for IG. Selection of k jobs is totally random and without repetition. After the destruction phase, $n - k$ jobs will be remained in the partial solution and k jobs will be in removed jobs sequence π'_R. The construction phase is the process of inserting the removed jobs into the partial schedule until all of the removed jobs are placed in the partial job sequence in order to obtain a full solution again.

A local search step can be added to IG to improve its performance. Ruiz and Stützle (2007) added a simple and effective local search algorithm to the proposed IG algorithm which is based on insertion neighborhood. The insertion operation is applied to all jobs in the current solution. In the local search method, named as iterative improvement insertion, one job is selected from π' without repetition and it is inserted into all possible positions in the permutation. The algorithm uses first improvement pivoting rule. The pseudo code of the local search method is shown in Figure 4-4 (Ruiz & Stützle, 2007).

$\boldsymbol{procedure}$ $IterativeImprovementInsertion(\pi)$
$improve := true;$
$\boldsymbol{while}(improve = true)$ $\boldsymbol{do}$
$improve := false$
$\boldsymbol{for}$ $i = 1$ $\boldsymbol{to}$ n $\boldsymbol{do}$
$remove\ a\ job\ k\ at\ random\ from\ \pi\ (without\ repetition)$
$\pi' := best\ permutation\ obtained\ by\ inserting\ k\ in\ all\ positions\ of\ \pi$
$\boldsymbol{if}$ $C_{max}(\pi') < C_{max}(\pi)$ $\boldsymbol{then}$
$\pi := \pi'$
$improve := true$
$\boldsymbol{endif}$
$\boldsymbol{endfor}$
$\boldsymbol{endwhile}$
$\boldsymbol{return}$ π
$\boldsymbol{end}$

Figure 4-4 Iterative Improvement procedure using insertion neighborhood

The IG algorithm by Ruiz and Stützle (2007) uses simulated annealing-like acceptance criterion. The Temperature value is calculated as shown in Eq. 12.

$$Temperature = T \cdot \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}}{n \cdot m \cdot 10} \quad (12)$$

T value is again an important parameter for the performance of the IG algorithm and needs to be calibrated carefully. R. Ruiz and T. Stützle also used the same IG algorithm for PFSP with sequence dependent set up times (Ruiz & Stützle, 2008) and obtained superior results.

4.3 Variable Neighborhood Search

Variable neighborhood search (VNS) is a popular metaheuristic algorithm proposed by Mladenovic and Hansen (Mladenovic & Hansen, 1997) . VNS algorithm visits the neighbor solutions of the current solution and updates the current solution if a better solution is found. The Basic VNS algorithm developed by Mladenovic and Hansen is given in Figure 4-5 (Hansen & Mladenović, 2001). $\mathcal{N}_{k}(k = 1, \dots, k_{max})$ is a finite set of pre-defined neighborhood structures and $\mathcal{N}_{k}(x)$ the set of solutions in the k^{th} neighborhood of x.

$\boldsymbol{Initialization.}$

$Select\ the\ set\ of\ neighborhood\ structures\ \mathcal{N}_{k}, k = 1, \dots, k_{max}$
$that\ will\ be\ used\ in\ the\ search$
$find\ initial\ solution\ x$
$choose\ a\ stopping\ condition$

$\boldsymbol{Repeat\ the\ following\ until\ the\ stopping\ condition\ is\ met}$:

(1)$Set\ k \leftarrow 1$

(2)$Until\ k = k_{max}, repeat\ the\ following\ steps$:

$(a)\ Shaking.\ Generate\ a\ point\ x'\ at\ random\ from\ the\ k^{th} neigborhood\ of\ x\ \left(x' \in \mathcal{N}_{k}(x)\right)$

$(b)\ Local\ Search.\ Apply\ some\ local\ search\ method\ with\ x'\ as\ initial\ solution : denote\ the\ so\ obtained\ local\ optimum\ with\ x''$

$\boldsymbol{(c)\ If\ this\ local\ optimum\ is\ better\ then\ the\ incumbent, move\ there\ (x \leftarrow x'')\ and\ continue\ the\ search\ with\ \mathcal{N}_{1}(k \leftarrow 1)}$
$\boldsymbol{otherwise, set\ k \leftarrow k + 1}$

Figure 4-5 Basic VNS Algorithm

Neighborhood search methodologies can be different; either random or predetermined neighbors can be visited to escape from local optimum values. Different solution sets can be constructed by using different neighborhood structures. Figure 4-6 (El-Ghazali, 2009) shows different local optima and global optima for two different neighborhoods.

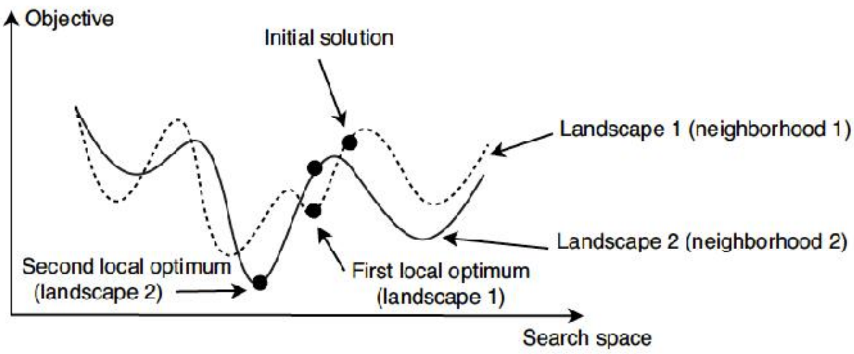

Figure 4-6 VNS using different neighborhoods

There are two main implementations of Variable Neighborhood Search (VNS) algorithm. The first implementation is Variable Neighborhood Descent (VND) which is the deterministic version of VNS. In VND, successive neighborhoods are used in iterative searching methodology. Initially, neighborhood structures are determined as $N_l(l = 1, \ldots, l_{max})$. First, an initial value x is generated. Then in a loop, all neighbors of respect to neighborhood N_l is generated. If the best neighbor x' is better than x, then x is replaced with x' and l is set to 1. When there is no improvement, VND algorithm switches to neighborhood N_{l+1}. At the end of the algorithm, the reported best solution will be a local optimum with respect to l_{max} neighborhoods. Determination of the neighborhood structures that will be used in VND is very crucial for the performance of the algorithm. Pseudocode of the VND algorithm is given in Figure 4-7 (El-Ghazali, 2009).

$\boldsymbol{Input}$: $a\ set\ of\ neighborhood\ structures\ N_l\ for\ l = 1, \ldots, l_{max}$
    $x = x_0$; $-Initial\ Solution\ Generation$
    $l = 1$;
    $\boldsymbol{While}\ l \le l_{max}\ \boldsymbol{Do}$
        $Find\ the\ best\ neighbor\ x'\ of\ x\ in\ N_l(x)$;
        $\boldsymbol{If}\ f(x')\ \boldsymbol{Then}\ x = x'$; $l = 1$;
        $\boldsymbol{Otherwise}\ l = l + 1$;
$\boldsymbol{Output}$: $Best\ found\ solution$

Figure 4-7 Pseudocode of the VND algorithm

The second common implementation of the VNS algorithm is the General Variable Neighborhood Search (GVNS) algorithm. In contrast to VND, GVNS is a stochastic algorithm. Neighborhood structures to be used for shaking and local search are determined first. GVNS algorithm consists of three phases: shaking, local search and changing the neighborhood. A new iteration starts with shaking step using the current neighborhood to generate a new solution x' randomly. Local search is applied to x' in the second phase. Last phase is determination and comparison. If x' is better than the current best solution, then the current neighborhood to be used in the next iteration to generate a new solution is set to the first neighborhood. If x' is not better, then algorithm moves to new neighborhood. The pseudo code for GVNS algorithm is given in Figure 4-8 (El-Ghazali, 2009).

$\boldsymbol{Input}$: $a\ set\ of\ neighborhood\ structures\ N_k\ \ (k = 1, \dots, k_{max})\ for\ shaking$
    $a\ set\ of\ neighborhood\ structures\ N_l\ (l = 1, \dots, l_{\max})\ for\ local\ search$
    $x = x_0\ \ - Generate\ initial\ Solution$
    $\boldsymbol{Repeat}$
        $\boldsymbol{For}\ k = 1\ \ \boldsymbol{To}\ k_{max}\ \boldsymbol{Do}$
            $-Shaking\ Step$:
            $pick\ a\ random\ solution\ x'\ from\ the\ k^{th}\ neighborhood\ N_k(x)\ of\ x$;
            $-Local\ Search\ using\ VND$:
            $\boldsymbol{For}\ l = 1\ \boldsymbol{To}\ l_{max}\ \boldsymbol{Do}$
                $Find\ the\ best\ neighbor\ x''\ of\ x'\ in\ N_l(x')$;
                $\boldsymbol{If}\ f(x'') < f(x')\ \boldsymbol{Then}\ x' = x''; l = 1$;
                $\boldsymbol{Otherwise}\ l = l + 1$;
            $\boldsymbol{If}\ local\ optimum\ is\ better\ than\ x\ \boldsymbol{Then}$
                $x = x'$;
                $Continue\ the\ search\ with\ N_l\ (\ k = 1)$;
            $\boldsymbol{Otherwise}\ k = k + 1$;
    $\boldsymbol{Until}\ Stopping\ criteria$
    $\boldsymbol{Output}$: $Best\ found\ solution$

Figure 4-8 Pseudocode for GVNS algorithm

Local search phase allows to find better local optimum solutions, whereas shaking phase may lead to a jump to better regions of the search space to find better solutions. In local search phase, the algorithm mostly looks for local optimum values in a narrow search space. If the shake values are big, then after each shake phase, the algorithm starts over from a new point away from the local optimum value which may cause the algorithm to steer away from a promising region in the search space. Determination of the neighborhood structures and the strength of the shake are crucial for the performance of the GVNS algorithm.

5 DESIGN OF EXPERIMENTS APPROACH

Designing an experiment is a procedure of planning to how collect convenient data, which can be analyzed using statistical methods. Statistical methods are necessary for creating valid and accountable results from the collected data. There are two major considerations in designing an experiment; creating an adequate experiment environment and analyzing the data with proper statistical analysis methods. Both factors play important roles in producing correct experiment results.

Experimental design has three fundamental factors: replication, randomization and blocking (Montgomery, 2001). Replication means repeating the test several times. Experimental assumption errors and variance of the results will be decreased with repetition of the experiments. Randomization means selecting the experimental material randomly. Randomization allows lowering the effects of boundary values in computations. Blocking is used for improving correctness of the experiment by creating comparable test environments.

A correctly planed experiment can give good and meaningful statistical results for showing the response of a variable. In many studies certain factors are held constant and the effect of a single factor is examined. This kind of approach called as one factor at a time (OFAT). Nevertheless, this approach is not sufficient for a system in which factor values change over time.

For systems whose planning concerns more than two factors, factorial design is more efficient. Factorial design is more efficient in showing effects of more than one factor on the output. By using factorial design, effects of the factors can be seen statistically in the results and cross interaction of the factors can be determined.

Number of experiments to be carried out can be lowered by testing more than one factor concurrently in one experiment by using factorial design. As an example to factorial design, let an independent factor A has “a” number of levels and independent factor B has “b” number of levels. Factorial design will have all combinations of independent factors A and B and will include all combinations of $a \times b$. Factorial design also shows the main effect and interaction effects of the independent factors. However, main effect and interaction effects only give analogous result of factors. In order to make a more adequate analysis, statistical methods such as regression should be applied. On the other hand, in factorial design, more than one factor is tested, but not all of these independent factors have to have an effect on the result. Therefore, it is also important to decide which of the independent factors are significant.

Main effect is the direct effect of the independent factor on the result. In some cases, the effect of the independent factor may change in relation to another independent factor. This is called as interaction effect. Both effects can clearly be seen at the end of the statistical analysis of factorial design experiment result.

To demonstrate factorial design, and main effect and interaction effect, an example is presented (ReliaSoft Corporation, 2015). Let X and Y be two independent factors. Moreover, let responses X_{low} and X_{high} represent the high and low values of X and Y_{low} and Y_{high} represent the high and low values for independent factor Y . There are two independent factors and two level design is planned. A two by two matrix is constructed for all four possible combinations as shown in Table 5-1.

		Y	
		Y_{low}	Y_{high}
X	X_{low}	25	35
	X_{high}	45	55

Table 5-1 Two factor factorial design - First example

The main effect of the factor X_{effect} can be found by calculating the average of the results when X is high and low. The change in the result caused by X is called the main effect of X.

$$X_{effect} = Average\ Response\ at\ X_{high} - Average\ Response\ at\ X_{low}$$

$$= \frac{45+55}{2} - \frac{25+35}{2}$$

$$= 50 - 30$$

$$= 20$$

As the factor X changes from X_{low} to X_{high}, experiment results change by 20 units regardless of Y values. As it is seen from the interaction plot in Figure 5-1, lines of the two factors go parallel, meaning that, there is no interaction between these two factors for this experiment.

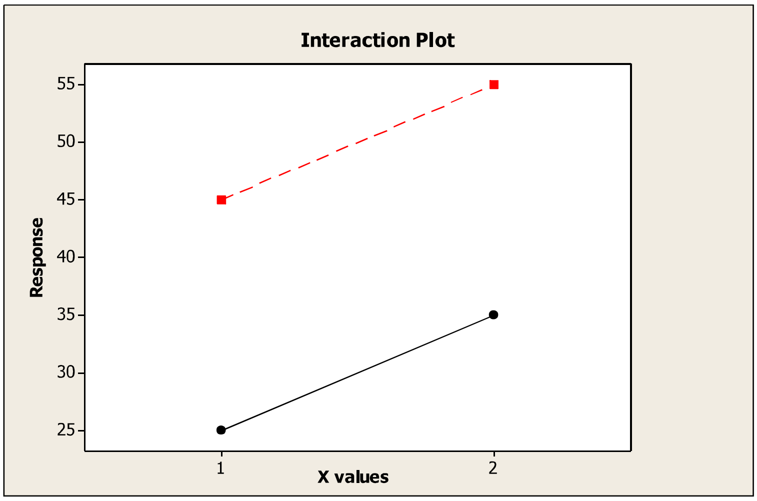

Figure 5-1 Interaction Plot of Factor X and Factor Y

There is no interaction between factor X and Y in the first example. Suppose that, a new experiment is designed for a different system. The result of the new experiment is shown in Table 5-2.

		Y	
		Y_{low}	Y_{low}
X	X_{low}	*20*	*30*
	X_{high}	*40*	*10*

Table 5-2 Two factor Factorial design - Second example

For this experiment, main effect of X can be calculated as follows:

$$X_{effect} = (40 + 10)/2 - (20 + 30)/2 \quad = 0$$

Effect of the independent factor is calculated as zero. However, with different values of X, response of the experiment differs (for $X_{low} = 20$ and $X_{high} = 40$). But this effect on the response is dependent on the level of Y. Interaction between X and Y can be calculated using the above formulation as follows:

$$XY = \left(Average\ Response\ at\ X_{high} - Y_{high}\ and\ X_{low} - Y_{low}\right) -$$
$$(Average\ Response\ at\ X_{low} - Y_{high} and\ A_{high} - Y_{low})$$
$$= (10 + 20)/2 - (40 + 30)/2$$
$$= \ -20$$

An intersection in interaction plot means that there is an interaction between factors. In Figure 5-2, there is an interaction between factors X and Y, so the effect of one factor depend on the other factor.

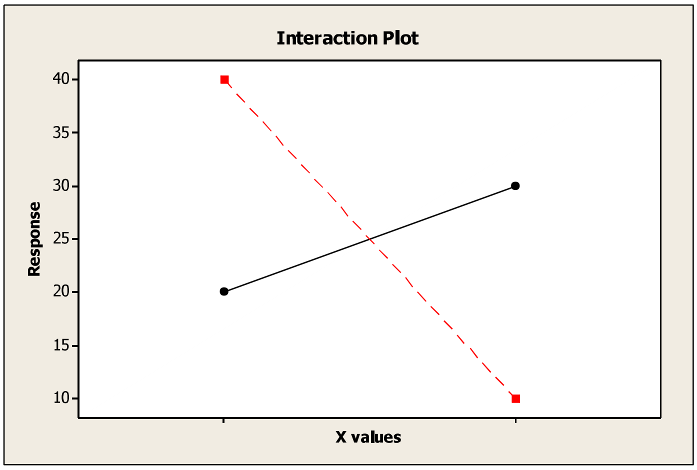

Figure 5-2 Interaction Plot of Factor X and Factor Y for second example

6 ALGORITHMS DEVELOPED TO SOLVE PERMUTATION FLOW SHOP PROBLEM WITH SEQUENCE DEPENDENT SETUP TIMES

In this chapter, the algorithms developed for solving the SDST-PFS problem under makespan and total flow time are presented. The design of experiment (DOE) approach used to fine tune the parameters of the proposed algorithms and the results of the DOE are given in detail.

6.1 Iterated Greedy Algorithm with Iteration Jumping for Makespan Minimization

The VND algorithm changes neighborhoods systemically; it moves from one neighborhood structure to another in the search space to find a solution which is a local optimum with respect to all neighborhood functions used in the search. In the literature it is proposed that, insertion neighborhood structure is more effective than swap neighborhood structure for makespan minimization (Grabowski & Wodecki, 2004) (Nowicki & Smutnicki, 1996) (Ruiz & Stützle, 2007). Implementing both neighborhood structures in a VND algorithm can increase the computational cost of the algorithm in terms of CPU time. However, using the swap neighborhood can help to escape local optimum values found using insertion neighborhood by traversing different points in the search space. Using swap neighborhood can increase the algorithm’s performance in terms of solution quality, in the expense of extra CPU time. So, swap neighborhood should be used much less than insertion. In this thesis, an iteration jumping structure is used to control when the swap neighborhood will be invoked in the local search phase. In the proposed iteration jumping scheme, a random number is generated at the start of the local search phase in order to determine whether to use the insertion or swap neighborhood. The probability of using the swap neighborhood is generally very small since the swap neighborhood is costly in terms of CPU time as described before. An IG algorithm using iteration jumping in the local search is proposed and named as Iterated Greedy algorithm with

Iteration Jumping (IG_IJ). The pseudo code of proposed IG_IJ algorithm is given in Figure 6-1.

$\boldsymbol{procedure}\ IG\_IJ\ ()$
    $Set\ d,\ jP, \tau$
    $\pi_0 = NEH\_RMB$
    $\pi = LocalSearch(\pi_0, jP)$
    $\pi_{best} = \pi$
    $\boldsymbol{while}\ (Not\ Termination\ )\ \boldsymbol{do}$
        $\pi_1 = DestructConstruct(\pi, d)$
        $\pi_2 = LocalSearch(\pi_1, jP)$
        $\boldsymbol{if}\ \left(f(\pi_2) < f(\pi)\right)\ \boldsymbol{then}$
            $\pi = \pi_2$
            $\boldsymbol{if}\left(f(\pi_2) < f(\pi_{best})\right)\ \boldsymbol{then}$
                $\pi_{best} = \pi_2$
            $\boldsymbol{endif}$
        $\boldsymbol{else\ if}\ (\ random() < exp\{-\frac{f(\pi_2)-f(\pi)}{T}\})\ \boldsymbol{then}$
            $\pi = \pi_2$
        $\boldsymbol{endif}$
    $\boldsymbol{end\ while}$
    $\boldsymbol{return}\ \pi$
$\boldsymbol{end\ procedure}$

Figure 6-1 The proposed IG_IJ algorithm

An initial solution is generated using the NEH_RMB algorithm and local search is applied to the initial solution. The IG algorithm proposed by Ruiz and Stützle (2007) can be considered as a special case of the IG_IJ algorithm with jumping probability set to zero. Unlike VND, IG_IJ algorithm does not always switch to swap neighborhood, but it uses the swap neighborhood with a small predetermined probability, thus giving more chance to the insertion neighborhood structure that is very effective under makespan optimization criterion. The local search with the iteration jumping probability is given Figure 6-2.

$\boldsymbol{procedure}\ LocalSearch(\pi\ , jP\ )$
    $flag = true$
    $\boldsymbol{while}\ (\ flag = true\ )\ \boldsymbol{do}$

```
            flag = false
            if (r < jP ) then
                for ( i = 1 to n − 1) do
                    for ( j = i + 1 to n)  do
                        swap job π_i with π_j and save makespan value
                    endfor
                endfor
                π_1 = best sequence obtained by n ( n − 1)/2  evaluations
                if ( f(π_1) <  f(π)) then
                    π =  π_1
                    flag = true
                endif
            else
                for ( i = 1 to n )  do
                  π_1 = remove job k randomly from π ( without repetion)
                  π_2 = best sequence obtained by inserting job k in all slots of π_1
                  if ( f(π_2) <  f(π)) then
                    π =  π_2
                    flag = true
                  endif
                endfor
            endif
        endwhile
        return π
end procedure
```

Figure 6-2 Local Search algorithm used in IG_IJ

The Proposed local search algorithm uses both insertion and swap neighborhoods with an iteration jumping probability (jP). The reason for using an iteration jumping probability to control the usage of the swap neighborhood is the computational cost of the swap neighborhood structure. The cost of swap operation is still high even with the new speed-up technique. However, usage of swap neighborhood has positive effects as shown in computational results. The proposed local search is simple in a way that if a uniform random number r is smaller than the iteration jumping probability (jP), the swap neighborhood-based local search is employed. Otherwise, the insertion neighborhood-based local search is carried out.

The size of the insertion neighborhood is $(n-1)^2$ and its complexity is $O\ (n^3m)$ in the proposed local search algorithm. Taillard's speed-up method reduces the complexity to $O\ (n^2m)$ as explained in previous chapters. The size of the swap neighborhood is $n(n-1)/2$ and each evaluation takes $n \cdot m$ operations to calculate the objective function, so total computational complexity is $O\ (n^3m)$. The proposed calculation technique does not reduce the computational complexity, but it decreases the average computational time by 50%. So, the computational time required for the swap operation can still be high for large n values.

6.2 Iterated Greedy Algorithm with Variable Neighborhood Search for Makespan Minimization

As considered before, VNS algorithm plays an important role in escaping from local optima (valleys) by switching the neighborhood structures systematically. Two different versions of the IG algorithm with VNS based local search is proposed. The first version uses insertion as the first neighborhood and swap as the second neighborhood and is called IG_VNS1, while the second version named IG_VNS2 uses swap neighborhood as the first neighborhood and insertion neighborhood as the second neighborhood. Both algorithms use destruction and construction procedure for shaking. The general structure of both algorithms in given in Figure 6-3. As mentioned above, the difference of the two IG_VNS versions is in the application order of the neighborhood structures used in local search phase of the VNS algorithm.

$\boldsymbol{procedure}\ IG\_VNS$

$\quad \pi = NEH\_RMB$

$\quad \pi_{best} = \pi$

$\quad \boldsymbol{while}\ (\,Not\ Termination)\ \boldsymbol{do}$

$\quad\quad k_{max} = 2$

$\quad\quad k = 1$

$\quad\quad \pi_1 = DestructConstruct(\pi, d)$

$\quad\quad \boldsymbol{do}\,\{$

$\quad\quad\quad \pi_2 = N_k(\pi_1)$

$\quad\quad\quad \boldsymbol{if}\big(f(\pi_2) < f(\pi_1)\big)\ \boldsymbol{then}$

$\quad\quad\quad\quad \pi_1 = \pi_2$

$k = 1$
else
$k = k + 1$
endif
while $(k \leq k_{max})$
if $\left(f(\pi_1) < f(\pi)\right)$ ***then***
$\pi = \pi_1$
if $\left(f(\pi_1) < f(\pi_{best})\right)$ ***then***
$\pi_{best} = \pi_1$
endif
else if $\left(r < exp\left\{-\frac{(f(\pi_1)-f(\pi))}{T}\right\}\right)$ ***then***
$\pi = \pi_1$
endif
end while
return π_{best}
end procedure

Figure 6-3 The pseudo code of the proposed IG_VNS algorithms

6.3 Experiment Design for Make Span Minimization Criterion

(Ruiz & Stützle, 2008) created a new benchmark set for permutation flow shop problem with sequence dependent setup times which is based on Taillard's benchmark suite. The original Taillard's benchmark suite does not have sequence dependent setup times. It has 120 instances in total in twelve groups. The groups contain different $n\,x\,m$ combinations, which are $\{20, 50, 100\}\,x\,\{5, 10, 20\}$, $\{200\}\,x\,\{10, 20\}$ and $\{500\}\,x\,\{20\}$. The processing times in Taillard's instances are generated from a uniform distribution in the range $[1{,}99]$. (Ruiz & Stützle, 2008) added sequence set up times to Taillard's suit using different uniform distributions whose ranges' are $[1{,}9]$, $[1{,}49]$, $[1{,}99]$ and $[1{,}124]$. These new test instance groups are named as SDST10, SDST50, SDST100 and SDST125 respectively. There are 480 test instances in total in four different SDST groups.

A new benchmark suite is created by following the same construction procedure used in (Ruiz & Stützle, 2008) in order to carry out experiments. New test instances are generated with combinations of

$\{20\,,50\,,100\,,200\,,500\,\}\,x\,\{5\,,10\,,20\,\}$. 75 new test instances are generated with sequence dependent setup times that are uniformly distributed between $[1{,}124]$.

The first proposed IG algorithm (IG_IJ) for makespan optimization problem has three parameters: destruction size d, iteration jumping probability jP and temperature adjustment parameter τ. A full factorial design is created by using the Design of Experiments (DoE) approach of Montgomery (2001). In the first step, the effects of three factors with different levels are examined. Destruction size d has five levels (4,5,6,7,8), iteration jumping probability jP also has five levels (0.0, 1.0, 0.1, 0.01, 0.001) and temperature adjustment parameter τ has ten levels (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0). Note that, when the iteration jumping probability is equal to 0.0, it means that only the insertion neighborhood structure is used in local search, whereas, when jP equals to 1.0, it means that, only swap neighborhood structure will be used in local search. When jP equals to 0, IG_IJ is equivalent to IG_RS_{LS}.

A full factorial design is conducted, resulting in $5\,x\,5\,x\,10 = 250$ treatments. Each instance in the generated test set is run for 250 treatments with a maximum CPU limit set to $T_{max} = n\,x(m/2)x15$ milliseconds for each run. The performance variable, relative percent deviation, is calculated as follows:

$$\boldsymbol{RPD} = \sum_{r=1}^{R}\left(\frac{(C_i - C_{min})x100}{C_{min}}\right) \tag{13}$$

where C_i is the makespan value generated in each run and C_{min} is the minimum makespan obtained amongst 250 treatments. Ten replications are carried out for each treatment. RPD values are calculated and averaged for each treatment. Then, the response variable is obtained by averaging the RPD values of 75 different job-machine combinations for each treatment.

The main effects plot is given in Figure 6-4. Figure 6-4 suggests that the destruction size should be taken at 3rd level as 6, the iteration jumping probability should be taken at 5th level as 0.001 and the temperature adjustment parameter should be taken at 10th level as 1.0. The main effects plot of the iteration jumping probability suggests some insights about the neighborhood structures. The first observation is that, insertion neighborhood structure is significantly much more effective than swap neighborhood structure. However, giving a little chance to swap neighborhood structure can enhance solution quality without jeopardizing the effectiveness of insertion neighborhood structure. RPD values with $jP = 0.001$ achieves almost the same level of using only insertion neighborhood structure.

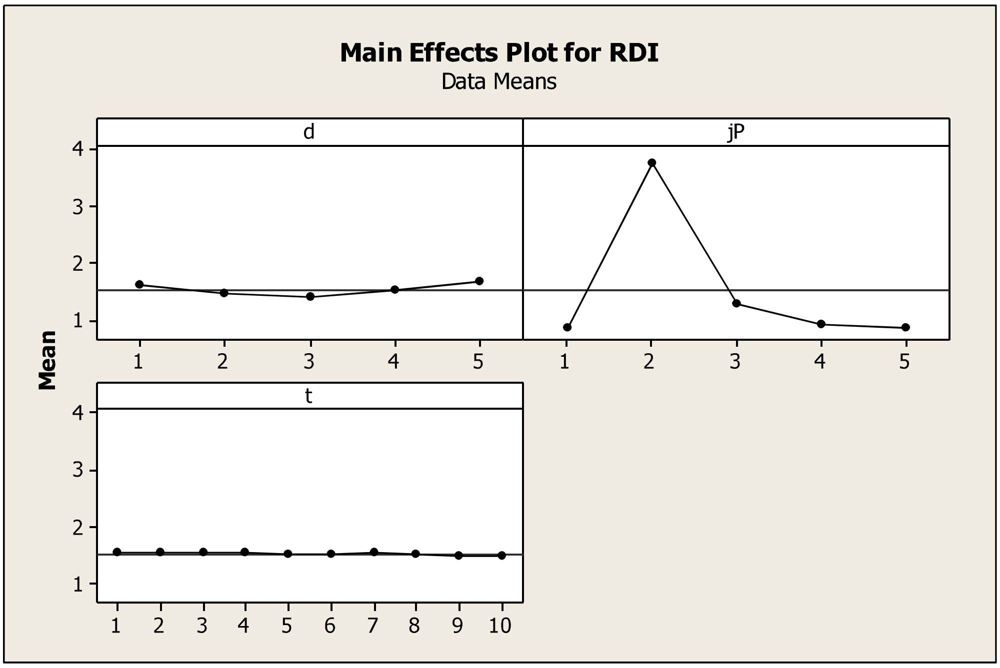

Figure 6-4 Main effect plots of parameters

Next, ANOVA test is carried out for this experiment. Note that, all three hypotheses of the ANOVA method (normality, homoscedasticity and independence of the residuals) were checked and accepted. The results of the ANOVA test is given in Table 6-1. As observed from the table, $d * jP$, and $d * \tau$ interactions are significant, since the calculated p values are less than $\alpha = 0.05$ level. For this reason, interaction plots should be analyzed in order to support the judgment inferred from main effects plots. The interaction plot for $d * jP$ is given in Figure 6-5.

Source	DF	Seq SS	Adj SS	Adj MS	F	p
d	4	2.364	2.364	0.591	**111.080**	**0.000**
jP	4	312.814	312.814	78.204	**14699.740**	**0.000**
τ	9	0.077	0.077	0.009	1.610	0.119
$d*jP$	16	2.718	2.718	0.170	**31.930**	**0.000**
$d*\tau$	36	0.385	0.385	0.011	**2.010**	**0.002**
$jP*\tau$	36	0.215	0.215	0.006	1.120	0.309
Error	144	0.766	0.766	0.005		
Total	249	319.339				

Table 6-1 ANOVA table of IG_IJ$_{LS}$

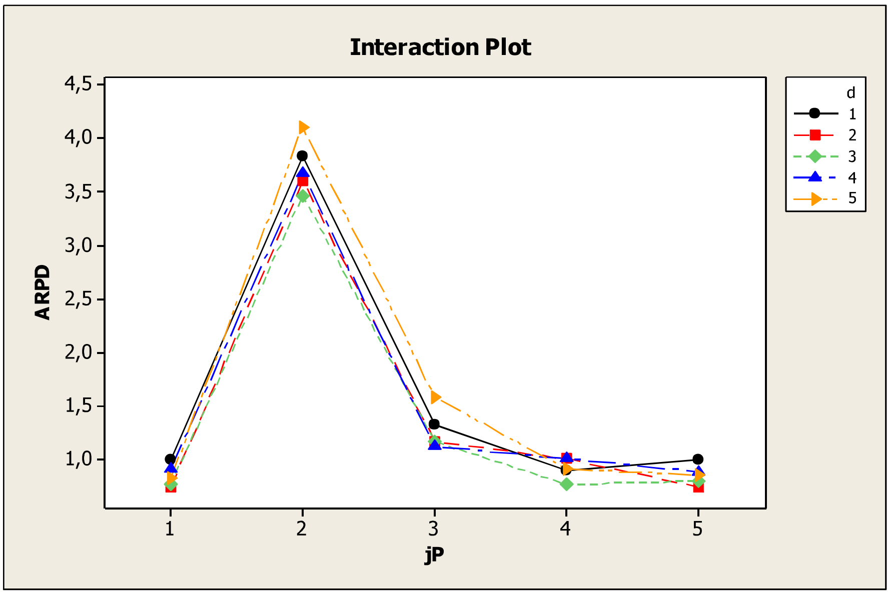

Figure 6-5 Interaction plot for destruction size versus jumping probability

In Figure 6-5, destruction size in the third level ($d = 6$) generates the lowest RPD value (RPD = 0.762966) with jP at the fifth level ($jP = 0.001$). Note that, level 1 and level 4 of iteration jumping probability are competitive, since their RPD values are 0.769684 and 0.767644, respectively.

The interaction plot for $d^*\tau$ is given in Figure 6-6. From Figure 6-6, it is interesting to see that the temperature adjustment parameter should be selected as $\tau = 0.2$, unlike the one suggested in the main effects plot.

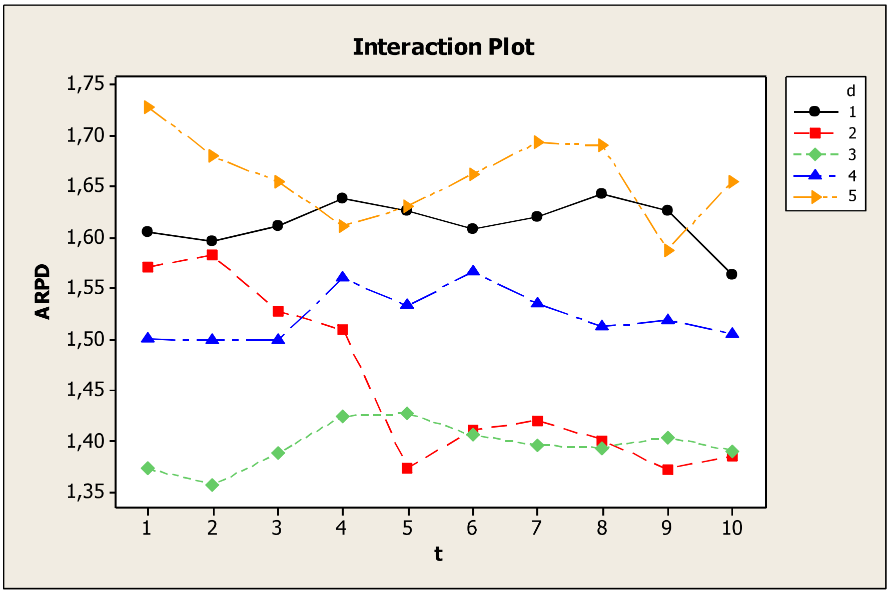

Figure 6-6 Interaction plot for destruction size versus temperature adjustment parameter

In Figure 6-7, it is observed that, jP levels 1, 4 and 5 have no statistical significance, so any of the three values can be selected.

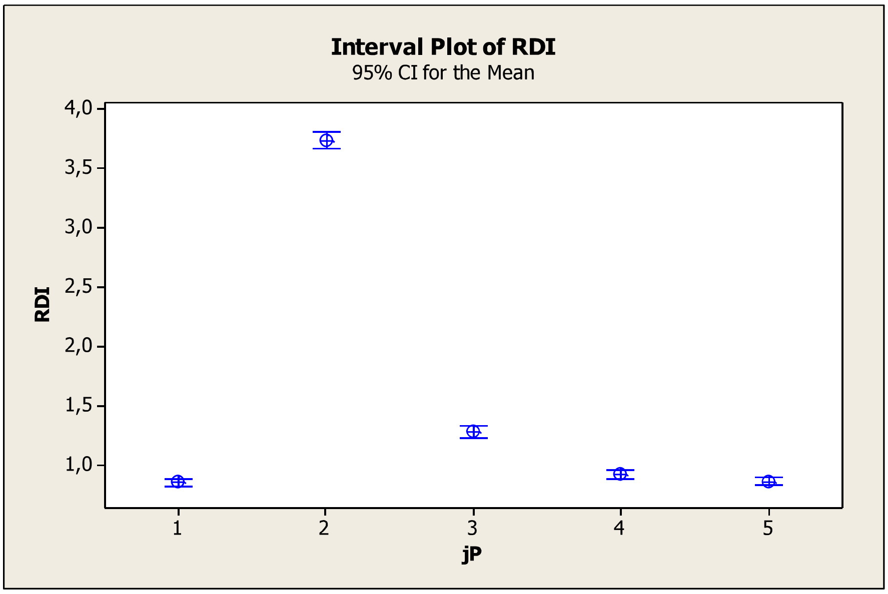

Figure 6-7 Interval Plot of jP (jumping probability)

Figure 6-8 suggests that different d values do not have statistical significance. After the above analysis, the following parameters parameter values will be used: $d = 6, jP = 0.001$ and $\tau = 0.2$.

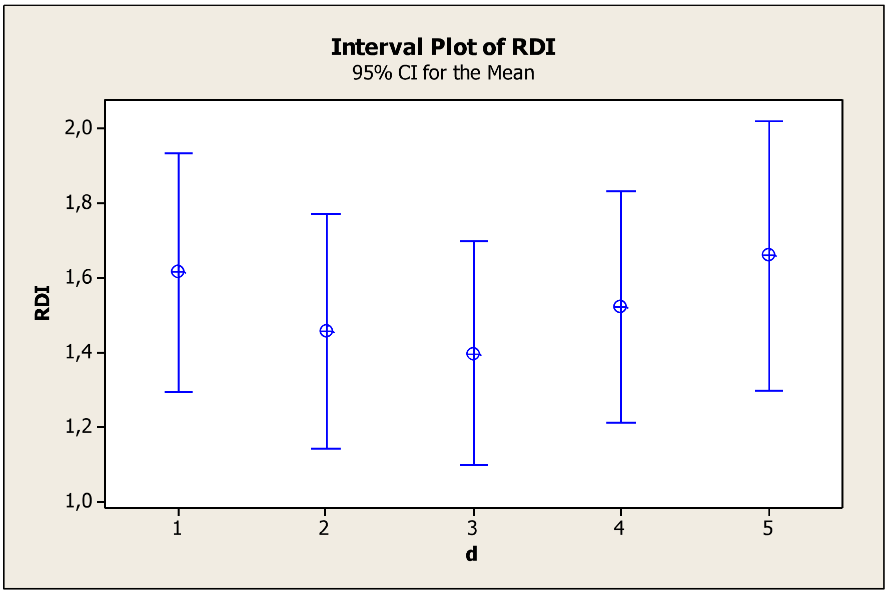

Figure 6-8 Interval Plot of d size

6.4 Variable Local Search Algorithm for Total Flow Time Minimization

Variable local search is a local search technique which gives opportunity for implementing different local searches using a specified criterion. Shi et al. (2007) implemented Variable meta – Heuristic Local Search (VHLS) algorithm for multi objective flow-shop problem. Ribas et al. (2015) developed a variable local search algorithm for blocking flow shop problem. They have developed a new variable local search algorithm called VLS_{RCT} and employed it in a discrete artificial bee colony algorithm. VLS_{RCT} is developed for finding so called good food sources in DABC algorithm framework.

VLS_{RCT} is a combination of two local search algorithms which are named as LS1 and LS2. VLS_{RCT} procedure uses both local search algorithms in its implementation. The interesting approach in their implementation is that, the first algorithm to be executed first is determined randomly, and the other algorithm is executed if the current algorithm can find a better solution than the current best solution. This process continues in a loop that terminates if the current algorithm cannot find an improvement and both of the algorithms have executed at least once. The pseudocode for the VLS_{RCT} is given in Figure 6.9.

$\boldsymbol{Procedure\ VLS\_RCT}$
    $TF^* = TF(\pi)$
    $\pi^* = \pi;$
    $nm = 0$
    $\boldsymbol{if}\ random < \beta\ \boldsymbol{then}$
        $ls = 0$
    $\boldsymbol{else}$
        $ls = 1$
    $\boldsymbol{endif}$
    $\boldsymbol{do}$
        $nm = nm + 1$
        $TF_0 = TF(\pi)$
        $\boldsymbol{If}\ ls = 0\ \boldsymbol{then}$
            $LS1$
        $\boldsymbol{else}$

$LS2$

endif

if $TF(\pi) < TF_0$ ***or*** $nm = 1$ ***then***

$ls = 1 - ls$

else

exit do

endif

loop

end procedure

Figure 6-9 VLS$_{RCT}$ algorithm

LS1 algorithm generates a new schedule by swapping a randomly selected job with all other jobs that comes after it. If the new solution is better than the current best solution, then best solution so far is updated with the new solution. This process continues until all jobs are swapped with other jobs. The job to be swapped is determined randomly in order to consider different schedules in each local search iteration. LS2 uses insertion neighborhood. In LS2, a job is selected randomly and is removed from the permutation. Then, this job is placed in all possible positions and the best position for insertion is selected. This procedure continues until all jobs are removed and inserted.

A new VLS algorithm based on VLS$_{RCT}$ is developed and is named as VLS$_{IKT}$. The proposed VLS$_{IKT}$ algorithm uses both insertion and swap neighborhoods. Following the notation used in VLS$_{RCT}$, local search algorithms are named as LS1 and LS2. Local search algorithm LS1 is the swap neighborhood structure as in VLS$_{RCT}$. However, usage of the swap neighborhood structure is different from the VLS$_{RCT}$. LS1 algorithm in VLS$_{IKT}$ does not select the job to be swapped randomly, instead, it first considers the first job, then the second job, and so on. If a better schedule is found, the current best permutation is updated with the new schedule and swap operation is terminated. Hence, first improvement pivoting rule is used. Implementation of the LS2 algorithm is same as it is in VLS$_{RCT}$: a randomly selected job is removed from the current permutation and is inserted to the all possible positions. The best insertion position is selected and if the new permutation is better than the current permutation, current best permutation is replaced.

The VLS_{IKT} is used as the local search method in an iterated greedy algorithm named as IG_VLS_{IKT} which is almost identical to IG_RS algorithm (Ruiz & Stützle, 2008). The difference in IG_VLS_{IKT} is that, it uses VLS_{IKT} in the local search phase instead of iterated insertion local search in IG_RS. The pseudocode for IG_VLS_{IKT} is given in Figure 6-10.

$\boldsymbol{Procedure\ IG\_VLS\_IKT}(\pi, d)$
$\quad \pi = NEH_{EDD}$
$\quad \pi = VLS\_IKT(\pi)$
$\quad \pi_{best} = \pi$
$\quad \boldsymbol{while}(\ termination\ criterion\ is\ not\ satisfied)\ \boldsymbol{do}$
$\qquad \pi' \coloneqq \pi;$
$\qquad \boldsymbol{for}\ i \coloneqq\ 1\ \boldsymbol{to}\ k\ \boldsymbol{do}$
$\qquad\quad \pi' \coloneqq remove\ a\ job\ at\ random\ from\ \pi' and\ insert\ it\ into\ \pi'_R$
$\qquad \boldsymbol{endfor}$
$\qquad \boldsymbol{for}\ i \coloneqq 1\ \boldsymbol{to}\ k\ \boldsymbol{do}$
$\qquad\quad \pi' \coloneqq best\ permutation\ obtained\ by\ inserting\ \pi'_{R_i}\ in\ all\ positions\ in\ \pi'$
$\qquad \boldsymbol{endfor}$
$\qquad \pi'' = VLS\_IKT(\pi')$
$\qquad \boldsymbol{if}\ C_{max}(\pi'') < C_{max}(\pi)$
$\qquad\quad \pi \coloneqq \pi''$
$\qquad\quad \boldsymbol{if}\ C_{max}(\pi) < C_{max}(\pi_{best})$
$\qquad\qquad \pi_{best} = \pi$
$\qquad\quad \boldsymbol{endif}$
$\qquad \boldsymbol{else\ if}\ random() < \exp\{-(C_{max}(\pi'') - C_{max}(\pi))/Temperature$
$\qquad\quad \pi \coloneqq \pi''$
$\qquad \boldsymbol{endif}$
$\quad \boldsymbol{endwhile}$
$\boldsymbol{end\ Procedure}$

Figure 6-10. The proposed IG_VLS_{IKT} algorithm

6.5 Design of Experiment for Total Flow Time Minimization

DoE approach is used again in order to identify the parameter values for IG_RS algorithm (Ruiz & Stützle, 2008), IG_VLS_{RCT} and IG_VLS_{IKT}. A full factorial design similar to the make span optimization problem is implemented. Full factorial design allows to observe the effect of the individual algorithm parameters and also interaction between them. The same test instances used in makespan minimization DoE are again used in the experiments. The same experiment design is used for three algorithms since their parameters are the same, however, results are evaluated separately.

All three algorithms have two parameters: destruction size (d) and temperature adjustment parameter (τ). Eleven levels (0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0) are considered for destruction size (d), and five levels (4, 5, 6, 7, 8) are considered for temperature adjustment parameter (τ). There are 5 x 11 = 55 different treatments in total. Maximum CPU time is set to $T_{max} = n\, x\, (m/2)\, x 15$ milliseconds for each run. The response variable relative percentage deviation is computed according to equation (5) in which C_i is the total flow time found in each run and C_{min} is the minimum total flow time found amongst all runs for the considered test instance. After all results are evaluated for each treatment, results are analyzed separately for each algorithm in order to determine their parameter values.

Main effect plots for destruction size and temperature adjustment parameter for IG_RS are shown in Figure 6-11 and Figure 6-12. The main effect plots suggest that, destruction size value should be taken as 8 and temperature adjustment parameter should be selected as 0.5 for IG_RS algorithm.

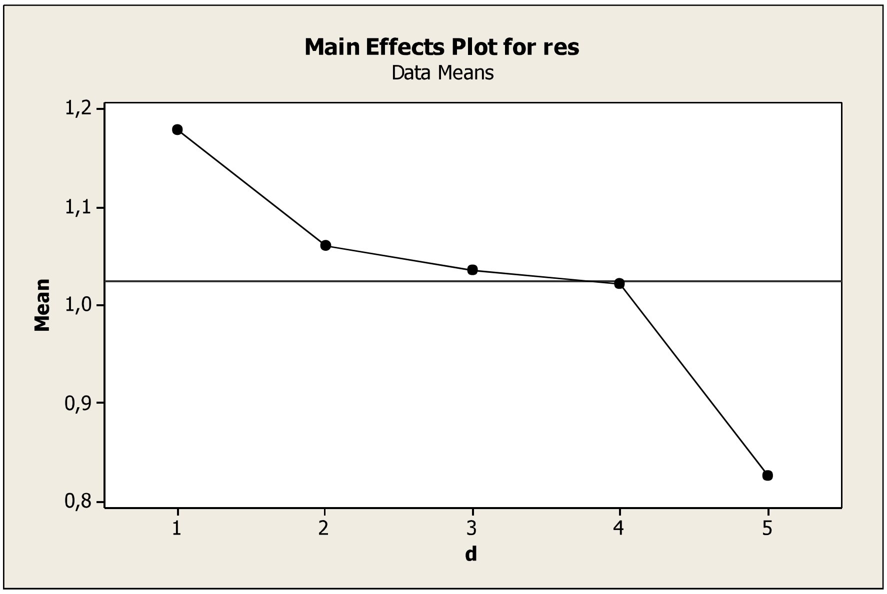

Figure 6-11 Main effect plots of d size for IG_RS algorithm

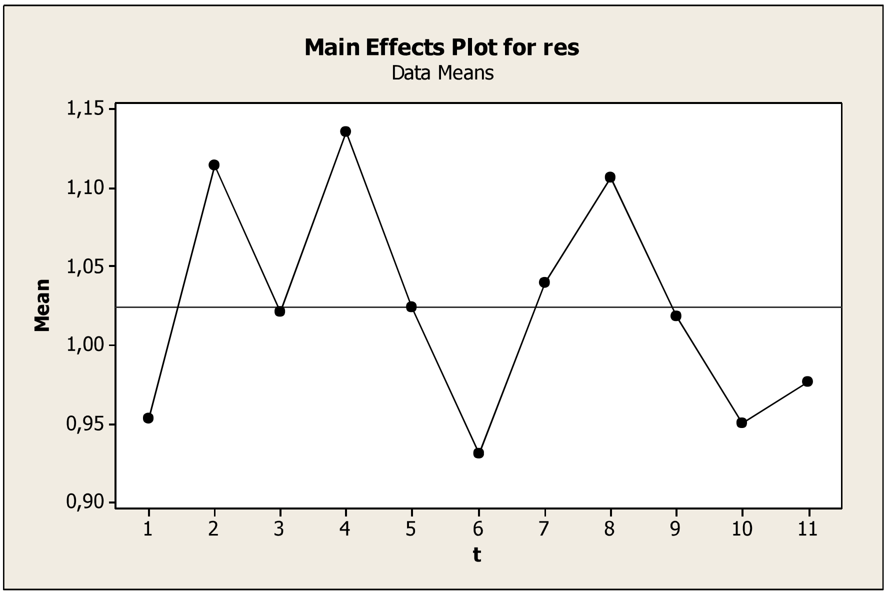

Figure 6-12 Main effect plots of temperature adjustment parameter for IG_RS algorithm

In order to determine whether the parameters and their interaction are significant, ANOVA analysis is used and the results of the analysis is given in Table 6-2. From the ANOVA table it is clear that, d is significant because its p value is less than $\alpha = 0.05$.

Source	DF	Sum Sq	Mean Sq	F	p
d	1	0.61385	0.61385	174.139	0.0001171
τ	1	0.01590	0.01590	0.4510	0.5048763
$d * \tau$	1	0.13026	0.13026	36.953	0.0601616
Residuals	51	179.779	0.03525		

Table 6-2 ANOVA table of parameters in IG_RS

Interaction plot for destruction size and temperature adjustment parameter is shown in Figure 6-13. In the interaction plot, the RPD values are lowest when $d = 8$ and $\tau = 0.0$ and also when $d = 6$ and $\tau = 0.9$.

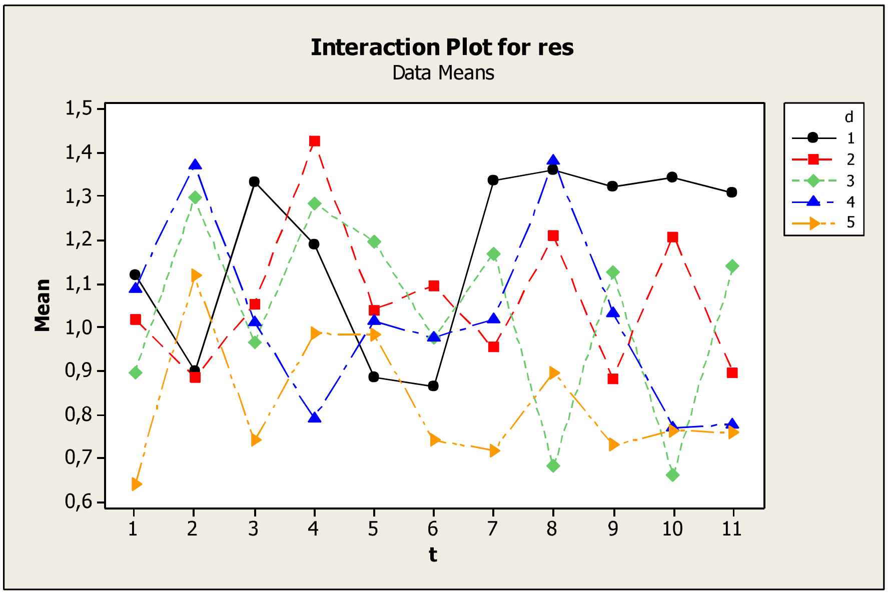

Figure 6-13. Interaction plot of temperature adjustment parameter and destruction size for IG_RS algorithm

Finally, interval plots of the two parameters destruction size and temperature adjustment parameter are given in Figure 6-14 and Figure 6-15. These plots suggest that the observed differences of RPD for different parameter values are not statistically significant. So, any value for the parameters can be chosen. The parameter combination $d = 4$ and $\tau = 0.5$ is chosen for IG_RS.

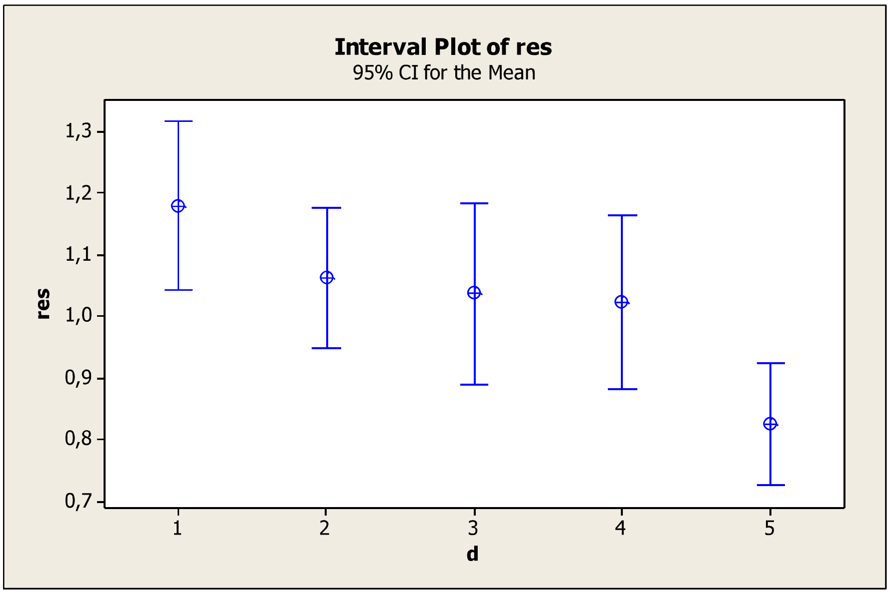

Figure 6-14 Interval plot of d size for IG_RS algorithm

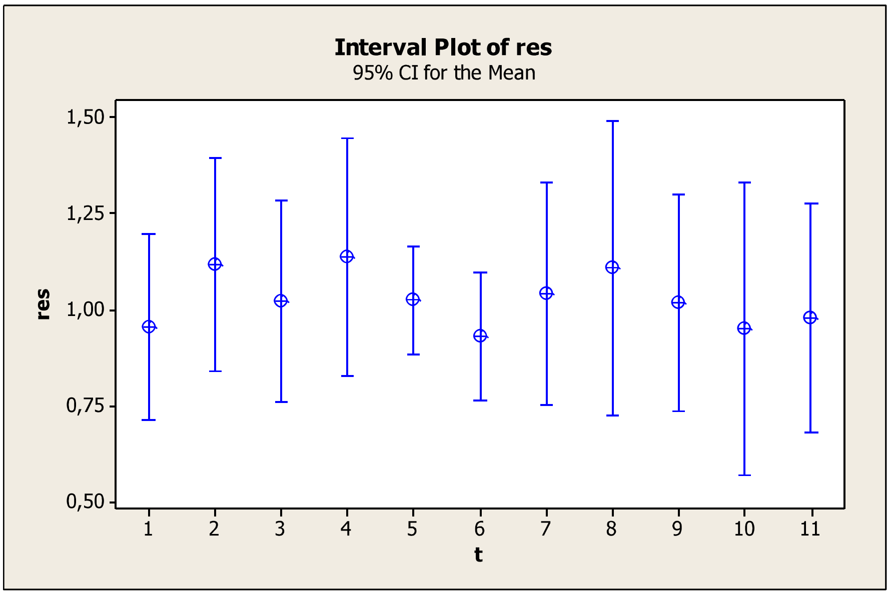

Figure 6-15 Interval plot of temperature adjustment parameter for IG_RS algorithm

The same procedure is followed for determining the parameters of IG_VLS$_{RCT}$ and IG_VLS$_{IKT}$. When the main effect graphs in Figure 6-16 and 6-17 are analyzed, it is observed that the lowest RPD values are achieved when $d = 8$ and $\tau = 0.4$.

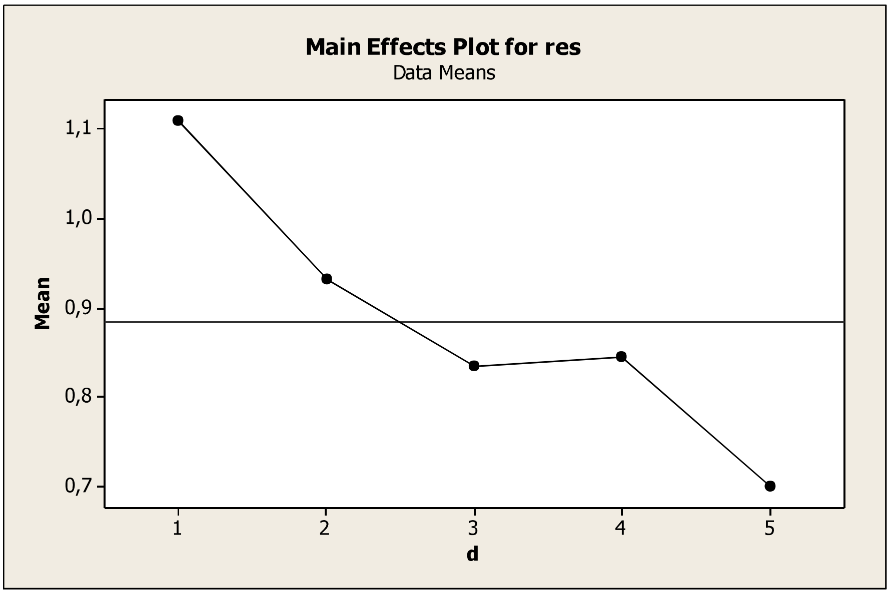

Figure 6-16 Main effect plots of d size for IG_VLS$_{RCT}$ algorithm

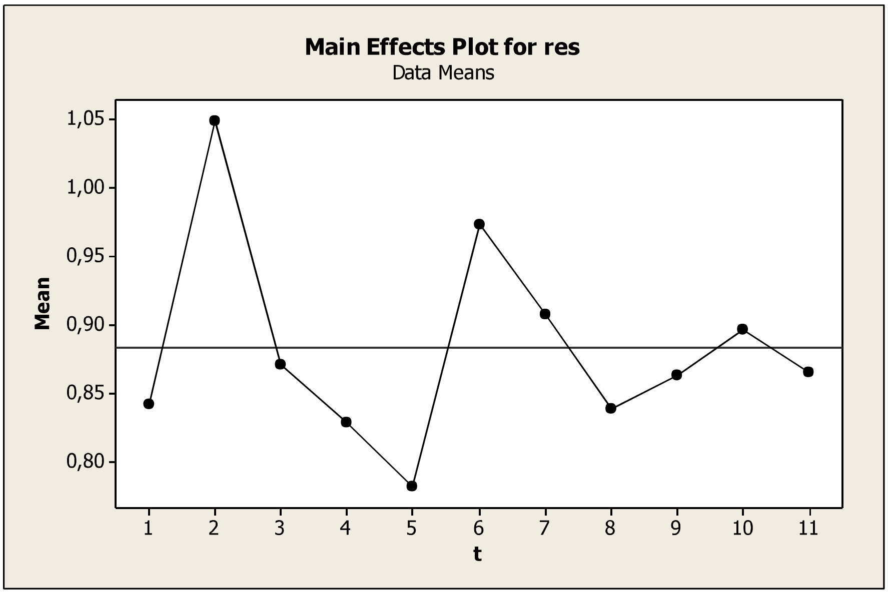

Figure 6-17 Main effect plots of temperature adjustment parameter for IG_VLS$_{RCT}$ algorithm

In order to see the significance of the parameters, ANOVA analysis is used again and the results are given in Table 6-3. It is seen that, only destruction size parameter is significant, while τ and $d * \tau$ interaction is not significant.

Source	DF	Sum Sq	Mean Sq	F	p
d	1	0.89496	0.89496	14.9734	0.0003109
τ	1	0.00620	0.00620	0.1037	0.7487798
$d * \tau$	1	0.00090	0.00090	0.0151	0.9026852
Residuals	51	3.04825	0.05977		

Table 6-3 ANOVA table for IG_VLS$_{RCT}$

When the interaction plot shown in Figure 6-18 is examined, it can be seen that lowest RPD values are observed when $d = 8$ and $\tau = 0.7$. This value of d is consistent with its value observed in main effect plot.

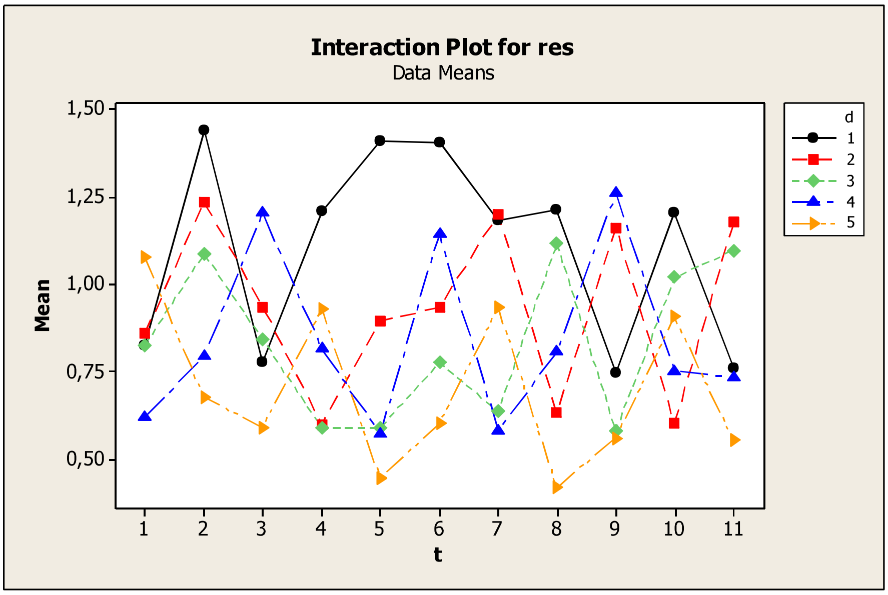

Figure 6-18 Interaction plot of temperature adjustment parameter and destruction size for IG_VLS$_{RCT}$ algorithm

For more detailed analysis for the parameters, interval plots are examined. In Figure 6-19, it is clear that increasing destruction size gives better results. However, intervals are overlapping, meaning that, none of the results have significant superiority to others. When the Figure 6-20 is examined, it is observed that, there is no significant τ value since the intervals overlap again. So, parameters are selected as $d = 4$ and $\tau = 0.5$.

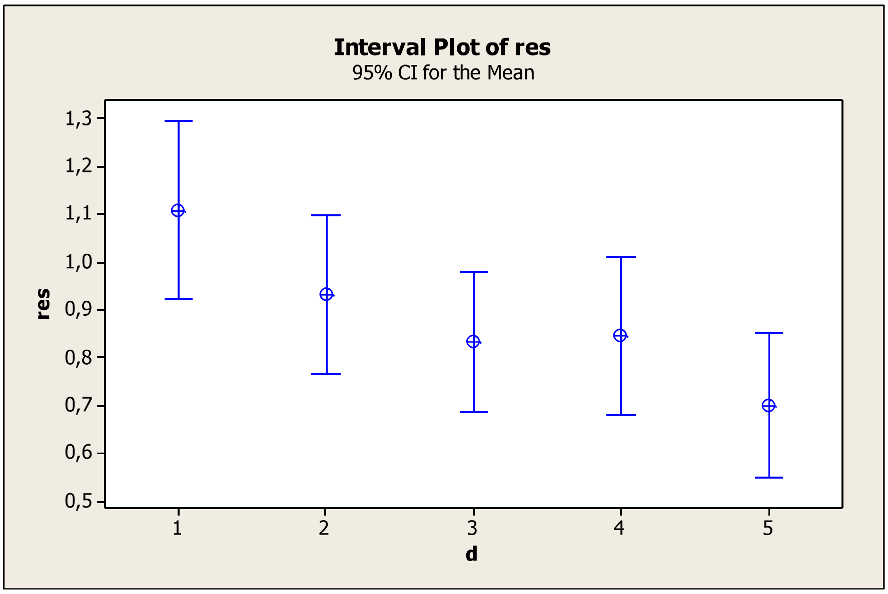

Figure 6-19. Interval plot of d size for IG_VLS$_{RCT}$ algorithm

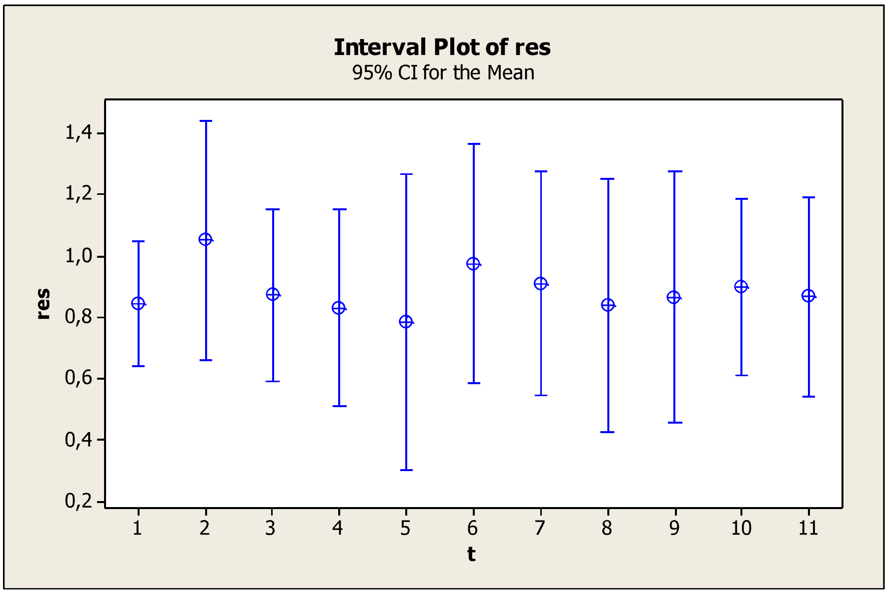

Figure 6-20. Interval plot of t size for IG_VLS$_{RCT}$ algorithm

The same procedure is followed for determining destruction size and temperature adjustment parameter for IG_VLS$_{IKT}$ algorithm. Main effect plots for *d* and τ are presented in Figure 6-21 and Figure 6-22, respectively. It is observed that, $d = 6$ and $d = 8$ yield to lowest RPD values, while, setting $\tau = 0.4$ for temperature adjustment parameter gives the lowest RPD results.

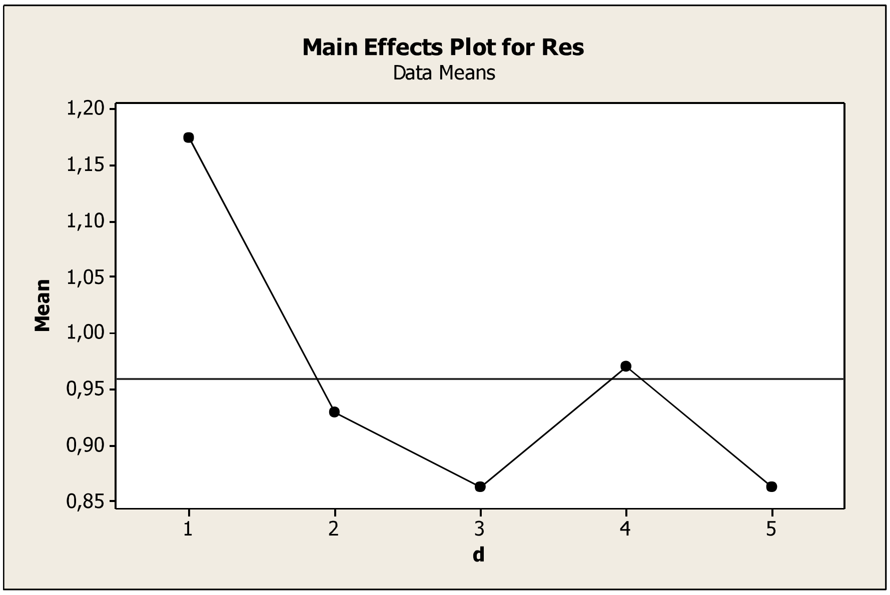

Figure 6-21. Main effect plots of d size for IG_VLS$_{IKT}$ algorithm

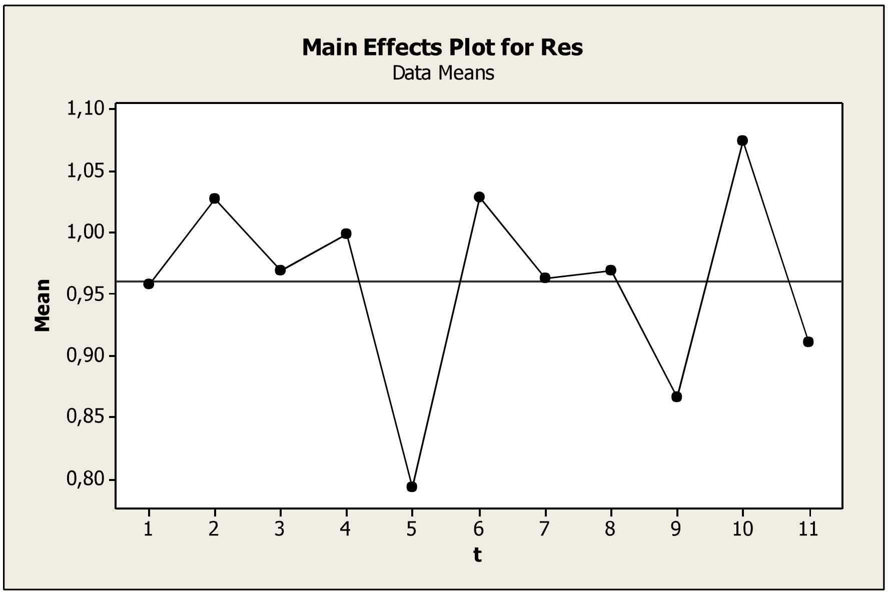

Figure 6-22 Main effect plots of t for IG_VLS$_{IKT}$ algorithm

ANOVA analysis results are given in Table 6-4. The result is the same as it is for IG_RS and IG_ VLS_{RCT} algorithms: only destruction size parameter is statistically significant.

Source	DF	Sum Sq	Mean Sq	F	p
d	1	0.37579	0.37579	7.0001	0.01081
τ	1	0.00271	0.00271	0.0505	0.82304
$d * \tau$	1	0.02262	0.02262	0.4214	0.51914
Residuals	51	2.73788	0.05368		

Table 6-4 ANOVA table for IG_VLS_{IKT}

Interaction plot for destruction size and temperature adjustment parameters for IG_VLS_{IKT} algorithm is given in Figure 6-23. Lowest RPD values are observed when $d = 6$ and $\tau = 0.8$.

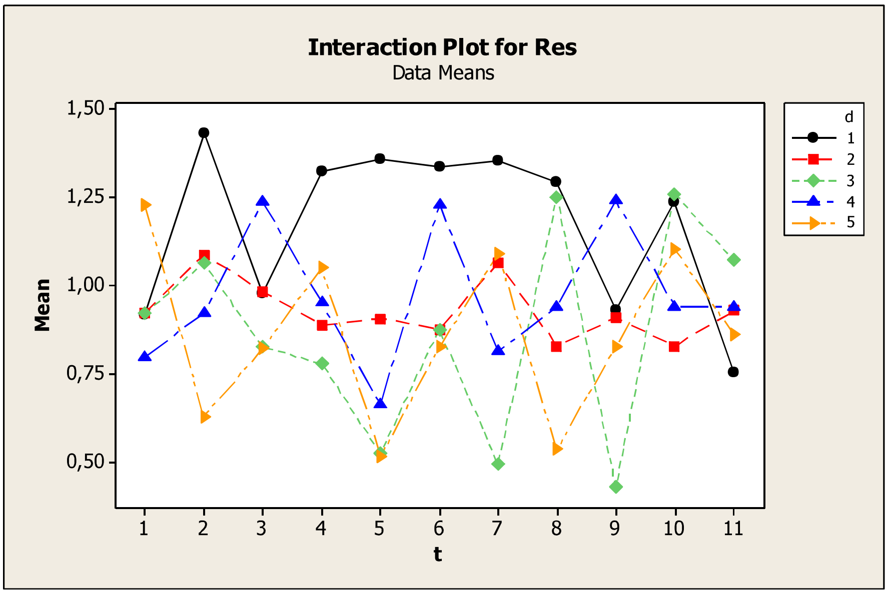

Figure 6-23 Interaction plot of temperature adjustment parameter and destruction size for IG_VLS_{IKT} algorithm

Similarly, interval plots for destruction size and temperature adjustment parameters given in Figure 6-24 and Figure 6-25 are analyzed finally, in order to determine the parameter values. Again, none of the values for d and τ parameters are better than the other for 95% confidence interval, so any of the values can be chosen. So, parameters are selected as $d = 4$ and $\tau = 0.5$.

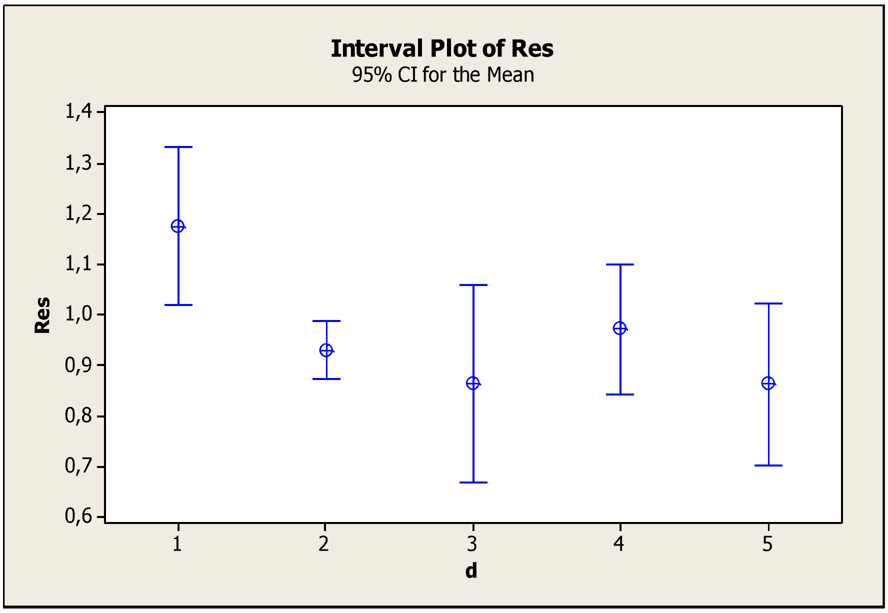

Figure 6-24 Interval plot of destruction size for IG_VLS$_{IKT}$ algorithm

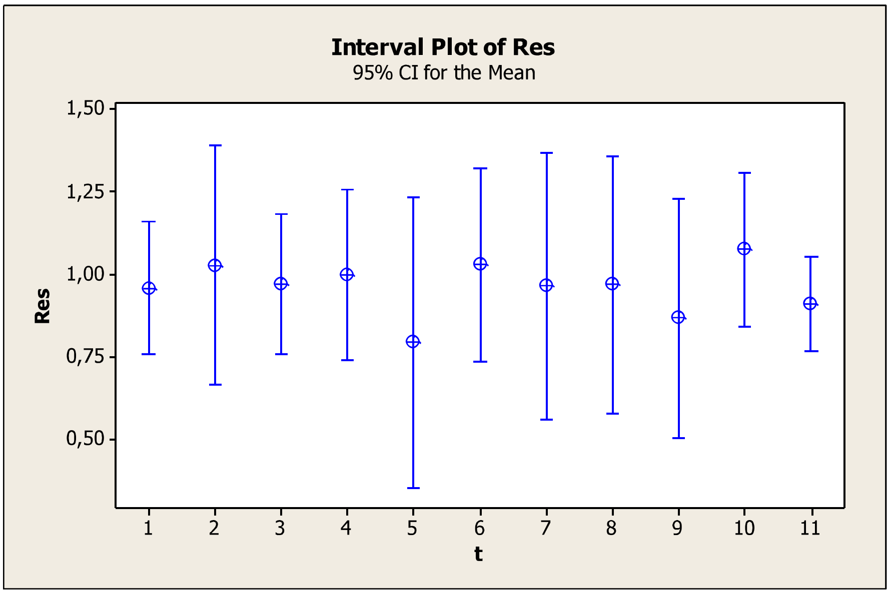

Figure 6-25 Interval plot of temperature adjustment parameter for IG_VLS$_{IKT}$ algorithm

7 COMPUTATIONAL RESULTS

The results of the computational experiments that are carried out are presented in this chapter. The parameters for all algorithms are determined using Design of Experiments approach as detailed in Chapter 6.

7.1 Permutation Flow Shop Problem under Make Span Optimization

Before analyzing the computational results of the proposed algorithms, the impact of the speed-up method that is developed for the lowering the CPU time for the swap neighborhood is analyzed. A simple composite heuristic is developed for this purpose. The algorithm starts with an initial solution generated by NEH_RMB heuristic and applies one pass swap local search with $n(n-1)/2$ objective function evaluations. The experiment is carried out on SDST125 instances. CPU time requirements with and without the developed speed-up method and time saved ratios are given Table 7-1. It can be seen that speed-up method saves up to almost 53% average CPU reduction when carrying out a full swap neighborhood.

Problems	CPU time without speed-up (seconds)	CPU time with speed-up(seconds)	Time saved ratio (%)
20x5	0.0007	0.0005	68.44
20x10	0.0011	0.0007	58.74
20x20	0.0022	0.0014	64.23
50x5	0.0085	0.0041	48.41
50x10	0.0166	0.0088	53.07
50x20	0.0338	0.0182	53.92
100x5	0.0710	0.0343	48.31
100x10	0.1376	0.0711	51.65
100x20	0.3097	0.1490	48.10
200x10	1.2676	0.5954	46.97
200x20	2.8035	1.3752	49.05
500x20	58.9363	26.5107	44.98
Average	5.2991	2.3974	**52.99**

Table 7-1. The impact of the speed-up method on CPU times on SDST125 instances

The proposed algorithms were executed on 480 benchmark instances that are available from http://www.upv.es/gio/rruiz. Ten runs were conducted for each problem instance and the performance parameter which is average percentage deviation (RPD) is calculated as follows:

$$RPD = \sum_{i=1}^{R} \frac{\left(\frac{(H_i - BKS)x100}{BKS}\right)}{R} \qquad (14)$$

where H_i is the makespan value generated in the ith replication of the IG algorithms and BKS is the best makespan values achieved so far in the literature, which are also provided in http://www.upv.es/gio/rruiz along with the problem instances. R is the total number of replications.

The previous best performing algorithms for SDST flowshop scheduling problem under makespan criterion are memetic algorithm (MA) with local search (MA_{LS}) and IG_RS with a local search (IG_RS_{LS}) by Ruiz and Stützle (Ruiz & Stützle, 2007). For this reason, these two algorithms will be used in comparing the performances of the developed IG algorithm version. Three different IG versions are developed as mentioned in Chapter 6: IG_VNS1, IG_VNS2 and IG_IJ_{LS}. These algorithms are implemented in Java language. IG_RS_{LS} (Ruiz & Stützle, 2008) algorithm is re-implemented in Java language and is run on the same computer as the other three algorithms to be able make a more fair comparison. The re-implemented IG_RS_{LS} is named as $IG^*\_RS_{LS}$.

The computational results are given Table 7-2 and Table 7-3, together with the results of the proposed IG algorithms. IG_RS_{LS} and MA_{LS} results are taken from (Ruiz & Stützle, 2008). Note that, IG_RS_{LS} and MA_{LS} are coded and run in different computers. Table 7-2 and Table 7-3 give the average results achieved by each algorithm for the stated instance set. The three values separated by slashes correspond to results obtained by three different CPU limits set to $T_{max} = n\ x\ (m/2) x\ t$ milliseconds where t is taken as 30, 60 and 90, respectively. The average of best results for each SDST problem set are shown in bold fonts.

Destruction size and temperature adjustment parameter values are set to $d = 4$ τ=0.4 for $IG^*\_RS_{LS}$ as in (Ruiz & Stützle, 2008). d is set to 6 and τ is set to 0.2 for IG_IJ. These values are determined by an experiment whose details are given in Chapter 6. The third parameter, iteration jumping probability jP, is set to 0.001. All algorithms are executed 10 times for each problem instance. There are 10 different problems in 12 $n \times m$ sets, so average results are calculated over a total of 120 problem instances and 1200 independent runs.

nxm	MA_{LS}	IG_RS_{LS}	$IG^{*}\_RS_{LS}$	IG_VNS1	IG_VNS2	IG_IJ_{LS}
SDST10						
20x5	0.12/0.10/0.08	0.08/0.05/0.04	0.02 / 0.01 / 0.01	0.02 / 0.02 / 0.01	0.02 / 0.02 / 0.01	0.02 / 0.02 / 0.02
20x10	0.13/0.13/0.13	0.08/0.05/0.04	0.04 / 0.01 / 0.01	0.04 / 0.01 / 0.01	0.04 / 0.01 / 0.01	0.01 / 0.01 / 0.00
20x20	0.14/0.09/0.10	0.07/0.05/0.04	0.03 / 0.01 / 0.00	0.03 / 0.03 / 0.02	0.03 / 0.02 / 0.02	0.04 / 0.03 / 0.02
50x5	0.43/0.31/0.30	0.37/0.32/0.27	0.24 / 0.18 / 0.16	0.31 / 0.24 / 0.20	0.34 / 0.27 / 0.25	0.28 / 0.21 / 0.20
50x10	1.12/0.83/0.81	0.76/0.60/0.53	0.41 / 0.26 / 0.19	0.57 / 0.39 / 0.30	0.67 / 0.48 / 0.37	0.44 / 0.32 / 0.19
50x20	1.16/0.96/0.82	0.91/0.64/0.60	0.61 / 0.40 / 0.28	0.82 / 0.57 / 0.43	0.9 0/ 0.66 / 0.48	0.56 / 0.33 / 0.20
100x5	0.54/0.40/0.31	0.43/0.38/0.33	0.30 / 0.24 / 0.20	0.38 / 0.31 / 0.27	0.50 / 0.39 / 0.36	0.31 / 0.24 / 0.21
100x10	0.78/0.60/0.48	0.61/0.44/0.38	0.27 / 0.13 / 0.06	0.42 / 0.26 / 0.18	0.58 / 0.43 / 0.34	0.30 / 0.16 / 0.09
100x20	1.27/0.97/0.82	0.88/0.71/0.54	0.47 / 0.27 / 0.15	0.78 / 0.52 / 0.37	0.98 / 0.72 / 0.57	0.42 / 0.23 / 0.12
200x10	0.79/0.61/0.48	0.58/0.43/0.32	0.26 / 0.13 / 0.05	0.50 / 0.34 / 0.25	0.69 / 0.53 / 0.43	0.27 / 0.15 / 0.06
200x20	1.11/0.87/0.76	0.79/0.53/0.38	0.43 / 0.23 / 0.10	0.69 / 0.51 / 0.39	1.04 / 0.78 / 0.65	0.37 / 0.23 / 0.07
500x20	0.69/0.54/0.43	0.46/0.31/0.21	0.14 / -0.01 / -0.08	0.39 / 0.27 / 0.20	0.69 / 0.64 / 0.57	0.14 / 0.03 / -0.09
Avg	0.69/0.53/0.46	0.50/0.38/0.31	**0.27 / 0.15 / 0.09**	0.41 / 0.289 / 0.21	0.54 / 0.41 / 0.33	**0.26 / 0.16 / 0.09**
SDST50						
20x5	0.37/0.35/0.30	0.26/0.18/0.10	0.18 / 0.11 / 0.08	0.13 / 0.07 / 0.05	0.12 / 0.05 / 0.04	0.10 / 0.07 / 0.07
20x10	0.41/0.31/0.32	0.28/0.20/0.19	0.20 / 0.12 / 0.09	0.15 / 0.09 / 0.06	0.15 / 0.09 / 0.06	0.15 / 0.08 / 0.07
20x20	0.20/0.16/0.16	0.1/0.09/0.07	0.09 / 0.06 / 0.03	0.07 / 0.04 / 0.03	0.09 / 0.04 / 0.04	0.07 / 0.03 / 0.02
50x5	1.79/1.39/1.13	1.41/1.13/1.04	1.15 / 0.91 / 0.78	1.25 / 0.97 / 0.80	1.46 / 1.16 / 0.99	1.08 / 0.83 / 0.69
50x10	1.49/1.24/1.08	1.33/1.17/0.92	1.00 / 0.77 / 0.65	1.14 / 0.80 / 0.66	1.21 / 0.93 / 0.79	0.91 / 0.65 / 0.55
50x20	1.33/1.07/0.89	1.16/0.93/0.82	0.98 / 0.75 / 0.62	1.14 / 0.82 / 0.67	1.18 / 0.91 / 0.81	0.93 / 0.70 / 0.60
100x5	2.23/1.72/1.38	1.51/1.27/1.09	1.13 / 0.82 / 0.67	1.73 / 1.33 / 1.08	2.08 / 1.58 / 1.35	1.22 / 0.83 / 0.65
100x10	1.84/1.53/1.21	1.37/1.04/0.88	0.91 / 0.56 / 0.42	1.29 / 0.93 / 0.71	1.65 / 1.26 / 1.04	0.99 / 0.68 / 0.51
100x20	1.73/1.35/1.03	1.29/0.96/0.81	0.68 / 0.41 / 0.28	1.12 / 0.75 / 0.58	1.39 / 1.04 / 0.83	0.73 / 0.43 / 0.29
200x10	1.88/1.43/1.21	1.33/0.88/0.63	0.47 / 0.11 / -0.07	1.19 / 0.81 / 0.58	1.84 / 1.33 / 1.08	0.60 / 0.19 / 0.00
200x20	1.61/1.17/1.02	1.10/0.74/0.53	0.38 / 0.09 / -0.08	0.94 / 0.60 / 0.45	1.44 / 1.05 / 0.84	0.47 / 0.13 / -0.04
500x20	1.23/0.96/0.79	0.86/0.50/0.31	0.22 / -0.08 / -0.26	0.92 / 0.65 / 0.48	1.49 / 1.29 / 1.15	0.27 / -0.08 / -0.26
Avg	1.34/1.06/0.88	1.00/0.76/0.62	**0.61 / 0.39 / 0.27**	0.92 / 0.65 / 0.51	1.17 / 0.89 / 0.75	**0.63 / 0.38 / 0.26**

Table 7-2 Average relative percentage deviations for SDST10 and SDST50 instances

nxm	MA_{LS}	IG_RS_{LS}	$IG^{*}\_RS_{LS}$	IG_VNS1	IG_VNS2	IG_IJ_{LS}
SDST100						
20x5	0.43/0.37/0.39	0.30/0.25/0.17	0.25 / 0.17 / 0.17	0.20 / 0.08 / 0.05	0.23 / 0.13 / 0.09	0.21 / 0.15 / 0.10
20x10	0.31/0.28/0.29	0.35/0.25/0.18	0.31 / 0.22 / 0.19	0.22 / 0.14 / 0.11	0.27 / 0.15 / 0.14	0.20 / 0.15 / 0.13
20x20	0.29/0.26/0.17	0.27/0.18/0.17	0.27 / 0.19 / 0.15	0.16 / 0.08 / 0.04	0.19 / 0.11 / 0.07	0.16 / 0.10 / 0.07
50x5	2.37/2.24/1.99	1.95/1.95/1.82	2.06 / 1.76 / 1.54	2.02 / 1.61 / 1.41	2.36 / 1.89 / 1.74	1.98 / 1.65 / 1.46
50x10	1.98/1.66/1.50	1.57/1.48/1.30	1.60 / 1.30 / 1.20	1.62 / 1.26 / 1.09	1.89 / 1.53 / 1.32	1.57 / 1.29 / 1.15
50x20	1.66/1.35/1.18	1.41/1.28/1.11	1.49 / 1.29 / 1.12	1.41 / 1.09 / 0.91	1.52 / 1.16 / 1.00	1.25 / 1.02 / 0.94
100x5	3.20/2.69/2.16	2.16/1.95/1.63	2.01 / 1.53 / 1.28	2.65 / 2.09 / 1.77	3.20 / 2.47 / 2.11	1.90 / 1.40 / 1.17
100x10	2.26/2.01/1.61	1.61/1.44/1.02	1.36 / 0.87 / 0.67	1.80 / 1.27 / 0.98	2.2 / 1.59 / 1.25	1.16 / 0.76 / 0.59
100x20	2.12/2.03/1.53	1.41/1.35/1.05	1.02 / 0.72 / 0.59	1.42 / 1.01 / 0.74	1.90 / 1.41 / 1.16	1.09 / 0.74 / 0.58
200x10	2.53/2.19/1.77	1.67/1.25/0.92	0.78 / 0.23 / -0.06	1.85 / 1.20 / 0.86	2.79 / 2.09 / 1.69	0.85 / 0.34 / 0.01
200x20	1.93/1.68/1.40	1.26/0.93/0.76	0.63 / 0.14 / -0.07	1.35 / 0.84 / 0.57	2.00 / 1.52 / 1.21	0.68 / 0.29 / 0.05
500x20	1.53/1.35/1.14	0.96/0.73/0.46	0.44 / -0.06 / -0.28	1.29 / 0.87 / 0.65	2.00 / 1.73 / 1.50	0.46 / 0.00 / -0.26
Average	1.72/1.51/1.26	1.24/1.09/0.88	**1.02 / 0.70 / 0.54**	1.33 / 0.96 / 0.76	1.71 / 1.31 / 1.15	**0.96 / 0.66 / 0.50**
SDST125						
20x5	0.67/0.34/0.32	0.46/0.35/0.3	0.28 / 0.16 / 0.13	0.26 / 0.13 / 0.07	0.29 / 0.18 / 0.09	0.30 / 0.23 / 0.17
20x10	0.51/0.42/0.37	0.53/0.41/0.36	0.56 / 0.40 / 0.33	0.39 / 0.24 / 0.16	0.33 / 0.24 / 0.20	0.31 / 0.20 / 0.19
20x20	0.28/0.22/0.24	0.26/0.22/0.19	0.26 / 0.19 / 0.16	0.14 / 0.07 / 0.06	0.13 / 0.09 / 0.06	0.15 / 0.11 / 0.08
50x5	2.97/2.47/1.97	2.37/2.18/2.01	2.33 / 1.91 / 1.73	2.61 / 2.11 / 1.83	2.67 / 2.19 / 1.94	2.01 / 1.77 / 1.56
50x10	2.07/1.78/1.5	1.94/1.67/1.54	1.71 / 1.45 / 1.33	1.99 / 1.55 / 1.26	2.02 / 1.62 / 1.44	1.68 / 1.41 / 1.25
50x20	1.59/1.43/1.26	1.42/1.45/1.18	1.54 / 1.30 / 1.16	1.58 / 1.24 / 1.05	1.76 / 1.41 / 1.22	1.34 / 1.15 / 1.03
100x5	3.55/3.02/2.52	2.41/2.27/1.91	2.44 / 1.78 / 1.52	3.24 / 2.62 / 2.29	3.85 / 2.94 / 2.63	2.34 / 1.80 / 1.47
100x10	2.78/2.37/1.94	2.07/1.65/1.34	1.67 / 1.16 / 0.91	2.24 / 1.58 / 1.26	2.71 / 2.09 / 1.82	1.66 / 1.12 / 0.87
100x20	2.31/1.8/1.50	1.52/1.22/1.00	1.03 / 0.62 / 0.39	1.51 / 1.06 / 0.81	1.91 / 1.39 / 1.17	0.91 / 0.55 / 0.36
200x10	2.73/2.51/2.14	1.79/1.6/1.17	1.13 / 0.52 / 0.19	2.41 / 1.75 / 1.37	3.14 / 2.38 / 1.95	1.07 / 0.43 / 0.13
200x20	2.04/1.74/1.49	1.38/1.06/0.76	0.59 / 0.13 / -0.10	1.54 / 1.04 / 0.76	2.15 / 1.58 / 1.28	0.65 / 0.13 / -0.13
500x20	1.7/1.53/1.23	1.08/0.83/0.52	0.61 / 0.05 / -0.29	1.66 / 1.31 / 0.98	2.32 / 2.10 / 1.88	0.59 / 0.09 / -0.23
Average	1.93/1.64/1.37	1.44/1.24/1.02	**1.18 / 0.81 / 0.62**	1.63 / 1.22 / 0.99	1.94 / 1.51 / 1.31	**1.08 / 0.75 / 0.56**

Table 7-3 Average relative percentage deviations for SDST100 and SDST125 instances

It is obvious from Table 7-2 and Table 7-3 that, better overall average results are achieved with higher t values, as expected. On the algorithm comparison side, it is clear that IG*_RS$_{LS}$ and IG_IJ algorithms outperform IG_VNS1 and IG_VNS2, MA$_{LS}$ and IG_RS$_{LS}$ with significant margins. Among the best two performing algorithms, IG_IJ achieves slightly better performance, especially for harder SDST100 and SDST125 instances.

Line plot of the RPD values for different SDST sets (SDST10, SDST50, SDST100 and SDST125) are presented in Figure 7-1, Figure 7-2, Figure 7-3 and Figure 7-4 respectively.

In Figure 7-1, it is clear that IG*_RS$_{LS}$ and IG_IJ$_{LS}$ outperform other algorithms. And also IG_VNS1 and IG_VNS2 show better performance than MA$_{LS}$ and IG_RS$_{LS}$. But it is hard to make a comparison between performances of IG*_RS$_{LS}$ and IG_IJ$_{LS}$. Average RPD values are very close for $t = 30$ and $t = 90$. But for $t = 60$ IG*_RS$_{LS}$ has slightly smaller RPD values, but it does not have much statistical significance.

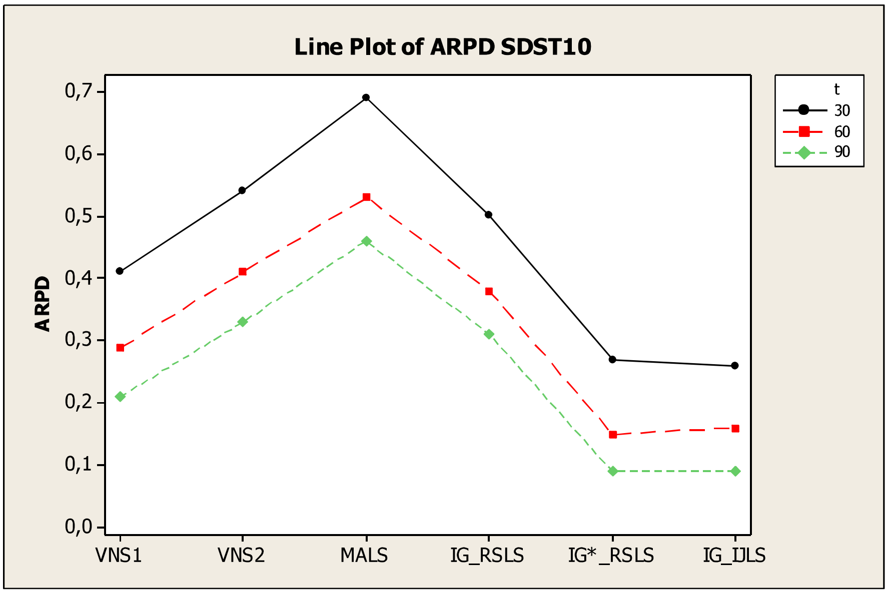

Figure 7-1 Plot of average percentage deviations for SDST10 instances

The results presented in Figure 7-2 are very similar to the outcomes of Figure 7-1. IG*_RS$_{LS}$ and IG_IJ$_{LS}$ outperform the other algorithms but there is no significant difference between these two algorithms. IG*_RS$_{LS}$ shows slightly better performance for $t = 30$. But it is hard to make such a verdict for $t = 60$ and $t = 90$.

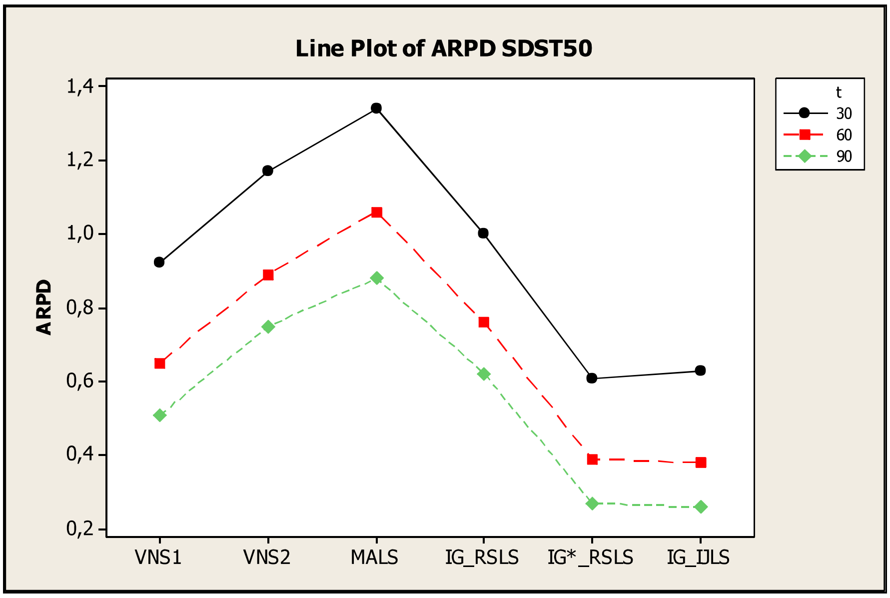

Figure 7-2 Plot of average percentage deviations for SDST50 instances

However, Figure 7-3 and Figure 7-4 show different characteristics. Both algorithms IG*_RS$_{LS}$ and IG_IJ$_{LS}$ again outperform the rest of the algorithms. IG_IJ$_{LS}$ has slightly better performance than IG*_RS$_{LS}$.

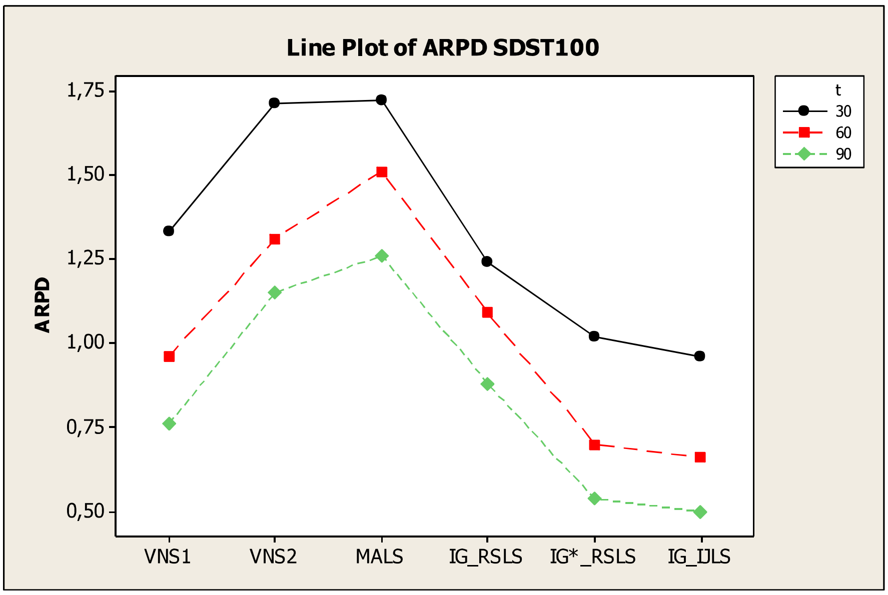

Figure 7-3 Plot of average percentage deviations for SDST100 instances

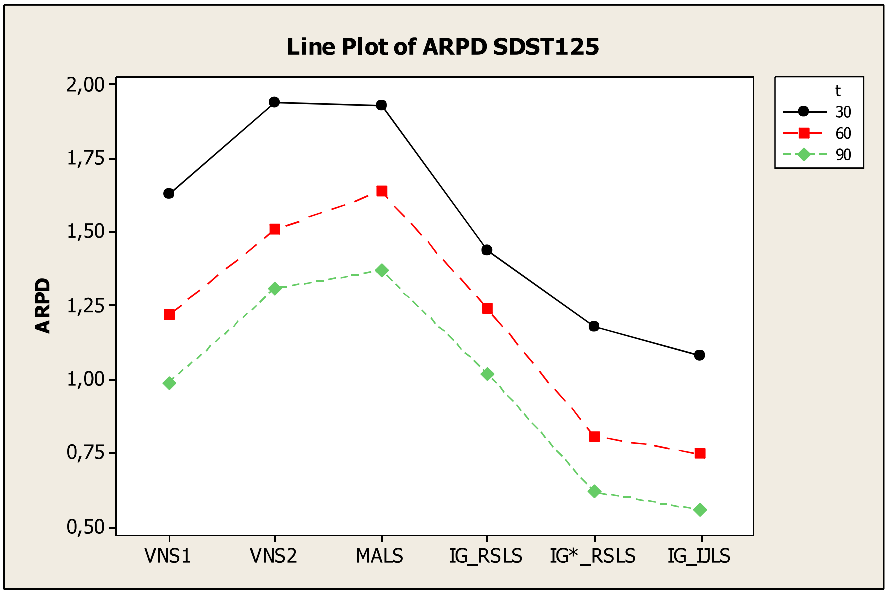

Figure 7-4 Plot of average percentage deviations for SDST125 instances

In order to make a more detailed performance analysis, Figure 7-5 to Figure 7-8 present the plots of average RPD values for each algorithm for the maximum CPU limit $t = 90$.

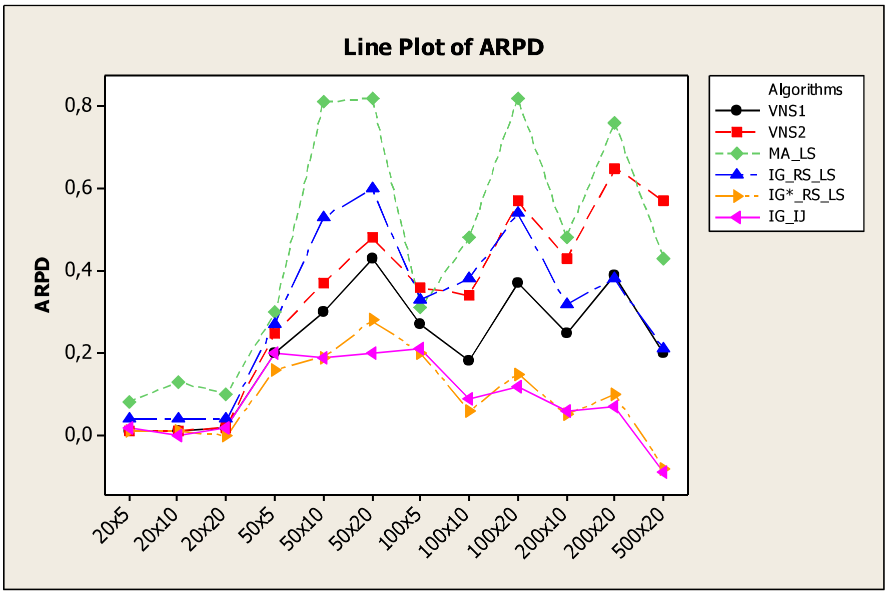

Figure 7-5 Plot of average percentage deviations for SDST10 instances with t=90

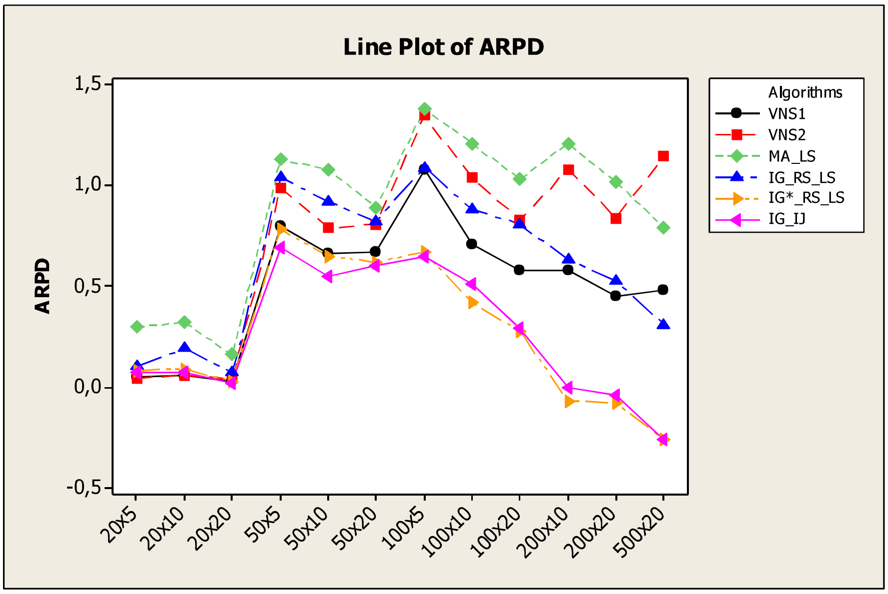

Figure 7-6 Plot of average percentage deviations for SDST50 instances with t=90

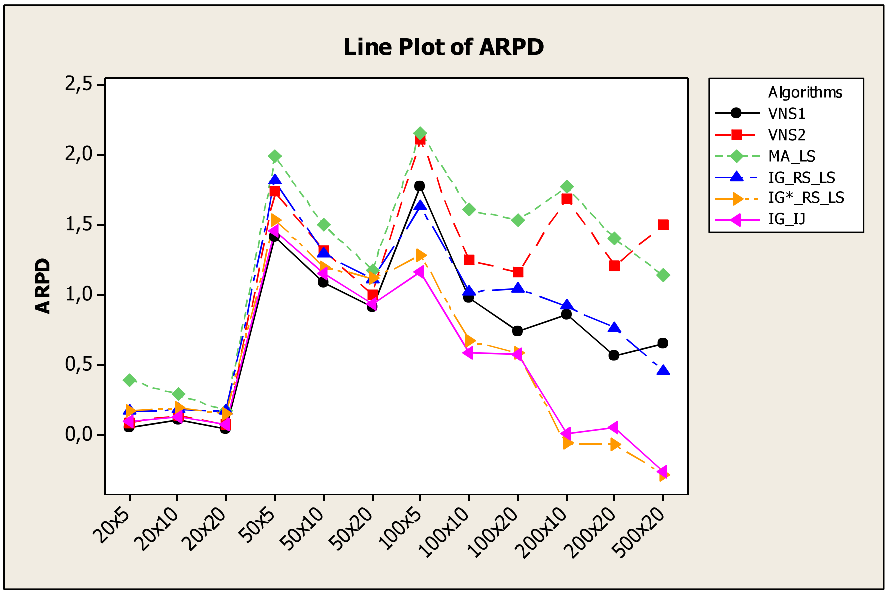

Figure 7-7 Plot of average percentage deviations for SDST100 instances with t=90

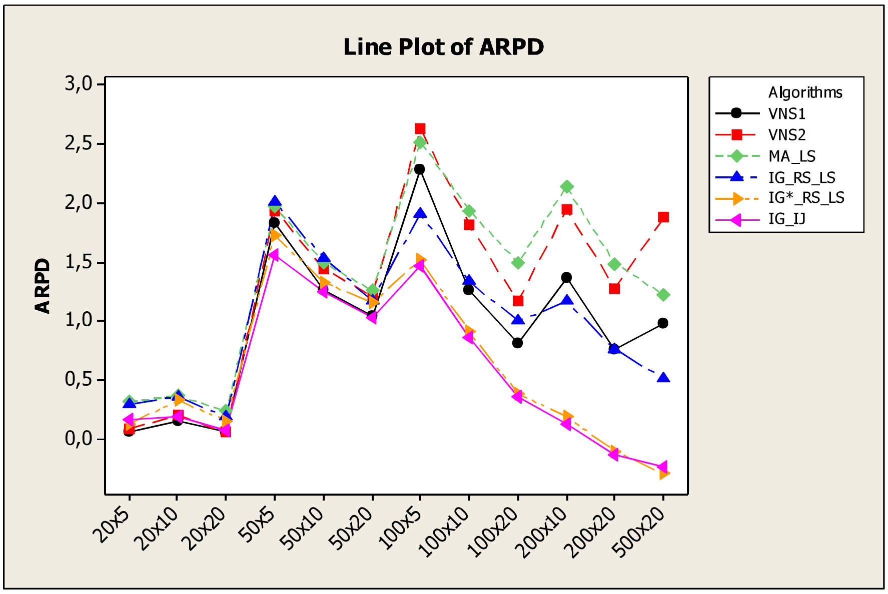

Figure 7-8 Plot of average percentage deviations for SDST125 instances with t=90

From the above figures, again it is obvious that IG*_RS$_{LS}$ and IG_JI$_{LS}$ show much better performance than rest of the algorithms. Performances of the IG*_RS$_{LS}$ and IG_IJ$_{LS}$ get better for larger instances like 100x5, 100x10 ,100x20, 200x20 and 500x20. Another observation is that, the performance of IG*_RS$_{LS}$ and IG_IJ$_{LS}$ algorithms are very similar, there is no clear difference between them.

Interval plots of average RPD values for IG*_RS$_{LS}$, IG_IJ$_{LS}$, IG_VNS1 and IG_VNS2 algorithms for four different SDST groups are presented in Figure 7-9 to Figure 7-11 for different t values. It can be seen from Figure 7-9 to Figure 7-11 that, IG*_RS$_{LS}$ and IG_IJ$_{LS}$ algorithms are equivalent in performance since their confidence intervals coincide for SDST10, SDST50, SDST100 and SDST125 instances.

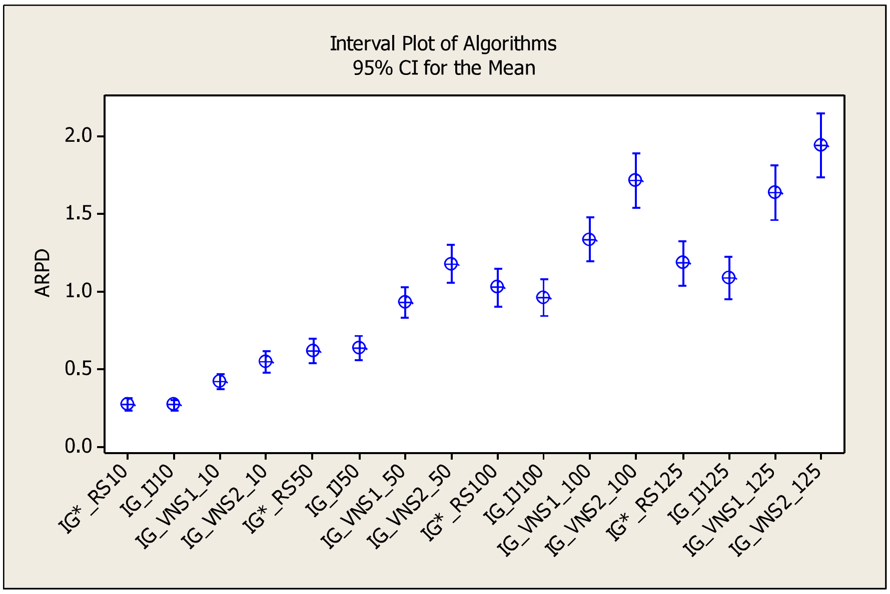

Figure 7-9 Interval plot of algorithms for t=30

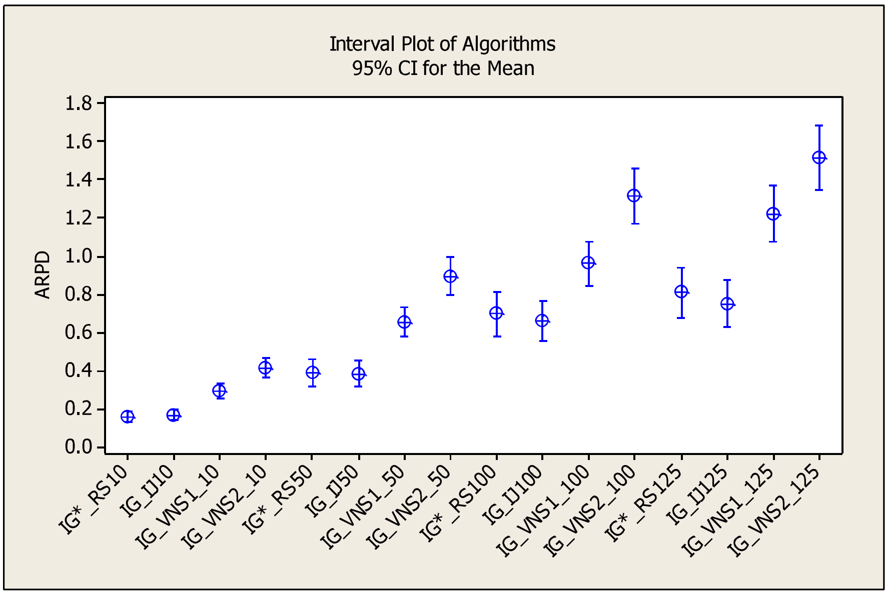

Figure 7-10 Interval plot of algorithms for *t*=60

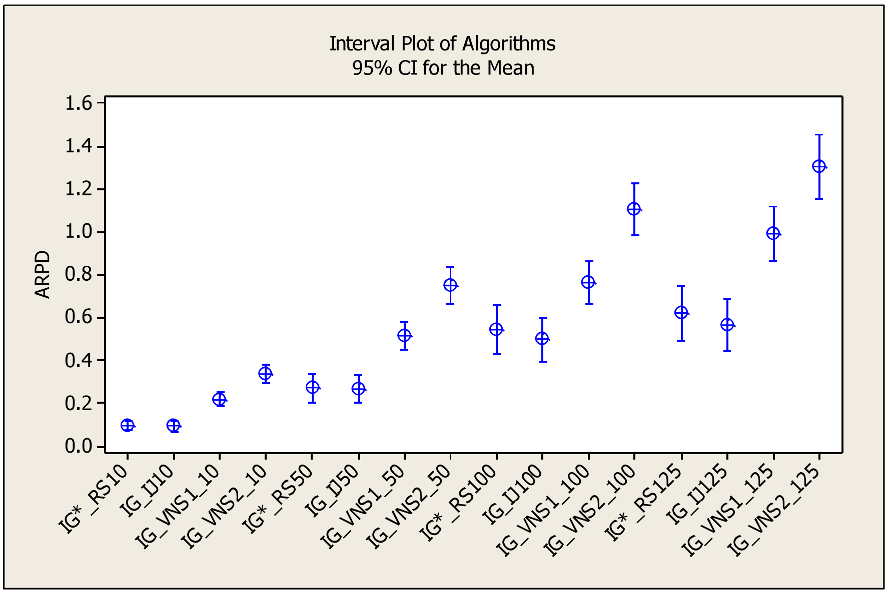

Figure 7-11 Interval plot of algorithms for t=90

However, IG*_RS$_{LS}$ and IG_IJ$_{LS}$ algorithms are statistically better than IG_VNS1 and IG_VNS2 algorithms since their confidence intervals do not coincide. When two VNS algorithms are compared, it is observed that, IG_VNS1 is statistically better than IG_VNS2 algorithm, suggesting that the first neighborhood should be taken as insertion neighborhood structure.

Finally, the new best known fitness values are listed in Appendix A-1. Ultimately, 246 out of 480 instances are improved by the proposed algorithms.

7.2 Permutation Flow Shop Problem under Total Flow Time Optimization

As mentioned before, permutation flow shop scheduling problem with sequence dependent set up times under total flow time minimization is a new problem, which has not been studied in the literature before. Five different IG algorithm versions are designed and implemented for this problem. The first algorithm is IG_RS$_{LS}$ algorithm by (Ruiz & Stützle, 2008) and the second one is IG_VLS$_{RCT}$ which uses the VLS algorithm (Ribas, Companys, & Tort-Martorell, 2015) in local search step. Three new algorithms that use different versions of the VLS algorithm are also developed.

IG_VLS$_{RCT}$_1 uses insertion neighborhood in LS1 and swap neighborhood in LS2, while, VLS$_{RCT}$_2 uses swap neighborhood in in LS1 and insertion neighborhood inLS2. In addition, both versions use first improvement pivoting rule instead of best improvement pivoting rule in VLS algorithm. IG_VLS$_{IKT}$ uses the VLS$_{IKT}$ algorithm in the local search step, whose details are given in Section 6.

All algorithms are coded in Java programming language and are run on the same computers that are used for makespan minimization problem. The proposed algorithms are executed on the same benchmark suite used for makespan minimization criterion. The termination condition is again set to $T_{max} = n\ x\ (m/2) x\ t$ milliseconds where t is taken as 30, 60 and 90 respectively. Five independent runs are conducted for each problem instance and the average percentage deviation is calculated using Eq. (13).

The computational results that summarize the average relative percent deviations for all algorithms are given in Table 7-4 and Table 7-5. Again, different SDST groups are listed separately. The three values separated by slashes correspond to results obtained by using three different CPU limits.

nxm	IG_RS$_{LS}$	IG_VLS$_{RCT}$	IG_VLS$_{RCT}$_1	IG_VLS$_{RCT}$_2	IG_VLS$_{IKT}$
SDST10					
20x5	0,00 / 0,00 / 0,00	0,01 / 0,00 / 0,00	0,02 / 0,01 / 0,00	0,00 / 0,00 / 0,00	0,01 / 0,00 / 0,00
20x10	0,00 / 0,00 / 0,00	0,02 / 0,02 / 0,01	0,07 / 0,05 / 0,03	0,03 / 0,02 / 0,00	0,01 / 0,01 / 0,01
20x20	0,01 / 0,01 / 0,01	0,04 / 0,04 / 0,04	0,12 / 0,09 / 0,07	0,05 / 0,04 / 0,04	0,04 / 0,04 / 0,03
50x5	0,78 / 0,54 / 0,44	0,57 / 0,45 / 0,37	0,97 / 0,73 / 0,64	0,53 / 0,38 / 0,33	0,51 / 0,41 / 0,31
50x10	0,73 / 0,48 / 0,41	0,79 / 0,6 / 0,51	1,40 / 1,19 / 1,04	0,69 / 0,52 / 0,43	0,77 / 0,57 / 0,47
50x20	0,61 / 0,36 / 0,30	0,8 / 0,66 / 0,56	1,24 / 1,07 / 0,94	0,75 / 0,54 / 0,44	0,73 / 0,61 / 0,53
100x5	0,94 / 0,69 / 0,56	0,70 / 0,51 / 0,40	1,00 / 0,78 / 0,68	0,62 / 0,44 / 0,33	0,59 / 0,36 / 0,26
100x10	0,95 / 0,61 / 0,47	0,97 / 0,74 / 0,59	1,39 / 1,16 / 0,99	0,86 / 0,61 / 0,48	0,88 / 0,59 / 0,48
100x20	0,86 / 0,53 / 0,38	1,15 / 0,85 / 0,67	1,58 / 1,34 / 1,21	0,99 / 0,72 / 0,57	0,91 / 0,65 / 0,51
200x10	0,79 / 0,52 / 0,38	0,82 / 0,6 / 0,48	0,93 / 0,8 / 0,68	0,8 / 0,58 / 0,43	0,75 / 0,56 / 0,43
200x20	0,86 / 0,48 / 0,33	0,91 / 0,73 / 0,57	1,12 / 0,92 / 0,81	0,88 / 0,69 / 0,57	0,77 / 0,55 / 0,41
500x20	0,32 / 0,31 / 0,27	0,29 / 0,29 / 0,29	0,7 / 0,58 / 0,52	0,29 / 0,29 / 0,28	0,27 / 0,27 / 0,23
Avg.	**0,57 / 0,38 / 0,30**	0,59 / 0,46 / 0,37	0,88 / 0,73 / 0,63	0,54 / 0,40 / 0,32	0,52 / 0,39 / 0,31
SDST50					
20x5	0,03 / 0,02 / 0,01	0,14 / 0,11 / 0,08	0,29 / 0,23 / 0,21	0,15 / 0,13 / 0,09	0,14 / 0,09 / 0,08
20x10	0,02 / 0,02 / 0,01	0,03 / 0,01 / 0,01	0,19 / 0,16 / 0,11	0,02 / 0,00 / 0,00	0,03 / 0,02 / 0,01
20x20	0,00 / 0,00 / 0,00	0,02 / 0,01 / 0,01	0,17 / 0,12 / 0,08	0,01 / 0,01 / 0,01	0,01 / 0,01 / 0,01
50x5	1,02 / 0,68 / 0,49	1,28 / 0,95 / 0,78	2,33 / 1,9 / 1,73	1,25 / 1,00 / 0,83	1,17 / 0,97 / 0,84
50x10	0,87 / 0,59 / 0,48	1,16 / 0,88 / 0,77	2,0 / 1,75 / 1,59	1,18 / 0,89 / 0,75	1,02 / 0,86 / 0,75
50x20	0,62 / 0,51 / 0,45	0,96 / 0,74 / 0,6	1,47 / 1,28 / 1,2	0,91 / 0,81 / 0,67	0,87 / 0,72 / 0,59
100x5	1,78 / 1,22 / 0,94	1,85 / 1,39 / 1,17	2,92 / 2,44 / 2,22	1,84 / 1,34 / 1,05	1,8 / 1,31 / 1,08
100x10	1,33 / 0,90 / 0,67	1,60 / 1,34 / 1,11	2,46 / 2,1 / 1,91	1,41 / 1,02 / 0,81	1,43 / 1,1 / 0,92
100x20	1,05 / 0,68 / 0,54	1,12 / 0,91 / 0,74	1,95 / 1,54 / 1,39	1,08 / 0,82 / 0,62	0,98 / 0,75 / 0,57
200x10	1,30 / 0,92 / 0,72	1,48 / 1,12 / 0,93	2,28 / 1,93 / 1,71	1,46 / 1,06 / 0,87	1,35 / 1,06 / 0,85
200x20	0,94 / 0,65 / 0,48	1,08 / 0,84 / 0,66	1,7 / 1,39 / 1,17	1,03 / 0,7 / 0,54	1,03 / 0,71 / 0,54
500x20	0,51 / 0,46 / 0,38	0,46 / 0,44 / 0,44	1,15 / 0,93 / 0,93	0,46 / 0,45 / 0,42	0,42 / 0,38 / 0,32
Avg	**0,79 / 0,55 / 0,43**	0,93 / 0,73 / 0,61	1,57 / 1,31 / 1,19	0,90 / 0,69 / 0,56	0,86 / 0,66 / 0,55

Table 7-4 Average relative percentage deviations for SDST10 and SDST50 instances

nxm	*IG_RS*$_{LS}$	IG_VLS$_{RCT}$	IG_VLS$_{RCT}$_1	IG_VLS$_{RCT}$_2	IG_VLS$_{IKT}$
SDST100					
20x5	0,07 / 0,03 / 0,03	0,29 / 0,20 / 0,10	0,57 / 0,43 / 0,42	0,17 / 0,15 / 0,15	0,19 / 0,11 / 0,1
20x10	0,04 / 0,02 / 0,02	0,13 / 0,10 / 0,05	0,29 / 0,22 / 0,17	0,12 / 0,11 / 0,08	0,11 / 0,08 / 0,08
20x20	0,04 / 0,04 / 0,04	0,15 / 0,09 / 0,08	0,28 / 0,24 / 0,21	0,13 / 0,1 / 0,09	0,11 / 0,08 / 0,06
50x5	1,63 / 1,15 / 1,05	2,15 / 1,78 / 1,60	3,62 / 3,11 / 2,83	1,97 / 1,37 / 1,18	1,98 / 1,55 / 1,38
50x10	1,22 / 0,80 / 0,68	1,57 / 1,22 / 1,10	2,58 / 2,18 / 2,04	1,48 / 1,17 / 0,99	1,36 / 1,13 / 1,01
50x20	0,99 / 0,72 / 0,65	1,31 / 1,14 / 1,03	1,94 / 1,70 / 1,53	1,24 / 1,01 / 0,87	1,17 / 1,03 / 0,9
100x5	2,53 / 1,46 / 1,29	2,86 / 2,17 / 1,88	4,06 / 3,52 / 3,27	2,41 / 1,67 / 1,37	2,51 / 2,01 / 1,67
100x10	1,64 / 1,01 / 0,82	1,98 / 1,53 / 1,26	3,05 / 2,64 / 2,36	1,82 / 1,32 / 0,94	1,82 / 1,32 / 1,1
100x20	1,27 / 0,84 / 0,71	1,64 / 1,30 / 1,10	2,5 / 2,05 / 1,87	1,46 / 1,11 / 0,94	1,4 / 1,09 / 0,95
200x10	1,45 / 1,00 / 0,81	1,79 / 1,32 / 1,04	2,52 / 2,06 / 1,82	1,60 / 1,09 / 0,81	1,61 / 1,14 / 0,86
200x20	1,26 / 0,81 / 0,62	1,27 / 0,96 / 0,79	1,94 / 1,60 / 1,43	1,16 / 0,88 / 0,58	1,11 / 0,87 / 0,67
500x20	0,57 / 0,52 / 0,48	0,61 / 0,60 / 0,56	1,18 / 0,96 / 0,96	0,61 / 0,58 / 0,50	0,53 / 0,49 / 0,43
Avg.	**1,06 / 0,70 / 0,60**	1,31 / 1,03 / 0,88	2,04 / 1,72 / 1,58	1,18 / 0,88 / 0,71	1,16 / 0,91 / 0,77
SDST125					
20x5	0,14 / 0,12 / 0,11	0,27 / 0,20 / 0,19	0,84 / 0,60 / 0,47	0,23 / 0,22 / 0,22	0,22 / 0,19 / 0,18
20x10	0,04 / 0,03 / 0,02	0,11 / 0,06 / 0,06	0,33 / 0,29 / 0,23	0,10 / 0,08 / 0,06	0,09 / 0,06 / 0,06
20x20	0,01 / 0,01 / 0,01	0,08 / 0,05 / 0,03	0,28 / 0,21 / 0,20	0,07 / 0,05 / 0,04	0,08 / 0,05 / 0,03
50x5	1,71 / 1,25 / 0,92	2,47 / 2,00 / 1,84	4,08 / 3,28 / 2,90	2,10 / 1,57 / 1,39	2,32 / 1,71 / 1,54
50x10	1,39 / 0,92 / 0,78	1,87 / 1,41 / 1,20	2,88 / 2,49 / 2,29	1,76 / 1,37 / 1,19	1,61 / 1,27 / 1,07
50x20	1,04 / 0,84 / 0,74	1,28 / 1,09 / 0,95	1,97 / 1,74 / 1,56	1,18 / 0,95 / 0,83	1,16 / 0,99 / 0,92
100x5	2,56 / 1,83 / 1,36	2,86 / 2,15 / 1,69	4,29 / 3,67 / 3,39	2,64 / 1,85 / 1,48	2,40 / 1,72 / 1,27
100x10	2,04 / 1,49 / 1,14	2,46 / 1,86 / 1,63	3,41 / 2,91 / 2,71	2,08 / 1,54 / 1,21	2,09 / 1,53 / 1,31
100x20	1,28 / 0,94 / 0,72	1,47 / 1,16 / 0,88	2,22 / 1,86 / 1,69	1,37 / 0,91 / 0,71	1,18 / 0,88 / 0,67
200x10	1,71 / 1,17 / 0,84	1,99 / 1,52 / 1,23	2,56 / 2,16 / 1,91	1,87 / 1,40 / 1,03	1,68 / 1,15 / 0,82
200x20	1,39 / 1,01 / 0,78	1,58 / 1,17 / 0,95	2,19 / 1,84 / 1,63	1,48 / 0,97 / 0,70	1,40 / 1,02 / 0,80
500x20	0,71 / 0,63 / 0,57	0,54 / 0,53 / 0,47	1,19 / 1,15 / 0,97	0,54 / 0,54 / 0,47	0,57 / 0,50 / 0,45
Avg.	**1,17 / 0,85 / 0,67**	1,42 / 1,10 / 0,93	2,19 / 1,85 / 1,66	1,28 / 0,96 / 0,78	1,23 / 0,92 / 0,76

Table 7-5 Average relative percentage deviations for SDST100 and SDST125 instances

IG_RS$_{LS}$ achieved the best results as observed from the tables. Although VLS$_{RCT}$_2 and VLS$_{IKT}$ achieved results that are competitive or even slightly better than IG_RS$_{LS}$ for SDST_10, IG_RS$_{LS}$ is clearly the best performing algorithm for the other three SDST groups. IG_RS$_{LS}$ uses only the insertion neighborhood, therefore, it may be concluded that insertion neighborhood based local search is better for total flow time minimization in the presence of especially large SDST.

Line plots of RPD values for SDST10, SDST50, SDST100 and SDST125 instances are presented in Figure 7-12, Figure 7-13, Figure 7-14 and Figure 7-15, respectively. In Figure 7-12, Average RPD values for SDST10 instances which have the smallest sequence set up time with values uniformly distributed between 1 and 9, are presented. It is clear that, IG_VLS$_{RCT}$_1 has got the worst performance for this instance set and the other instances sets also. When VLS$_{RCT}$, VLS$_{RCT}$_2 and VLS$_{IKT}$ algorithms are compared, it is observed that, none of the algorithms perform significantly better than the others. The difference between the algorithms are less than 0.01%.

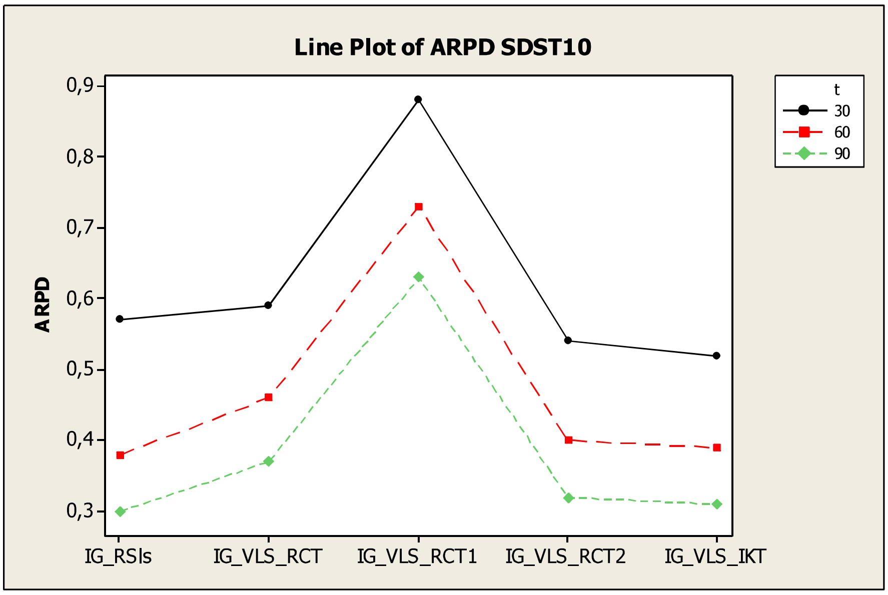

Figure 7-12 Plot of average percentage deviations for SDST10 instances

Average RPD values are presented for SDST50 instances in Figure 7-13. SDST50 instances have sequence dependent setup times uniformly distributed between 1 and 49. Increasing effect of sequence dependent setup times can be observed in the results for these instances. In original Taillard instances without setup times, job processing times are also uniformly distributed between 1 and 99, as mentioned before. Addition of SDST uniformly distributed between 1 and 49 begins to effect the performance of the algorithms. IG_RS$_{LS}$ shows slightly better performance than other algorithms. IG_VLS$_{RST}$_1's performance is worst again.

Performance of IG_VLS$_{RCT}$_2 and IG_VLS$_{IKT}$ are almost the same, whereas IG_VLS$_{RCT}$ is slightly worse than these two.

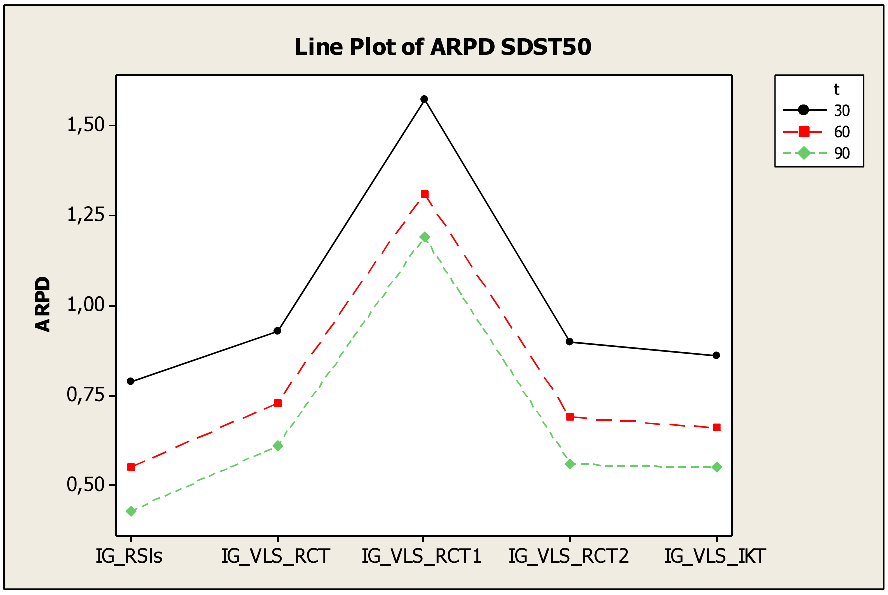

Figure 7-13 Plot of average percentage deviations for SDST50 instances

Average RPD values for SDST100 and SDST125 instances are presented in Figure 7-14 and Figure 7-15. Effect of the sequence dependent setup times is much clearer for these instances. SDST are the same or even larger than processing times in these instances. It means that sequence setup time plays an important role. Obviously, IG_RS$_{LS}$ outperforms other algorithms, while IG_VLS$_{RCT}$_1 is the worst algorithm again. When the other algorithm performances are inspected, it is seen that, IG_VLS$_{RCT}$ is slightly worse than IG_VLS$_{RCT}$_2 and IG_VLS$_{IKT}$.

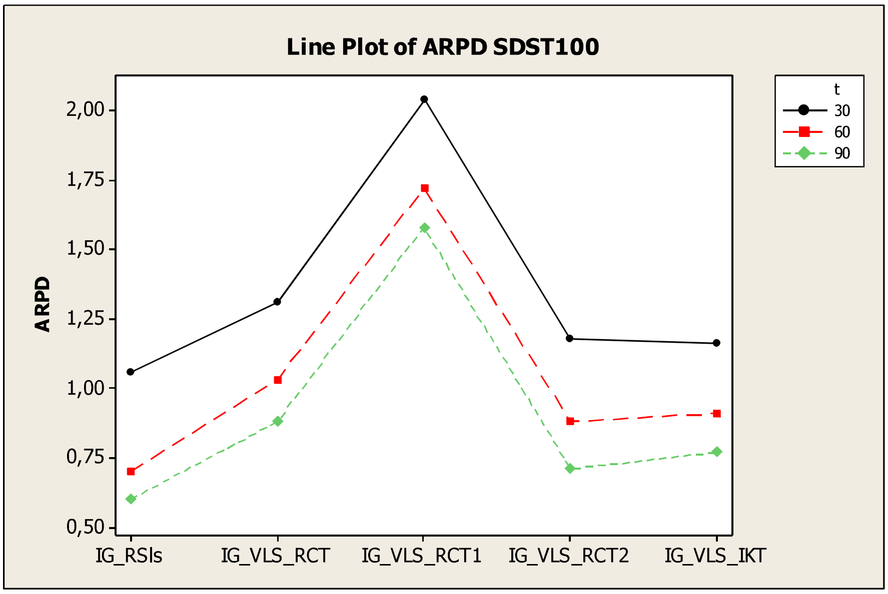

Figure 7-14 Plot of average percentage deviations for SDST100 instances

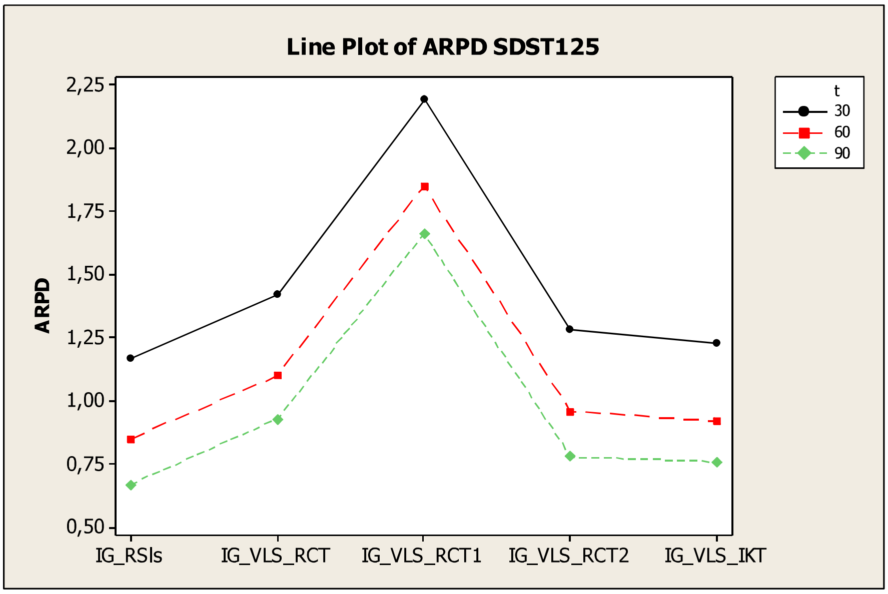

Figure 7-15 Plot of average percentage deviations for SDST125 instances

Line plots of average RPD values of the algorithms for different problem sizes and different SDST groups are given in Figure 7-16, Figure 7-17, Figure 7-18 and Figure 7-19. These plots summarize the results of the algorithms for $t = 90$. Again, IG_VLS$_{RCT}$_1 shows the worst performance. However, performance of the algorithms are variable in these plots. For example, IG_RS$_{LS}$ is almost the worst algorithm for $100\ x\ 5$ SDST10 instances. However, IG_RS$_{LS}$ is the best performing algorithm for the most of the problem groups, especially for large SDST distributions. Again, the conclusion is that swap neighborhood is not useful for problems with SDST, especially large ones.

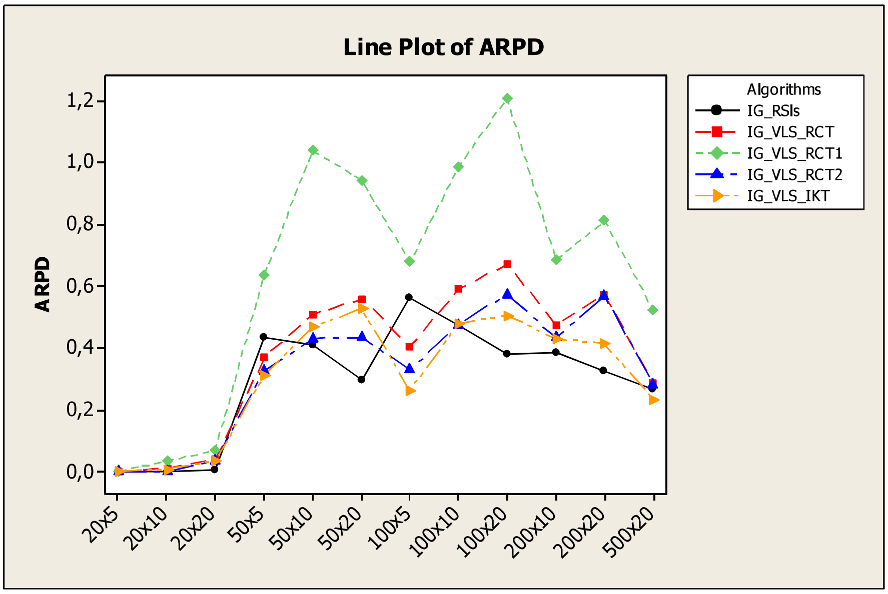

Figure 7-16 Plot of average percentage deviations of algorithms for SDST10 instances with t=90

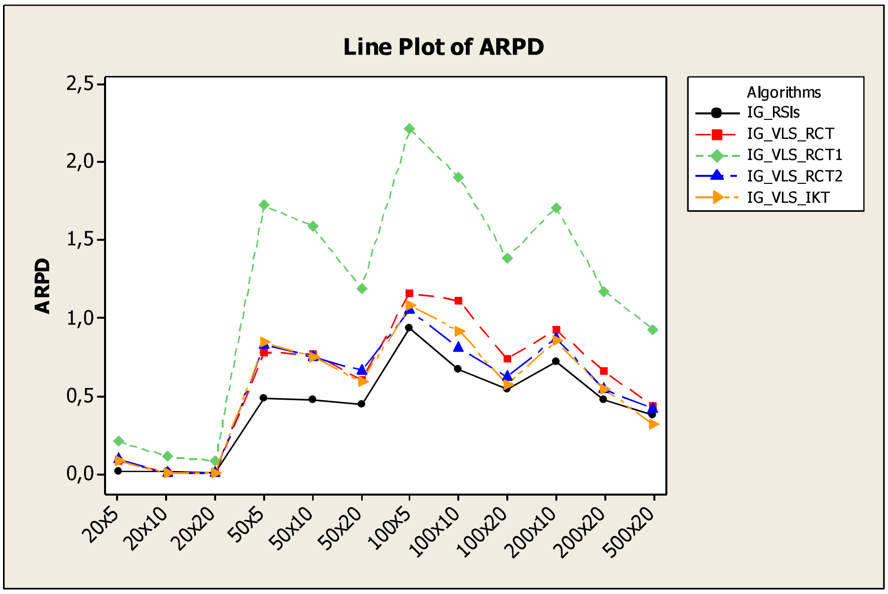

Figure 7-17 Plot of average percentage deviations of algorithms for SDST50 with t=90

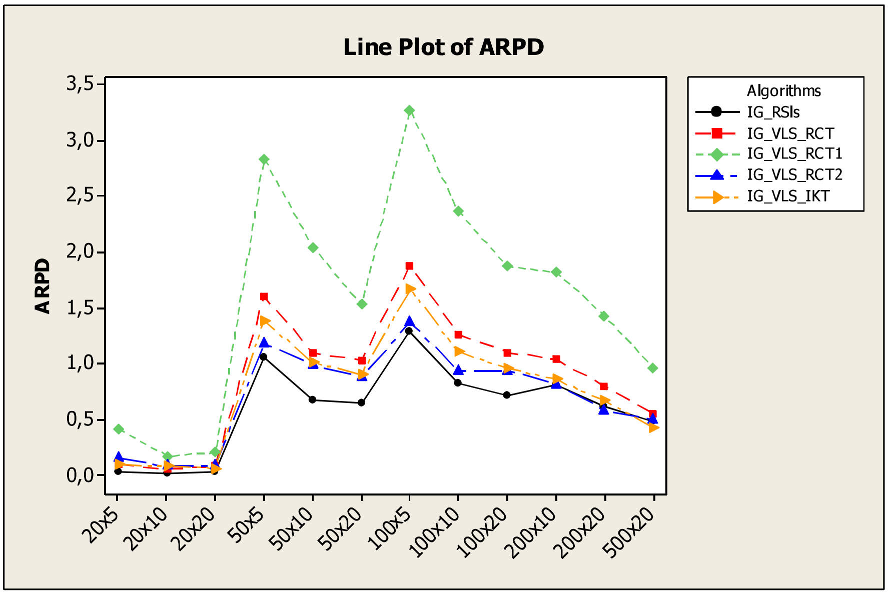

Figure 7-18 Plot of average percentage deviations of algorithms for SDST100 with t=90

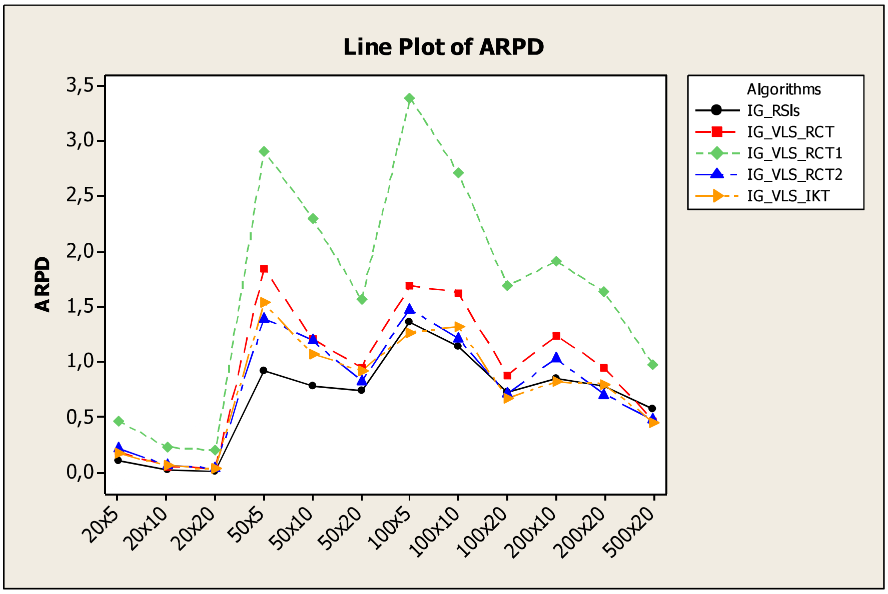

Figure 7-19 Plot of average percentage deviations of algorithms for SDST125 with t=90

Interval plots for the algorithms for different CPU limits are given in Figure 7-20, Figure 7-21 and Figure 7-22 for detailed performance comparison of the algorithms, in order to clarify whether the observed differences in average relative percentage deviations are statistically significant. IG_VLS$_{RCT}$_1 is clearly worse than the other algorithms. Although IG_RS$_{LS}$ performs better than the other algorithms, this difference does not have a statistical significance since the interval plots of IG_RS$_{LS}$ collides with other algorithms.

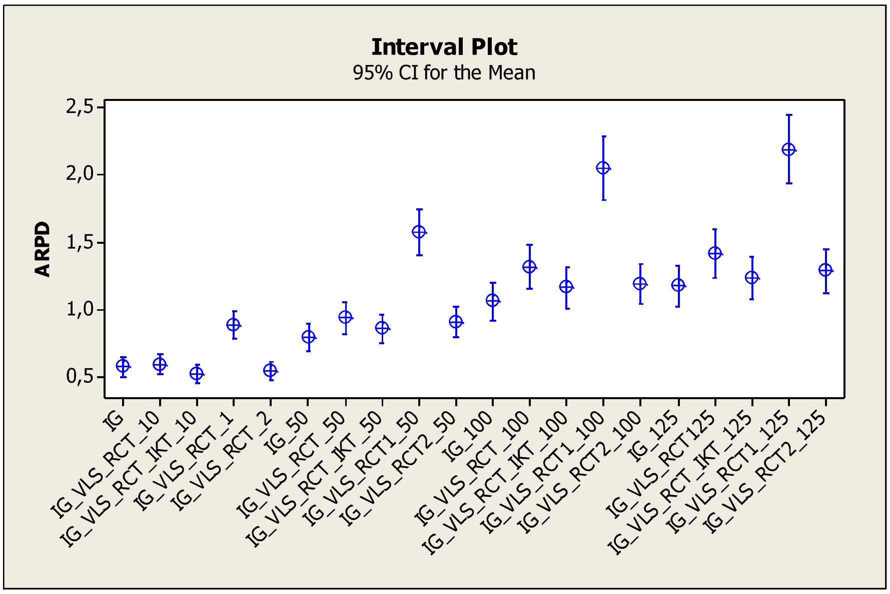

Figure 7-20 Interval plot of algorithms for t=30

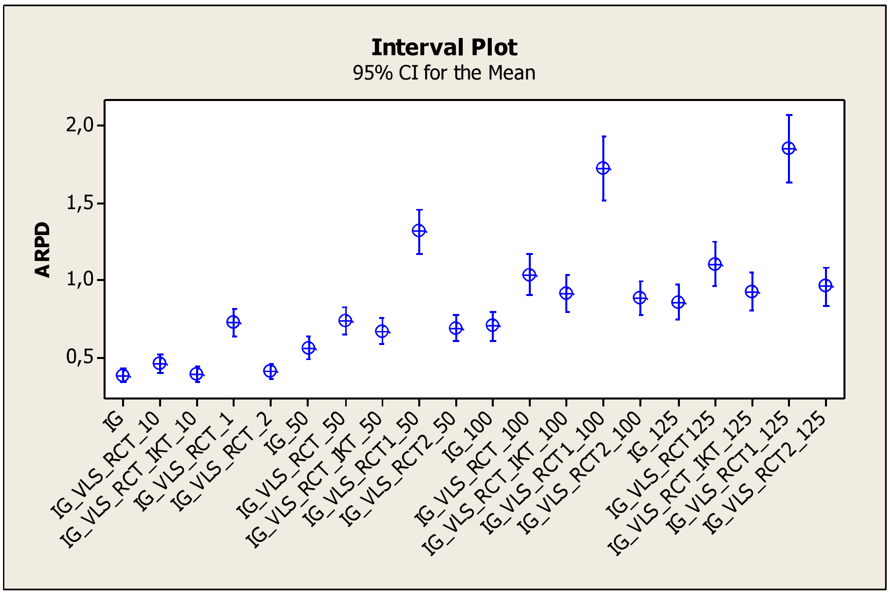

Figure 7-21 Interval plot of algorithms for t=60

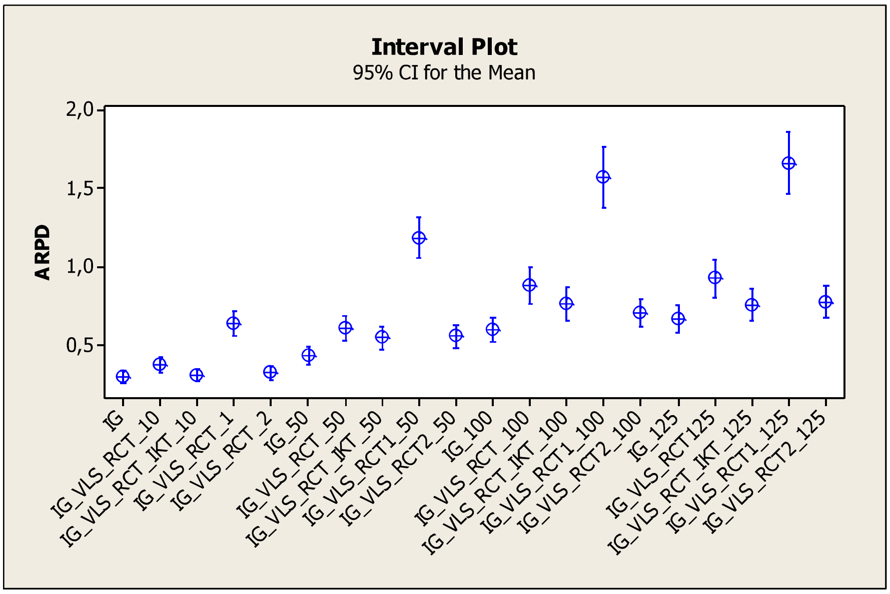

Figure 7-22 Interval plot of algorithms for t=90

Finally, best known solutions found for each test instance is given in Appendix A-2 for the permutation flow shop problem with sequence dependent setup times under total flow time minimization objective.

At this point, in order to investigate the effect of the swap neighborhood search in total flow time minimization with sequence dependent set up times, a new experiment is carried out. IG_RS$_{LS}$ and IG_VLS$_{IKT}$ are executed on original Taillard instances which do not have sequence dependent setup times using two different CPU limits, using $t = 60$ and $t = 90$. The results are presented in Table 7-6.

	t=60		t=90	
$n\ x\ m$	**IG_RS$_{LS}$**	**IG_VLS$_{IKT}$**	**IG_RS$_{LS}$**	**IG_VLS$_{IKT}$**
$(20\ x\ 5)$	0.00	0.00	0.00	0.00
$(20\ x\ 10)$	0.01	0.00	0.01	0.00
$(20\ x\ 20)$	0.01	0.01	0.01	0.01
$(50\ x\ 5)$	0.37	0.22	0.29	0.17
$(50\ x\ 10)$	0.53	0.41	0.40	0.33
$(50\ x\ 20)$	0.54	0.49	0.48	0.41
$(100\ x\ 5)$	0.70	0.34	0.58	0.25
$(100\ x\ 10)$	0.70	0.34	0.53	0.25
$(100\ x\ 20)$	0.72	0.61	0.56	0.46
$(200\ x\ 10)$	0.51	0.37	0.39	0.24
$(200\ x\ 20)$	0.58	0.50	0.48	0.33
$(500\ x\ 20)$	0.28	0.26	0.24	0.24
Average	**0.41**	**0.30**	**0.33**	**0.22**

Table 7-6 Average relative percentage deviations of IG_RS$_{LS}$ and VLS$_{IKT}$ algorithms for original Taillard instances with t = 60 and t = 90

From the previous results, it is obvious that performance of IG_RS$_{LS}$ algorithm is better than other IG versions in the presence of sequence dependent setup times. However, the performance of IG_VLS$_{IKT}$ is better for problems without sequence dependent setup times. This experiment shows that swap neighborhood does not work well in the presence of sequence dependent setup times, especially with high values.

The results in Table 6-7 is also presented graphically in Figure 7-24 and Figure 7-25 where average relative deviations are plotted for each $n \times m$ combination.

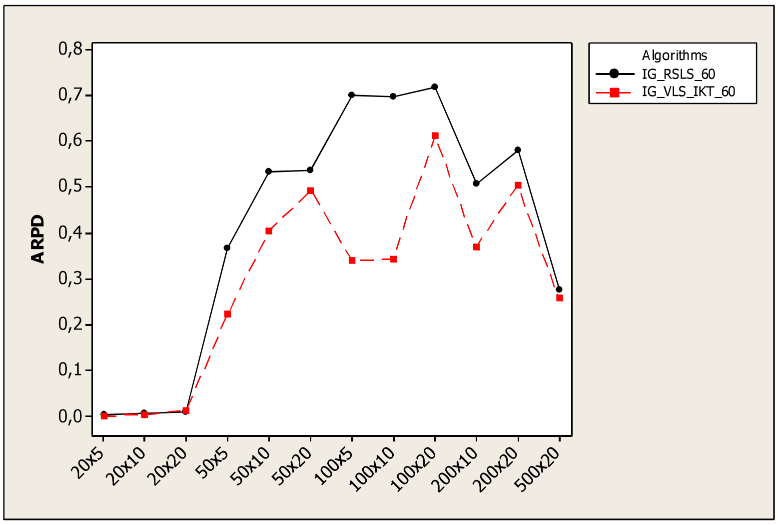

Figure 7-23 Plot of average percentage deviations of IG_RS$_{LS}$ and IG_VLS$_{IKS}$ algorithms for original Taillard instances with t=60

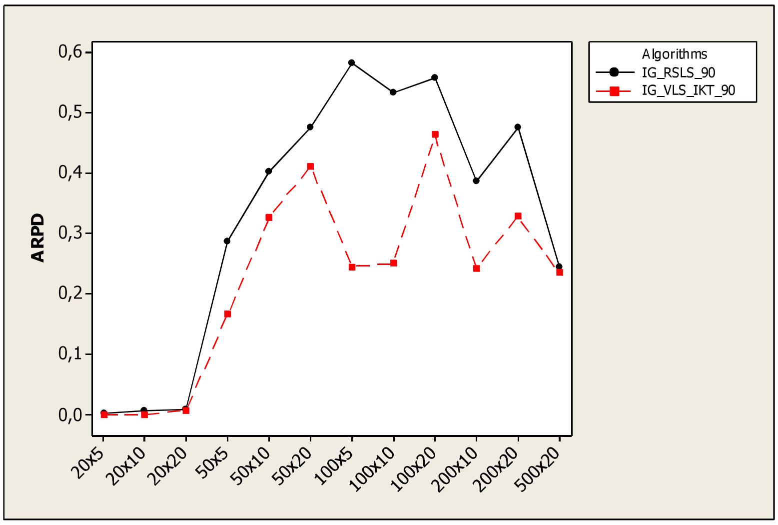

Figure 7-24 Plot of average percentage deviations of IG_RS$_{LS}$ and IG_VLS$_{IKS}$ algorithms for original Taillard instances with t=90

Interval plot in Figure 7-25 shows that the observed performance difference between IG_VLS$_{IKT}$ and IG_RS$_{LS}$ algorithm is statistically significant for instances without sequence dependent setup times.

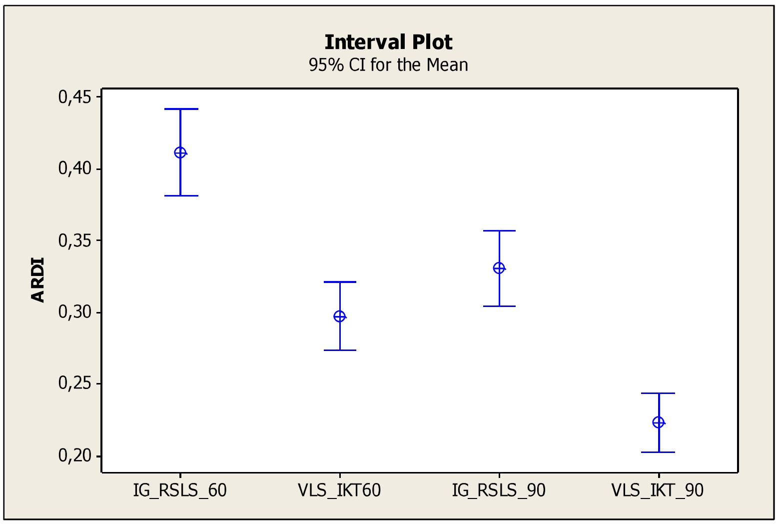

Figure 7-25 Interval plot of algorithms for original Taillard instance with t=60 and t=90

8 CONCLUSION

In this thesis study, metaheuristic algorithms for permutation flow shop problem with sequence dependent setup time are developed for two different optimization criteria. Permutation flow shop problem is one of the most studied scheduling problem, in contrast, scheduling problems that include sequence dependent setup times did not attract much attention from researchers. However, proper handling of SDST plays important role in optimizing the performance of the manufacturing systems. Researchers showed that, proper handling of SDST can increase production efficiency up to 20% (Pinedo, 2008). Two optimization criteria are studied in this thesis; makespan and total flow time minimization. Maximum machine utilization and the total production time minimization can be achieved using makespan optimization criterion. Total flow time optimization criterion on the other hand, allows minimization of the in-process inventory and stabilize resource utilization (Framinan, Leisten, & Ruiz-Usano, 2002).

Sequence dependent setup time for permutation flow shop problem under makespan minimization criterion is studied as the first problem. A new speed-up algorithm for swap neighborhood is developed for the first time in the literature for this problem. It is important to present and use this neighborhood in the search process. This speedup method allows swap neighborhood to be used in makespan minimization. Search space traversal of swap and insertion neighborhoods have different characteristics. Some points in the search space cannot be reached by insertion neighborhood search, and vice-versa. The proposed speed-up algorithm can be used to improve the search performance by visiting points that cannot be visited by the insertion neighborhood. The speedup algorithm is inspired from the Taillard's acceleration method developed for insertion neighborhood. Taillard's speed up method decreases the complexity of the NEH algorithm from $O(n^3m)$ to $O(n^2m)$. The speed up technique presented in this thesis cannot decrease the complexity but it decrements the required CPU time by 53% in average. Usage of swap neighborhood is still expensive in terms of CPU cost even with the implementation of the proposed speed up algorithm. For this reason, a version of the IG algorithm named IG_IJ_{LS} is developed. In this algorithm, swap neighborhood is applied with a predefined probability in the local search phase. The value of this probability parameter and the others parameters of the algorithm are determined by using the design of experiment approach. The proposed algorithm is coded in Java. 75 new test instances are created with random setup times that are uniformly distributed in the range [1,124]. Then, the calibrated algorithms are run on a test set from the literature (Ruiz & Stützle, 2008). IG_RS_{LS} and two other IG algorithm version that use variable neighborhood search algorithm in the local search phase are also coded in Java, in order to make a fair comparison. Experimental results show that IG_IJ_{LS} with a local search guided by an iteration jumping probability outperforms two other variants of the IG algorithms. Performances of IG_RS_{LS} and IG_IJ_{LS} were almost identical. Ultimately, 250 out of 480 best known solutions provided in http://www.upv.es/gio/rruiz are further improved by the proposed algorithms, together with 124 being equal and 106 being inferior. These new best results are presented in Appendix A-1. The new best known solutions also support our idea about the insertion neighborhood and swap neighborhood is able to find different minimum values. Giving even a little chance to swap neighborhood in local search may lead to better solutions.

The second problem studied in the thesis is total flow time minimization for permutation flow shop problem with sequence dependent setup times. This problem is studied for the first time in the literature. The speed up method for total flow time computation that is proposed by Li and Wu (2009) is adapted to consider sequence dependent setup times for this problem. This speedup method is used in the implementation of the NEH heuristic, in construction phase of the destruction-construction procedure of the IG algorithm, as well as, in the local search method that utilizes insert and swap neighborhoods. While the speedup method of Li and Wu (2009) cannot decrease the complexity, it reduces the required CPU time by 50% in average. Five different algorithms are designed and implemented for this problem. IG_RS_{LS}, IG_VLS_{RCT}, $IG\_VLS_{RCT}\_1$, $IG\_VLS_{RCT}\_2$ and IG_VLS_{IKT} algorithms are coded in Java. Design of experiment approach is used to determine values of the algorithms' parameters. Experiments are carried out individually for each algorithm and the results are analyzed in order to find the best parameter values. The results of IG_RS_{LS} are better than the other IG variants, however, there is no statistically significant difference. The analysis of the algorithm performances show that insertion neighborhood is better than swap neighborhood for total flow time minimization with the presence of sequence dependent setup times. Another experiment showed that using insertion and swap neighborhood is better than using only insertion neighborhood for total flow time minimization for problems without sequence dependent setup times. Best results found for total flow time minimization are listed in Appendix A-2.

For future work, the developed swap speedup algorithm can be adapted to other scheduling variants.

REFERENCES

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times / costs. *European Journal of Operational Research, 246*, 345-378.

Allahverdi, A., & Aldowaisan, T. (2002). New heuristics to minimize total completion time in m-machine flowshops. *International Journal of Production Economics, 77*(1), 71 - 83.

Allahverdi, A., Gupta, J. N., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *OMEGA the International Journal of Management Sciences*(27), 219-239.

Allahverdi, A., Ng, C., Cheng, T., & Kovalyov, M. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research, 187*, 985-1032.

Allahverdi, T., Gupta, & Aldowaisan, J. (1999). A review of scheduling research involving setup considerations. *OMEGA The International Journal of Management Sciences, 27*, 219-39.

Baker, K. R., & Trietsch, D. (2009). *PRINCIPLES OF SEQUENCING AND SCHEDULING.* A JOHN WILEY & SONS, INC. PUBLICATION.

Cheng, T., G. J., & Wang, G. (2000). A review of flowshop scheduling research with setup times. *Production and Operations Management, 9*, 262-282.

Ciavotta, M., Minella, G., & Ruiz, R. (2010). Multi-objective sequence dependent setup times flowshop scheduling: a new algorithm and a comprehensive study. *European Journal of Operational Research, 227*(2), 301-313.

Corwin, B. D., & Esogbue, A. O. (1974). Two machine flow shop scheduling problems with sequence dependent setup times: A dynamic-programming approach. *Nav. Res. Logist., 21*, 515-524.

Cox, J., Blackstone, J., & Spencer, M. (1992). *APICS Dictionary*. Falls Church, Virginia: American Production and Inventory Control Society.

Das, S., Gupta, J., & Khumawala, B. (1995). A savings index heuristic algorithm for flowshop scheduling with sequence-dependent set-up times. *Journal of the Operational Research Society, 46*(11), 1365-73.

Dhingra, A. K. (2012). *MULTI-OBJECTIVE FLOW SHOP SCHEDULING USING METAHEURISTICS.* Department of Mechanical Engineering. Kurukshetra: National Institute of Technology Kurukshetra.

Dipak, L., & Sarin, S. C. (2008). Aheuristic to minimize total flowtime in permutation flowshop. *Omega*(37), 734 - 739.

Dong, X., Huang, H., & Chen, P. (2008). An improved NEH-based heuristic for the permutation flowshop problem. *Computers & Operations Research, 35*(12), 3962-3968.

Dorigo, M., & Gambardella, L. M. (1997). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 53-66.

Ekşioğlu, B., Ekşioğlu, S., & Jain, P. (2008). A tabu search algorithm for the flowshop scheduling problem with changing neighoods. *Computers & Industrial Engineering, 54*, 1-11.

El-Ghazali, T. (2009). *METAHEURISTICS FROM DESIGN TO IMPLEMENTATION.* Hoboken, New Jersey, United States of America: John Wiley & Sons, Inc.

Fernandez-Viagas, V., & Framinan, J. M. (2014). On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem. *Computers & Operations Research, 45*, 60-67.

Fernandez-Viagas, V., & Framinan, J. M. (2015). A newsetofhigh-performingheuristicstominimise flowtime in permutation flowshops. *Computers & Operations Research, 53*, 68-80.

Framinan, J., & Leisten, R. (2003). An efficient constructive heuristic for flowtime minimisation in permutation flow shops. *Omega—International Journalof Management Science, 31*(4), 311-317.

Framinan, J., Leisten, R., & Ruiz-Usano, R. (2002). Efficient heuristics for flowshop sequencing with the objectives of make span and flow time minimisation. *European Journal of Operational Research, 141*(3), 559-69.

Gajpal, Y., Rajendran, C., & Ziegler, H. (2006). An ant colony algorithm for scheduling in flowshops with sequence-dependent setup times of jobs. *International Journal of Advanced Manufacturing Technology, 30*, 416-424.

Garey, M., Johnson, D., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*(1), 117-129.

Gendreaua, M., Laportea, G., & Guimarães, E. M. (2001). A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 183-189.

Grabowski, J., & Wodecki, M. (2004). A very fast tabu search algorithm for the permutation flowshop problem with makespan criterion. *Computers and Operations Research, 31*(11), 1891-1909.

Graves, S. C. (1981). A Review of Production Scheduling. *OPERATIONS RESEARCH*, 646-675.

Gray, M., Johnson, D., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research, 1*, 117-129.

Gupta, J. (1972). Heuristic algorithms for multistage flowshop scheduling problem. *AIIE Transactions*(4), 11-18.

Gupta, J., & Darrow, W. (1986). The two-machine sequence dependent flowshop scheduling problem. *European Journal of Operational Research, 24*(3), 439-446.

Hansen, P., & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research, 130*(3), 449-467.

Ho, J. C. (1995). Flowshop sequencing with mean flowtime objective. *European Journal of Operational Research*(81), 571-578.

Jackson, J. (1955). *Scheduling a Production Line to Minimize Maximum Tardiness.* Los Angeles: Management Science Research Project, University of California.

Jacobs, L., & Brusco, M. (1995). A local search heuristic for large set-covering problems. *Naval Research Logistics Quarterly, 42*(7), 1129–1140.

JK., W., & Prescott, W. (1969). The influence of setup time on job shop performance. *Manage Sci*(16), B274-B280.

Johnson, C. G. (2008, April). A Design Framework for Metaheuristics. *Artificial Intelligence Review, 29*(2), 163-178.

Johnson, S. (1954). Optimal two and three stage production schedule with set up times included. *Res Log Quart Vol 1*, 61-68.

Kalczynski, P. J., & Kamburowski, J. (2007). An improved NEH heuristic to minimize makespan in permutation flow shops. *Omega , 35*, 53-60.

Korst, J. H., Aarts, E. H., & Michiels, W. (2005). Simulated Annealing. *Search Methodologies* (s. 187-210). içinde New York, NY: Springer.

Laha, D., & Sarin, S. (2009). A heuristic to minimize total flow time in permutation flow shop. *Omega, 37*(3), 734-739.

Li, X., & Zhang, Y. (2012). Adaptive Hybrid Algorithms for the Sequence-Dependent Setup Time Permutation Flow Shop Scheduling Problem. *IEEE Transactions on automation science and engineering, 9*(3), ***.

Li, X., Wang, Q., & Wu, C. (2009). Efficient composite heuristics for total flowtime minimization in permutation flow shops. *Omega, 37*(1), 155-164.

Liu, J., & Reeves, C. (2001). Constructive and composite heuristic solutions to the P‖∑Ci scheduling problem. *European Journal of Operational Research, 23*(2), 439-452.

Luh, P., Gou, L., Zhang, Y., Nagahora, T., Tsuji, M., Yoneda, K., . . . and Kano, T. (1998). Job shop scheduling with group-dependent setups, finite buffers, and long time horizon. *Annals of Operation Research*(76), 233-259.

Luke, S. (2015). *Essentials of Metaheuristics : A Set of Undergraduate Lecture Notes* (second b.). George Mason University: Lulu.

Marchiori, E., & Steenbeek, A. (2000). An evolutionary algorithm for large set covering problems with applications to airline crew scheduling. *Real-World Applications of Evolutionary Computing, EvoWorkshops - 2000* (s. 367–381). Berlin,: Springer-Verlag.

Metropolis, M., Rosenbluth, A., Rosenbluth, M., & Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics, 21*(6), 1087-1092.

Mirabi, M. (2011). Ant colony optimization technique for the sequence-dependent flowshop scheduling problem. *The International Journal of Advanced Manufacturing Technology, 55*, 317-326.

Mirabi, M. (2014). A novel hybrid genetic algorithm to solve the sequence-dependent permutation flow-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology, 71*, 429-437.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms.* Cambridge, MA: MIT Press.

Miyazaki, S., Nishiyama, N., & Hashimoto, F. (1978). An adjacent pairwise approach to the mean flowtime scheduling problem. *Journal of the Operations Research Society of Japan*(21), 287-299.

Mladenovic, N., & Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research*, *24*(11), 1097-1100.

Montgomery, D. C. (2001). *Design and Analysis of Experiments.* New York: Wiley .

Nawaz, M., Enscore, J. E., & H. I. (1983). A heuristic algorithm for the m machine, n job flowshop sequencing problem. *Omega—International JournalofManagement Science*, *11*(1), 91-95.

Nowicki, E., & Smutnicki, C. (1996). A fast tabu search algorithm for the permutation flowshop problem. *European Journal of Operational Research*, *91*, 160-175.

Osman, I. H., & Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operational Research*, 513-628.

Osman, I., & Potts, C. (1989). Simulated annealing for permutation flow-shop scheduling. *Omega*, 551-557.

Pan, Q.-K., & Ruiz, R. (2013). A comprehensive review and evaluation of permutation flow shop heuristics. *Computers and OperationsResearch*(40), 117-128.

Pinedo, M. L. (2008). *Scheduling Theory, Algorithms, and Systems* (Third Edition b.). NewYork, NY: Prentice Hall.

Potts, C., & Kovalyov, M. (2000). Scheduling with batching: A review. *European Journal of Operational Research, 120*, 228-349.

Rajendran, C., & Ziegler, H. (1997). An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs. *European Journal of Operational Research*(103), 129-138.

ReliaSoft Corporation, T. A. (2015). *ReliaSoft Corporation*. ReliaSoft Corporation Wikipage: http://reliawiki.org/index.php/General_Full_Factorial_Designs adresinden alındı

Ribas, I., Companys, R., & Tort-Martorell, X. (2015, April). An efficient Discrete Artificial Bee Colony algorithm for the blocking flow shop problem with total flowtime minimization. *Expert Systems with Applications, 42*, 6155-6167.

Rios-Mercado, R., & Bard, J. (1998a). Computational experience with a branch-and-cut algorithm for flowshop scheduling with setups. *Computers and Operations Research*, *25*(5), 351-366.

Rios-Mercado, R., & Bard, J. (1998b). Heuristics for the flowline problem with setup costs. *European Journal of Operational Research, 110*(1), 76-98.

Rios-Mercado. R.Z, B. J. (1999b). An enhanced TSP-based heuristic for makespan minimization in a flow shop with setup times. *Journal of Heuristics*, *5*(1), 53-70.

Rios-Mercado. R.Z., B. J. (1999a). A branch-and-bound algorithm for permutation flow shops with sequence-dependent setup times. *IIE Transactions, 31*(8), 721-731.

Ruiz, R., & Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research, 17,* 2033-2049.

Ruiz, R., & Stützle, T. (2008). An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research, 188,* 1143-1159.

Ruiz, R., Maroto, C., & Alcaraz, J. (2005). Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. *European Journal of Operational Research, 165,* 34-54.

Russell, S. J., & Norvig, P. (2010). *Artificial Intelligence A Modern Approach - Third Edition.* New Jersey: Pearson.

Shi, R., Zhou, Y., & Zhou, H. (2007). A Hybrid Escalating Evolutionary Algorithm for Multi-objective Flow-Shop Scheduling. *Third International Conference on Natural Computation ICNC 2007* (s. 426-430). Haikou: IEEE.

Simons, J. J. (1992). Heuristics in flow shop scheduling with sequence dependent setup times. *Omega-International Journal of Management Science, 20*(2), 215-225.

Smith, W. (1956). Various Optimizers for Single Stage Production. *Naval Research Logistics Quarterly, 3*, 59-66.

Stafford, J., & Tseng, E. (2002). Two models for a family of flowshop sequencing problems. *European Journal of Operational Research, 142*, 282-293.

Stuetzle, T. (1998). An ant approach for the flow shop problem. *Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing (EUFIT).* Verlag Mainz, Aachen, Germany,.

Taillard, E. D. (1990). Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research, 47*(1), 65-74.

Vanchipura, R., & Sridharan, R. (2013). Development and analysis of constructive heuristic algorithms for flow shop scheduling problems with sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology, 67*(5-8), 1337-1353.

VictorFernandez-Viagas, J. (2015). Efficient non-population-based algorithms for the permutation flow shop scheduling problem with make span minimisation subject toamaximum tardiness. *Computers&OperationsResearch, 64*(86-96).

Wang Chengen, C. C.-M. (1997). Heuristicapproachesfor n/m/F/Ci, scheduling problems. *European Journal of Operational Research*(96), 636-44.

Wight, O. (1984). *Production and Inventory Management in the Computer Age.* New York: Van Nostrand Reinhold Company, Inc.

Woo, H., & Yim, D. (1998). A heuristic algorithm for mean flowtime objective in flowshop scheduling. *Computers & Operations Research*, *25*(3), 175-182.

Yang. W.H., L. C. (1999). Survey of scheduling research involving setup times. *International Journal of Systems Science, 30*, 143-155.

Yi, Y., & Wang, D. W. (2003). Soft computing for scheduling with batch setup times and earliness-tardiness penalties on parallel machines. *Journal of Intelligent Manufacturing*, 311-322.

CURRICULUM VITEA

Yavuz İNCE was born on November 28, 1981 in İzmir. He graduated from Dokuz Eylül University Electrical and Electronics Engineering Department in 2004. He got his master's degree in Electronics Engineering from the same university in 2007. He worked as a design engineer in Vestel Electronics and Beko Electronics companies from 2004 to 2009. Between 2009 and 2010, he worked as intern software engineer in Brightstorm Inc., in San Francisco CA/USA. He has been working as a lecturer in Gediz University Computer Engineering Department since March 2011.

APPENDIX

A-1) New best known solutions for Make Span Problem

Problem Instance	BKS	IG_VNS1	IG_VNS2	IG_IJ	IG*_RS	Problem Instance	BKS	IG_VNS1	IG_VNS2	IG_IJ	IG*_RS
SDST10_ta001 (20 x 5)	***1330***	***1330***	***1330***	***1330***	***1330***	SDST50_ta001 (20 x 5)	***1567***	***1567***	***1567***	***1567***	***1567***
SDST10_ta002 (20 x 5)	***1401***	***1401***	***1401***	***1401***	***1401***	SDST50_ta002 (20 x 5)	***1580***	***1580***	***1580***	***1580***	***1580***
SDST10_ta003 (20 x 5)	***1161***	***1161***	***1161***	***1161***	***1161***	SDST50_ta003 (20 x 5)	***1446***	***1446***	***1446***	***1446***	***1446***
SDST10_ta004 (20 x 5)	***1370***	***1370***	***1370***	***1370***	***1370***	SDST50_ta004 (20 x 5)	***1644***	***1644***	***1644***	***1644***	***1644***
SDST10_ta005 (20 x 5)	***1303***	***1303***	***1303***	***1303***	***1303***	SDST50_ta005 (20 x 5)	***1526***	***1526***	***1526***	***1526***	1528
SDST10_ta006 (20 x 5)	***1269***	***1269***	***1269***	***1269***	***1269***	SDST50_ta006 (20 x 5)	***1510***	***1510***	***1510***	***1510***	***1510***
SDST10_ta007 (20 x 5)	***1294***	***1294***	***1294***	***1294***	***1294***	SDST50_ta007 (20 x 5)	***1531***	***1531***	***1531***	***1531***	***1531***
SDST10_ta008 (20 x 5)	***1282***	***1282***	***1282***	***1282***	***1282***	SDST50_ta008 (20 x 5)	***1554***	***1554***	***1554***	***1554***	***1554***
SDST10_ta009 (20 x 5)	***1313***	***1313***	***1313***	***1313***	***1313***	SDST50_ta009 (20 x 5)	***1585***	***1585***	***1585***	***1585***	***1585***
SDST10_ta010 (20 x 5)	***1178***	***1178***	***1178***	***1178***	***1178***	SDST50_ta010 (20 x 5)	***1426***	***1426***	***1426***	***1426***	***1426***
SDST10_ta011 (20 x 10)	***1677***	***1677***	***1677***	***1677***	***1677***	SDST50_ta011 (20 x 10)	***2009***	***2009***	***2009***	***2009***	***2009***
SDST10_ta012 (20 x 10)	***1751***	***1751***	***1751***	***1751***	***1751***	SDST50_ta012 (20 x 10)	***2065***	***2065***	***2065***	***2065***	***2065***
SDST10_ta013 (20 x 10)	***1588***	***1588***	***1588***	***1588***	***1588***	SDST50_ta013 (20 x 10)	***1897***	***1897***	***1897***	***1897***	***1897***
SDST10_ta014 (20 x 10)	***1465***	***1465***	***1465***	***1465***	***1465***	SDST50_ta014 (20 x 10)	***1794***	***1794***	***1794***	***1794***	***1794***
SDST10_ta015 (20 x 10)	***1510***	***1510***	***1510***	***1510***	***1510***	SDST50_ta015 (20 x 10)	***1842***	***1842***	***1842***	***1842***	***1842***
SDST10_ta016 (20 x 10)	***1487***	***1487***	***1487***	***1487***	***1487***	SDST50_ta016 (20 x 10)	***1816***	***1816***	***1816***	***1816***	***1816***
SDST10_ta017 (20 x 10)	***1573***	***1573***	***1573***	***1573***	***1573***	SDST50_ta017 (20 x 10)	***1858***	***1858***	***1858***	***1858***	***1858***
SDST10_ta018 (20 x 10)	***1630***	***1630***	***1630***	***1630***	***1630***	SDST50_ta018 (20 x 10)	***1962***	***1962***	***1962***	***1962***	***1962***
SDST10_ta019 (20 x 10)	***1676***	***1676***	***1676***	***1676***	***1676***	SDST50_ta019 (20 x 10)	***1985***	***1985***	***1985***	***1985***	***1985***
SDST10_ta020 (20 x 10)	***1688***	***1688***	***1688***	***1688***	***1688***	SDST50_ta020 (20 x 10)	***2013***	***2013***	***2013***	***2013***	***2013***
SDST10_ta021 (20 x 20)	***2391***	***2391***	***2391***	***2391***	***2391***	SDST50_ta021 (20 x 20)	***2754***	***2754***	***2754***	***2754***	***2754***
SDST10_ta022 (20 x 20)	***2193***	***2193***	***2193***	***2193***	***2193***	SDST50_ta022 (20 x 20)	***2565***	***2565***	***2565***	***2565***	***2565***
SDST10_ta023 (20 x 20)	***2414***	***2414***	***2414***	***2414***	***2414***	SDST50_ta023 (20 x 20)	***2748***	***2748***	***2748***	***2748***	***2748***
SDST10_ta024 (20 x 20)	***2315***	***2315***	***2315***	***2315***	***2315***	SDST50_ta024 (20 x 20)	***2658***	***2658***	***2658***	***2658***	***2658***
SDST10_ta025 (20 x 20)	***2386***	***2386***	***2386***	***2386***	***2386***	SDST50_ta025 (20 x 20)	***2760***	***2760***	***2760***	***2760***	***2760***
SDST10_ta026 (20 x 20)	***2321***	***2321***	***2321***	***2321***	***2321***	SDST50_ta026 (20 x 20)	***2686***	***2686***	***2686***	***2686***	***2686***
SDST10_ta027 (20 x 20)	***2360***	***2360***	***2360***	***2360***	***2360***	SDST50_ta027 (20 x 20)	***2712***	***2712***	***2712***	***2712***	***2712***
SDST10_ta028 (20 x 20)	***2296***	***2296***	***2296***	***2296***	***2296***	SDST50_ta028 (20 x 20)	***2668***	***2668***	***2668***	***2668***	***2668***
SDST10_ta029 (20 x 20)	***2335***	***2335***	***2335***	***2335***	***2335***	SDST50_ta029 (20 x 20)	***2701***	***2701***	***2701***	***2701***	***2701***
SDST10_ta030 (20 x 20)	***2267***	***2267***	***2267***	***2267***	***2267***	SDST50_ta030 (20 x 20)	***2635***	***2635***	***2635***	***2635***	***2635***
SDST10_ta031 (50 x 5)	2814	2816	2817	2816	**2813**	SDST50_ta031 (50 x 5)	**3250**	3256	3262	3258	3258
SDST10_ta032 (50 x 5)	**2946**	2949	2948	2949	2947	SDST50_ta032 (50 x 5)	3429	3433	3438	**3427**	3448
SDST10_ta033 (50 x 5)	**2734**	2738	2739	2739	2739	SDST50_ta033 (50 x 5)	3245	3259	3257	**3244**	3247

SDST10_ta034 (50 x 5)	**2883**	2888	2888	2889	2888	SDST50_ta034 (50 x 5)	**3391**	3396	3416	3395	3395
SDST10_ta035 (50 x 5)	2952	2957	2956	2956	2955	SDST50_ta035 (50 x 5)	***3400***	3411	3415	3403	***3400***
SDST10_ta036 (50 x 5)	2945	2947	2950	2948	2945	SDST50_ta036 (50 x 5)	**3429**	3452	3458	3453	3451
SDST10_ta037 (50 x 5)	***2848***	2850	2852	***2848***	2850	SDST50_ta037 (50 x 5)	**3338**	3349	3359	3351	3356
SDST10_ta038 (50 x 5)	2809	2813	2811	2812	2811	SDST50_ta038 (50 x 5)	**3306**	3313	3320	3308	3314
SDST10_ta039 (50 x 5)	2673	2675	2674	2674	**2669**	SDST50_ta039 (50 x 5)	3174	3170	3184	**3159**	3176
SDST10_ta040 (50 x 5)	**2867**	2871	2871	2869	2868	SDST50_ta040 (50 x 5)	**3350**	3362	3365	3351	3361
SDST10_ta041 (50 x 10)	3210	3211	3216	3209	**3208**	SDST50_ta041 (50 x 10)	**3923**	3948	3939	3927	3950
SDST10_ta042 (50 x 10)	**3080**	3083	3087	3081	***3080***	SDST50_ta042 (50 x 10)	3807	3811	3829	**3794**	3809
SDST10_ta043 (50 x 10)	3060	**3055**	3059	**3055**	3056	SDST50_ta043 (50 x 10)	3796	3815	3811	**3785**	3818
SDST10_ta044 (50 x 10)	3227	3232	3228	3228	**3225**	SDST50_ta044 (50 x 10)	3956	3967	**3935**	3962	3966
SDST10_ta045 (50 x 10)	3200	3201	**3199**	3203	3202	SDST50_ta045 (50 x 10)	3939	3939	3940	3932	**3927**
SDST10_ta046 (50 x 10)	3196	3204	3200	3201	3199	SDST50_ta046 (50 x 10)	3926	3935	3932	3935	**3925**
SDST10_ta047 (50 x 10)	3285	3287	3290	**3281**	**3281**	SDST50_ta047 (50 x 10)	3986	3987	3994	3986	**3981**
SDST10_ta048 (50 x 10)	**3222**	3224	3225	3223	3224	SDST50_ta048 (50 x 10)	3950	3949	3958	**3939**	**3939**
SDST10_ta049 (50 x 10)	**3093**	3096	3093	3090	3096	SDST50_ta049 (50 x 10)	**3829**	3841	3839	3833	3836
SDST10_ta050 (50 x 10)	3272	3274	3278	**3268**	3270	SDST50_ta050 (50 x 10)	3983	**3979**	4004	3988	3988
SDST10_ta051 (50 x 20)	4108	4107	4107	**4101**	4106	SDST50_ta051 (50 x 20)	4980	**4977**	4978	4991	4988
SDST10_ta052 (50 x 20)	3942	3947	3953	**3934**	3939	SDST50_ta052 (50 x 20)	**4812**	4833	4848	4820	4820
SDST10_ta053 (50 x 20)	3895	3909	3894	**3883**	3894	SDST50_ta053 (50 x 20)	4781	4787	4795	4781	**4779**
SDST10_ta054 (50 x 20)	***3973***	3985	3982	3981	***3973***	SDST50_ta054 (50 x 20)	4866	**4858**	4888	4873	4873
SDST10_ta055 (50 x 20)	3867	3873	3876	**3865**	3871	SDST50_ta055 (50 x 20)	**4769**	4788	4794	4791	4775
SDST10_ta056 (50 x 20)	3930	3943	**3924**	3933	3935	SDST50_ta056 (50 x 20)	**4791**	4807	4820	4796	4798
SDST10_ta057 (50 x 20)	3966	3960	3969	**3956**	3959	SDST50_ta057 (50 x 20)	4832	4852	4852	**4825**	4865
SDST10_ta058 (50 x 20)	3964	3962	3965	3964	**3955**	SDST50_ta058 (50 x 20)	**4831**	4844	4847	4842	4837
SDST10_ta059 (50 x 20)	3996	4003	3992	**3985**	3992	SDST50_ta059 (50 x 20)	4864	**4854**	4861	4866	4865
SDST10_ta060 (50 x 20)	4008	3998	4014	**3995**	4000	SDST50_ta060 (50 x 20)	**4891**	4919	4917	4909	4906
SDST10_ta061 (100 x 5)	**5647**	5662	5671	5654	5652	SDST50_ta061 (100 x 5)	6542	6577	6595	6558	**6534**
SDST10_ta062 (100 x 5)	**5465**	5471	5477	5473	5470	SDST50_ta062 (100 x 5)	6389	6409	6450	**6388**	6397
SDST10_ta063 (100 x 5)	**5406**	5411	5415	5411	5410	SDST50_ta063 (100 x 5)	**6333**	6383	6412	6346	6367
SDST10_ta064 (100 x 5)	5213	5219	5213	5219	**5212**	SDST50_ta064 (100 x 5)	6182	6201	6213	6167	**6157**
SDST10_ta065 (100 x 5)	**5466**	5477	5476	5475	5472	SDST50_ta065 (100 x 5)	6417	6465	6485	6423	**6406**
SDST10_ta066 (100 x 5)	**5312**	5320	5321	5321	5317	SDST50_ta066 (100 x 5)	6270	6324	6297	**6254**	6273
SDST10_ta067 (100 x 5)	**5459**	5472	5479	5471	5472	SDST50_ta067 (100 x 5)	6390	6420	6434	6383	**6381**
SDST10_ta068 (100 x 5)	**5316**	5321	5332	5323	5327	SDST50_ta068 (100 x 5)	**6199**	6275	6296	6249	6250
SDST10_ta069 (100 x 5)	**5641**	5648	5652	5646	5642	SDST50_ta069 (100 x 5)	**6576**	6590	6611	6579	6579
SDST10_ta070 (100 x 5)	**5537**	5540	5548	5540	5541	SDST50_ta070 (100 x 5)	6492	6540	6523	**6490**	6497
SDST10_ta071 (100 x 10)	6084	6088	6083	6083	**6070**	SDST50_ta071 (100 x 10)	7450	7458	7482	7435	**7425**
SDST10_ta072 (100 x 10)	5683	5683	5693	**5681**	**5681**	SDST50_ta072 (100 x 10)	**7033**	7076	7067	7050	7041

SDST10_ta073 (100 x 10)	5931	5923	5928	**5920**	5921	SDST50_ta073 (100 x 10)	7262	7265	7275	7257	**7245**
SDST10_ta074 (100 x 10)	6182	6177	6185	**6167**	6175	SDST50_ta074 (100 x 10)	7549	**7531**	7608	7559	7542
SDST10_ta075 (100 x 10)	***5842***	5834	5855	***5842***	***5842***	SDST50_ta075 (100 x 10)	7240	7254	7253	**7214**	7230
SDST10_ta076 (100 x 10)	5607	5606	5605	5595	**5589**	SDST50_ta076 (100 x 10)	**6964**	7001	7020	6981	6973
SDST10_ta077 (100 x 10)	5884	5883	5898	**5876**	5879	SDST50_ta077 (100 x 10)	7126	7142	7144	7144	**712**5
SDST10_ta078 (100 x 10)	5958	**5949**	5969	5956	5961	SDST50_ta078 (100 x 10)	7290	7309	7333	**7277**	7281
SDST10_ta079 (100 x 10)	6177	**6171**	6195	6178	**6171**	SDST50_ta079 (100 x 10)	7452	7472	7465	7457	**7450**
SDST10_ta080 (100 x 10)	6081	6082	6080	**6076**	**6076**	SDST50_ta080 (100 x 10)	7364	**7352**	7393	7362	7367
SDST10_ta081 (100 x 20)	6744	6755	6750	**6738**	6747	SDST50_ta081 (100 x 20)	8437	**8377**	8448	8403	8395
SDST10_ta082 (100 x 20)	6701	6696	6717	6690	**6688**	SDST50_ta082 (100 x 20)	8387	8432	8423	**8376**	8392
SDST10_ta083 (100 x 20)	6770	6750	6788	**6741**	6751	SDST50_ta083 (100 x 20)	8422	8448	8439	**8403**	8418
SDST10_ta084 (100 x 20)	6734	6722	6737	6700	**6714**	SDST50_ta084 (100 x 20)	8389	8405	8406	8364	**8352**
SDST10_ta085 (100 x 20)	**6785**	6804	6813	6787	6789	SDST50_ta085 (100 x 20)	8471	8478	8495	8451	**8444**
SDST10_ta086 (100 x 20)	6867	6874	6884	**6850**	**6850**	SDST50_ta086 (100 x 20)	8548	8549	8567	**8527**	8528
SDST10_ta087 (100 x 20)	**6779**	6817	6801	6778	6782	SDST50_ta087 (100 x 20)	8482	8522	8546	**8464**	8511
SDST10_ta088 (100 x 20)	6954	6960	6971	**6935**	6943	SDST50_ta088 (100 x 20)	8662	8664	8666	**8639**	8648
SDST10_ta089 (100 x 20)	6808	6814	6815	**6781**	6796	SDST50_ta089 (100 x 20)	**8473**	8511	8533	8482	8494
SDST10_ta090 (100 x 20)	**6870**	6887	6895	6881	6879	SDST50_ta090 (100 x 20)	8519	8540	8523	**8500**	8508
SDST10_ta091 (200 x 10)	11354	11406	11411	11347	**11340**	SDST50_ta091 (200 x 10)	14005	14026	14045	13955	**13891**
SDST10_ta092 (200 x 10)	11226	11209	11242	**11202**	11208	SDST50_ta092 (200 x 10)	13902	13912	14000	**13808**	13841
SDST10_ta093 (200 x 10)	11521	11526	11525	11517	**11491**	SDST50_ta093 (200 x 10)	14087	14116	14164	14035	**13992**
SDST10_ta094 (200 x 10)	11294	11318	11350	11310	**11282**	SDST50_ta094 (200 x 10)	13873	13905	13960	**13818**	13838
SDST10_ta095 (200 x 10)	11207	11213	11226	11180	**11187**	SDST50_ta095 (200 x 10)	13849	13934	13996	**13839**	13840
SDST10_ta096 (200 x 10)	10974	10968	10983	10941	**10933**	SDST50_ta096 (200 x 10)	13653	13652	13716	13619	**13561**
SDST10_ta097 (200 x 10)	11423	11428	11441	**11407**	**11407**	SDST50_ta097 (200 x 10)	14115	14118	14187	**14019**	14046
SDST10_ta098 (200 x 10)	11362	11369	11385	11357	**11353**	SDST50_ta098 (200 x 10)	14018	14048	14114	**13947**	13961
SDST10_ta099 (200 x 10)	11098	11105	11124	11087	**11082**	SDST50_ta099 (200 x 10)	13857	13844	13910	**13759**	13763
SDST10_ta100 (200 x 10)	11284	11289	11303	11279	**11271**	SDST50_ta100 (200 x 10)	13894	13890	14038	**13850**	13851
SDST10_ta101 (200 x 20)	12168	12183	12225	12142	**12150**	SDST50_ta101 (200 x 20)	15450	15461	15540	15411	**15389**
SDST10_ta102 (200 x 20)	12278	12323	12339	**12257**	12267	SDST50_ta102 (200 x 20)	15644	15610	15727	15558	**15556**
SDST10_ta103 (200 x 20)	12337	12369	12371	12324	**12316**	SDST50_ta103 (200 x 20)	15689	15728	15751	15621	**15613**
SDST10_ta104 (200 x 20)	12300	12348	12354	**12283**	12303	SDST50_ta104 (200 x 20)	15627	15627	15704	15560	**15556**
SDST10_ta105 (200 x 20)	12196	12215	12247	12172	**12173**	SDST50_ta105 (200 x 20)	15470	15438	15542	15436	**15364**
SDST10_ta106 (200 x 20)	12220	12223	12245	12173	**12189**	SDST50_ta106 (200 x 20)	15514	15565	15622	15477	**15469**
SDST10_ta107 (200 x 20)	12329	12342	12381	**12314**	12331	SDST50_ta107 (200 x 20)	15669	15668	15744	15640	**15604**
SDST10_ta108 (200 x 20)	12362	12367	12391	**12331**	12338	SDST50_ta108 (200 x 20)	15645	15615	15678	**15557**	15602
SDST10_ta109 (200 x 20)	12228	12264	12274	12232	**12208**	SDST50_ta109 (200 x 20)	15544	15609	15663	**15526**	15540
SDST10_ta110 (200 x 20)	12330	12336	12372	**12277**	12302	SDST50_ta110 (200 x 20)	15694	15631	15760	**15566**	15587
SDST10_ta111 (500 x 20)	28491	28507	28625	28410	**28380**	SDST50_ta111 (500 x 20)	36729	36802	36950	36538	**36520**

SDST10_ta112 (500 x 20)	28940	29036	29129	28922	**28914**	SDST50_ta112 (500 x 20)	37113	37225	37377	**36974**	**36974**
SDST10_ta113 (500 x 20)	28695	28679	28751	28611	**28608**	SDST50_ta113 (500 x 20)	36854	36849	37035	36627	**36581**
SDST10_ta114 (500 x 20)	28719	28711	28864	28650	**28648**	SDST50_ta114 (500 x 20)	36904	36973	37265	**36691**	36722
SDST10_ta115 (500 x 20)	28596	28650	28727	**28526**	28587	SDST50_ta115 (500 x 20)	36793	36816	37021	36572	**36523**
SDST10_ta116 (500 x 20)	28756	28763	28866	**28679**	28681	SDST50_ta116 (500 x 20)	37006	37079	37293	36840	**36824**
SDST10_ta117 (500 x 20)	28516	28554	28650	**28447**	28451	SDST50_ta117 (500 x 20)	36674	36711	37051	**36499**	36523
SDST10_ta118 (500 x 20)	28884	28864	28954	**28780**	28797	SDST50_ta118 (500 x 20)	36942	37111	37335	**36803**	36822
SDST10_ta119 (500 x 20)	28398	28391	28468	28366	**28352**	SDST50_ta119 (500 x 20)	36575	36724	37026	36474	**36404**
SDST10_ta120 (500 x 20)	28737	28741	28854	**28674**	28703	SDST50_ta120 (500 x 20)	36843	36875	37201	36729	**36680**

	BKS	VNS1	VNS2	IG_IJ	IG*_RS		BKS	VNS1	VNS2	IG_IJ	IG*_RS
SDST100_ta001 (20 x 5)	***1891***	***1891***	***1891***	***1891***	***1891***	SDST125_ta001 (20 x 5)	***2065***	***2065***	***2065***	***2065***	***2065***
SDST100_ta002 (20 x 5)	***1881***	***1881***	***1881***	***1881***	***1881***	SDST125_ta002 (20 x 5)	***2040***	***2040***	***2040***	***2040***	***2040***
SDST100_ta003 (20 x 5)	***1758***	***1758***	***1758***	***1758***	***1758***	SDST125_ta003 (20 x 5)	***1933***	***1933***	***1933***	***1933***	***1933***
SDST100_ta004 (20 x 5)	***1973***	***1973***	***1973***	***1973***	***1973***	SDST125_ta004 (20 x 5)	***2137***	***2137***	***2137***	***2137***	***2137***
SDST100_ta005 (20 x 5)	***1813***	***1813***	***1813***	***1813***	***1813***	SDST125_ta005 (20 x 5)	***1979***	***1979***	***1982***	***1979***	***1979***
SDST100_ta006 (20 x 5)	***1824***	***1824***	***1824***	***1824***	***1824***	SDST125_ta006 (20 x 5)	***1979***	***1979***	***1979***	***1983***	***1979***
SDST100_ta007 (20 x 5)	***1855***	***1855***	***1855***	***1855***	***1855***	SDST125_ta007 (20 x 5)	***2002***	***2002***	***2002***	***2002***	***2002***
SDST100_ta008 (20 x 5)	***1894***	***1894***	***1894***	***1894***	***1894***	SDST125_ta008 (20 x 5)	***2060***	***2060***	***2060***	***2060***	***2060***
SDST100_ta009 (20 x 5)	***1879***	***1879***	***1879***	***1879***	***1879***	SDST125_ta009 (20 x 5)	***2005***	***2005***	***2005***	***2005***	***2005***
SDST100_ta010 (20 x 5)	***1732***	***1732***	***1732***	***1732***	***1732***	SDST125_ta010 (20 x 5)	***1876***	***1876***	***1876***	***1876***	***1876***
SDST100_ta011 (20 x 10)	***2444***	***2444***	***2444***	***2444***	***2444***	SDST125_ta011 (20 x 10)	***2656***	***2656***	***2656***	***2662***	***2656***
SDST100_ta012 (20 x 10)	***2458***	***2458***	***2458***	***2458***	***2458***	SDST125_ta012 (20 x 10)	***2661***	***2661***	***2661***	***2661***	***2661***
SDST100_ta013 (20 x 10)	***2303***	***2303***	***2303***	***2303***	***2303***	SDST125_ta013 (20 x 10)	***2515***	***2515***	***2515***	***2515***	***2515***
SDST100_ta014 (20 x 10)	***2212***	***2212***	***2212***	***2212***	***2212***	SDST125_ta014 (20 x 10)	***2415***	***2415***	***2415***	***2415***	***2433***
SDST100_ta015 (20 x 10)	***2282***	***2286***	***2286***	***2286***	***2286***	SDST125_ta015 (20 x 10)	***2502***	***2502***	***2502***	***2502***	***2502***
SDST100_ta016 (20 x 10)	***2231***	***2231***	***2231***	***2231***	***2231***	SDST125_ta016 (20 x 10)	***2445***	***2445***	***2445***	***2445***	***2445***
SDST100_ta017 (20 x 10)	***2282***	***2282***	***2282***	***2282***	***2282***	SDST125_ta017 (20 x 10)	***2485***	***2485***	***2485***	***2485***	***2485***
SDST100_ta018 (20 x 10)	***2381***	***2381***	***2381***	***2381***	***2381***	SDST125_ta018 (20 x 10)	***2586***	***2586***	***2586***	***2586***	***2593***
SDST100_ta019 (20 x 10)	***2376***	***2376***	***2376***	***2376***	***2376***	SDST125_ta019 (20 x 10)	***2588***	***2588***	***2588***	***2588***	***2588***
SDST100_ta020 (20 x 10)	***2443***	***2443***	***2443***	***2443***	***2448***	SDST125_ta020 (20 x 10)	***2655***	***2655***	***2655***	***2655***	***2655***
SDST100_ta021 (20 x 20)	***3244***	***3244***	***3244***	***3244***	***3245***	SDST125_ta021 (20 x 20)	***3498***	***3498***	***3499***	***3499***	***3499***
SDST100_ta022 (20 x 20)	***3047***	***3047***	***3047***	***3047***	***3047***	SDST125_ta022 (20 x 20)	***3290***	***3290***	***3290***	***3290***	***3290***
SDST100_ta023 (20 x 20)	***3207***	***3207***	***3207***	***3207***	***3207***	SDST125_ta023 (20 x 20)	***3475***	***3475***	***3475***	***3475***	***3475***
SDST100_ta024 (20 x 20)	***3164***	***3164***	***3164***	***3164***	***3164***	SDST125_ta024 (20 x 20)	***3437***	***3437***	***3437***	***3437***	***3437***
SDST100_ta025 (20 x 20)	***3242***	***3242***	***3242***	***3242***	***3242***	SDST125_ta025 (20 x 20)	***3514***	***3514***	***3514***	***3514***	***3514***
SDST100_ta026 (20 x 20)	***3168***	***3168***	***3168***	***3168***	***3168***	SDST125_ta026 (20 x 20)	***3442***	***3442***	***3442***	***3442***	***3442***

SDST100_ta027 (20 x 20)	***3191***	***3191***	***3191***	***3191***	***3191***	SDST125_ta027 (20 x 20)	***3452***	***3452***	***3452***	***3452***	3457
SDST100_ta028 (20 x 20)	***3165***	***3165***	***3165***	***3165***	***3169***	SDST125_ta028 (20 x 20)	***3431***	***3431***	***3431***	***3431***	***3431***
SDST100_ta029 (20 x 20)	***3192***	***3192***	***3192***	***3192***	***3192***	SDST125_ta029 (20 x 20)	***3456***	***3456***	***3456***	***3456***	***3456***
SDST100_ta030 (20 x 20)	***3111***	***3111***	***3111***	***3111***	***3111***	SDST125_ta030 (20 x 20)	***3378***	***3378***	***3378***	***3378***	***3378***
SDST100_ta031 (50 x 5)	**3893**	3944	3939	3928	3932	SDST125_ta031 (50 x 5)	**4226**	4269	4292	4239	4258
SDST100_ta032 (50 x 5)	**4056**	4067	4067	4073	4079	SDST125_ta032 (50 x 5)	**4349**	4399	4398	4396	4391
SDST100_ta033 (50 x 5)	**3900**	3928	3928	3905	3935	SDST125_ta033 (50 x 5)	**4212**	4230	4256	4251	4268
SDST100_ta034 (50 x 5)	**4020**	4049	4056	4035	4045	SDST125_ta034 (50 x 5)	**4356**	4390	4395	4375	4378
SDST100_ta035 (50 x 5)	**4014**	4018	4042	4055	4046	SDST125_ta035 (50 x 5)	**4342**	4364	4353	4358	4362
SDST100_ta036 (50 x 5)	**4073**	4124	4117	4118	4101	SDST125_ta036 (50 x 5)	**4405**	4453	4490	4465	4451
SDST100_ta037 (50 x 5)	**3999**	4038	4043	4019	4013	SDST125_ta037 (50 x 5)	4327	4366	4398	**4322**	4381
SDST100_ta038 (50 x 5)	3966	3985	3979	4005	**3961**	SDST125_ta038 (50 x 5)	**4294**	4336	4313	4298	4307
SDST100_ta039 (50 x 5)	**3808**	3815	3854	3843	3832	SDST125_ta039 (50 x 5)	4145	4170	**4144**	4155	4164
SDST100_ta040 (50 x 5)	4022	4040	4042	**4016**	4053	SDST125_ta040 (50 x 5)	**4341**	4394	4364	4374	4366
SDST100_ta041 (50 x 10)	4812	4819	**4808**	4844	4815	SDST125_ta041 (50 x 10)	**5275**	5281	5283	5283	5301
SDST100_ta042 (50 x 10)	4714	**4710**	4727	4712	4713	SDST125_ta042 (50 x 10)	**5177**	5186	5212	5193	5191
SDST100_ta043 (50 x 10)	**4705**	4725	4715	4725	4718	SDST125_ta043 (50 x 10)	5193	**5158**	5213	5189	5176
SDST100_ta044 (50 x 10)	**4830**	4856	4868	4880	4857	SDST125_ta044 (50 x 10)	**5286**	5291	5301	5301	5294
SDST100_ta045 (50 x 10)	4812	4822	4835	**4805**	4819	SDST125_ta045 (50 x 10)	**5236**	5270	5239	5240	5253
SDST100_ta046 (50 x 10)	**4816**	4839	4835	4824	4844	SDST125_ta046 (50 x 10)	**5262**	5281	5325	5293	5291
SDST100_ta047 (50 x 10)	**4898**	4910	4914	4914	4905	SDST125_ta047 (50 x 10)	**5340**	5370	5362	5365	5366
SDST100_ta048 (50 x 10)	4849	4881	4884	4861	**4843**	SDST125_ta048 (50 x 10)	**5317**	5350	5336	5340	5339
SDST100_ta049 (50 x 10)	**4723**	4732	4757	4778	4761	SDST125_ta049 (50 x 10)	**5194**	5210	5268	5241	5246
SDST100_ta050 (50 x 10)	**4880**	4899	4927	4906	4885	SDST125_ta050 (50 x 10)	**5334**	5344	5393	5353	5355
SDST100_ta051 (50 x 20)	**6074**	6084	6097	6098	6087	SDST125_ta051 (50 x 20)	**6643**	6651	6674	6652	6645
SDST100_ta052 (50 x 20)	**5910**	5918	5923	5939	5960	SDST125_ta052 (50 x 20)	**6489**	6509	6551	6538	6547
SDST100_ta053 (50 x 20)	**5908**	5924	5953	5941	5941	SDST125_ta053 (50 x 20)	6502	6535	**6498**	6514	6536
SDST100_ta054 (50 x 20)	5997	6007	6002	5997	**5990**	SDST125_ta054 (50 x 20)	6587	6586	6618	**6582**	6600
SDST100_ta055 (50 x 20)	5927	**5925**	5929	5932	5935	SDST125_ta055 (50 x 20)	**6495**	6527	6534	6528	6537
SDST100_ta056 (50 x 20)	**5920**	5946	5934	5938	5943	SDST125_ta056 (50 x 20)	**6494**	6548	6570	6500	6522
SDST100_ta057 (50 x 20)	5958	5963	6002	**5952**	5982	SDST125_ta057 (50 x 20)	**6548**	6564	6584	6550	6575
SDST100_ta058 (50 x 20)	**5939**	5956	5962	5978	5965	SDST125_ta058 (50 x 20)	**6519**	6539	6567	6562	6524
SDST100_ta059 (50 x 20)	5948	**5935**	5979	5992	5964	SDST125_ta059 (50 x 20)	**6539**	6568	6611	6556	6585
SDST100_ta060 (50 x 20)	**6026**	6029	6039	6027	6052	SDST125_ta060 (50 x 20)	**6596**	6630	6645	6647	6637
SDST100_ta061 (100 x 5)	7714	7789	7813	**7691**	7778	SDST125_ta061 (100 x 5)	**8339**	8430	8428	8357	8356
SDST100_ta062 (100 x 5)	7610	7684	7650	7650	**7559**	SDST125_ta062 (100 x 5)	8230	8280	8293	8248	**8186**
SDST100_ta063 (100 x 5)	**7539**	7610	7664	7618	7605	SDST125_ta063 (100 x 5)	**8168**	8286	8335	8188	8245
SDST100_ta064 (100 x 5)	7421	7490	7525	**7394**	7450	SDST125_ta064 (100 x 5)	**8005**	8129	8154	8050	8057
SDST100_ta065 (100 x 5)	**7620**	7679	7709	7656	7672	SDST125_ta065 (100 x 5)	**8231**	8294	8356	8280	8338

SDST100_ta066 (100 x 5)	7468	7551	7496	7502	**7466**	SDST125_ta066 (100 x 5)	**8082**	8143	8199	8127	8087
SDST100_ta067 (100 x 5)	**7611**	7702	7722	7677	7640	SDST125_ta067 (100 x 5)	8267	8320	8399	**8250**	8290
SDST100_ta068 (100 x 5)	7424	7551	7470	7439	**7403**	SDST125_ta068 (100 x 5)	**7993**	8143	8182	8088	8138
SDST100_ta069 (100 x 5)	7773	7846	7831	7807	**7769**	SDST125_ta069 (100 x 5)	8393	8469	8518	8409	**8347**
SDST100_ta070 (100 x 5)	7735	7796	7830	**7729**	7766	SDST125_ta070 (100 x 5)	8290	8406	8394	**8247**	8357
SDST100_ta071 (100 x 10)	9201	9202	9245	**9157**	9202	SDST125_ta071 (100 x 10)	**10070**	10072	10149	10108	10124
SDST100_ta072 (100 x 10)	8794	8845	8867	**8788**	8811	SDST125_ta072 (100 x 10)	9631	9675	9671	**9630**	9663
SDST100_ta073 (100 x 10)	9004	9010	9041	**8995**	**8995**	SDST125_ta073 (100 x 10)	**9808**	9940	9997	9924	9887
SDST100_ta074 (100 x 10)	9276	9315	9363	**9268**	9299	SDST125_ta074 (100 x 10)	10168	10161	10172	**10143**	10163
SDST100_ta075 (100 x 10)	9002	9025	9009	**8975**	9019	SDST125_ta075 (100 x 10)	**9852**	9903	9907	9906	9853
SDST100_ta076 (100 x 10)	8689	8713	8736	**8638**	8671	SDST125_ta076 (100 x 10)	**9529**	9638	9602	9537	9588
SDST100_ta077 (100 x 10)	8858	8879	**8805**	8845	8837	SDST125_ta077 (100 x 10)	**9696**	9761	9881	9721	9729
SDST100_ta078 (100 x 10)	9028	9025	9074	8984	**8961**	SDST125_ta078 (100 x 10)	9891	9880	9933	9816	**9795**
SDST100_ta079 (100 x 10)	9133	9177	9191	9145	**9114**	SDST125_ta079 (100 x 10)	10004	10003	10059	**9929**	9950
SDST100_ta080 (100 x 10)	9114	9160	9187	**9096**	9126	SDST125_ta080 (100 x 10)	10013	10094	10078	**10006**	10012
SDST100_ta081 (100 x 20)	10578	**10552**	10627	10568	10572	SDST125_ta081 (100 x 20)	11694	11723	11752	11687	**11622**
SDST100_ta082 (100 x 20)	10535	**10518**	10569	10531	10544	SDST125_ta082 (100 x 20)	11679	11711	11737	**11628**	11640
SDST100_ta083 (100 x 20)	**10552**	10576	10602	10562	10600	SDST125_ta083 (100 x 20)	11701	11600	11780	**11604**	11618
SDST100_ta084 (100 x 20)	**10479**	10516	10596	10514	10518	SDST125_ta084 (100 x 20)	11634	11604	11691	11542	**11498**
SDST100_ta085 (100 x 20)	**10539**	10627	10583	10550	10554	SDST125_ta085 (100 x 20)	11675	11694	11726	11615	**11614**
SDST100_ta086 (100 x 20)	10679	10709	10659	**10655**	10690	SDST125_ta086 (100 x 20)	11740	11770	11808	**11718**	11729
SDST100_ta087 (100 x 20)	10645	10635	10661	**10564**	10593	SDST125_ta087 (100 x 20)	11784	11819	11804	**11755**	11782
SDST100_ta088 (100 x 20)	10794	10766	10863	10767	**10751**	SDST125_ta088 (100 x 20)	11883	11919	11979	11908	**11863**
SDST100_ta089 (100 x 20)	10612	10630	10662	10618	**10608**	SDST125_ta089 (100 x 20)	11731	11753	11788	**11678**	11714
SDST100_ta090 (100 x 20)	10651	10657	10720	10679	**10629**	SDST125_ta090 (100 x 20)	11753	11784	11800	11737	**11710**
SDST100_ta091 (200 x 10)	17307	17312	17408	**17116**	17145	SDST125_ta091 (200 x 10)	18930	18987	19179	**18817**	18842
SDST100_ta092 (200 x 10)	17210	17244	17427	**17047**	17068	SDST125_ta092 (200 x 10)	18876	19002	19059	18720	**18707**
SDST100_ta093 (200 x 10)	17386	17425	17597	17305	**17265**	SDST125_ta093 (200 x 10)	19059	19254	19333	**18933**	**18933**
SDST100_ta094 (200 x 10)	17206	17306	17421	**17061**	17079	SDST125_ta094 (200 x 10)	18934	18987	19077	**18733**	18749
SDST100_ta095 (200 x 10)	17244	17277	17343	17173	**17032**	SDST125_ta095 (200 x 10)	18906	19065	19137	18826	**18767**
SDST100_ta096 (200 x 10)	17022	17017	17222	16905	**16876**	SDST125_ta096 (200 x 10)	18659	18785	18919	18564	**18528**
SDST100_ta097 (200 x 10)	17428	17478	17630	17334	**17306**	SDST125_ta097 (200 x 10)	19118	19296	19437	**18964**	19087
SDST100_ta098 (200 x 10)	17407	17415	17419	17270	**17241**	SDST125_ta098 (200 x 10)	19058	19082	19168	18904	**18893**
SDST100_ta099 (200 x 10)	17194	17182	17327	**17105**	17116	SDST125_ta099 (200 x 10)	18819	18990	19094	**18836**	18844
SDST100_ta100 (200 x 10)	17263	17316	17450	17219	**17196**	SDST125_ta100 (200 x 10)	18793	19017	19065	18759	**18688**
SDST100_ta101 (200 x 20)	19618	19649	19750	19542	**19455**	SDST125_ta101 (200 x 20)	21765	21622	21917	**21611**	21653
SDST100_ta102 (200 x 20)	19816	19809	19916	**19752**	19760	SDST125_ta102 (200 x 20)	21973	22093	22143	**21788**	21882
SDST100_ta103 (200 x 20)	19881	19834	20035	19754	**19673**	SDST125_ta103 (200 x 20)	21975	22103	22150	**21878**	21907
SDST100_ta104 (200 x 20)	19810	19832	19943	**19736**	19761	SDST125_ta104 (200 x 20)	21984	22010	22017	21843	**21798**

SDST100_ta105 (200 x 20)	19589	19651	19686	19522	**19401**	SDST125_ta105 (200 x 20)	21773	21807	21869	21607	**21578**
SDST100_ta106 (200 x 20)	19677	19630	19709	**19579**	19599	SDST125_ta106 (200 x 20)	21829	21830	21939	**21583**	21652
SDST100_ta107 (200 x 20)	19888	19935	19904	19750	**19733**	SDST125_ta107 (200 x 20)	22055	22017	22083	**21873**	21904
SDST100_ta108 (200 x 20)	19826	19865	19956	**19680**	19700	SDST125_ta108 (200 x 20)	21902	21954	21991	21840	**21752**
SDST100_ta109 (200 x 20)	19757	19837	19923	19664	**19650**	SDST125_ta109 (200 x 20)	21821	21971	22081	21863	**21737**
SDST100_ta110 (200 x 20)	19813	19839	19961	19700	**19680**	SDST125_ta110 (200 x 20)	21975	22001	22099	**21798**	21804
SDST100_ta111 (500 x 20)	46716	46758	47382	**46460**	46517	SDST125_ta111 (500 x 20)	52021	52060	52622	51511	**51429**
SDST100_ta112 (500 x 20)	47291	47499	47910	**46893**	47053	SDST125_ta112 (500 x 20)	52380	52763	53126	52110	**52016**
SDST100_ta113 (500 x 20)	46964	47171	47512	46764	**46703**	SDST125_ta113 (500 x 20)	52110	52326	52914	51749	**51707**
SDST100_ta114 (500 x 20)	47021	47177	47546	46792	**46790**	SDST125_ta114 (500 x 20)	52194	52388	52834	**51776**	51902
SDST100_ta115 (500 x 20)	46994	47028	47505	**46553**	46739	SDST125_ta115 (500 x 20)	51932	52287	52696	**51599**	51630
SDST100_ta116 (500 x 20)	47074	47277	47676	**46862**	46979	SDST125_ta116 (500 x 20)	52269	52710	53049	51995	**51895**
SDST100_ta117 (500 x 20)	46889	46794	47340	**46477**	46569	SDST125_ta117 (500 x 20)	51917	52162	52644	51564	**51430**
SDST100_ta118 (500 x 20)	47183	47150	47602	46862	**46710**	SDST125_ta118 (500 x 20)	52185	52629	53216	52085	**51888**
SDST100_ta119 (500 x 20)	46636	46842	47131	46345	**46293**	SDST125_ta119 (500 x 20)	51746	52004	52555	51495	**51457**
SDST100_ta120 (500 x 20)	46902	46984	47458	**46636**	46655	SDST125_ta120 (500 x 20)	52118	52388	52910	51873	**51838**

A-2) Best known solutions for Total Flow time Minimization Problem

Problem Instance	SDST10	SDST50	SDST100	SDST125
ta001 (20 x 5)	14875	17828	21490	23452
ta002 (20 x 5)	15998	18911	22207	23695
ta003 (20 x 5)	14261	17329	20666	22394
ta004 (20 x 5)	16230	19056	22601	24165
ta005 (20 x 5)	14436	17076	20433	21985
ta006 (20 x 5)	14070	17051	20216	21765
ta007 (20 x 5)	14372	17298	20797	22729
ta008 (20 x 5)	14736	18020	21395	23236
ta009 (20 x 5)	15286	18273	21396	23133
ta010 (20 x 5)	13759	16721	19708	21291
ta011 (20 x 10)	21900	25650	30259	32737
ta012 (20 x 10)	23435	26846	31570	33956
ta013 (20 x 10)	20722	23889	28619	31033
ta014 (20 x 10)	19517	22852	26910	29222
ta015 (20 x 10)	19565	23084	27716	30175
ta016 (20 x 10)	20088	23324	27666	29915
ta017 (20 x 10)	19178	22868	27568	29743
ta018 (20 x 10)	21127	24525	29427	31869
ta019 (20 x 10)	21270	24935	29298	31555
ta020 (20 x 10)	22198	25845	30488	32848
ta021 (20 x 20)	34733	38694	43954	46470
ta022 (20 x 20)	32510	36456	41728	44463
ta023 (20 x 20)	34801	38507	42998	45401
ta024 (20 x 20)	32628	36562	42034	45066
ta025 (20 x 20)	35459	39008	44363	47154
ta026 (20 x 20)	33428	37713	43305	46166
ta027 (20 x 20)	33837	37894	43292	46160
ta028 (20 x 20)	33306	37279	42796	45746
ta029 (20 x 20)	34581	38665	44209	47119
ta030 (20 x 20)	33143	36832	41804	44527
ta031 (50 x 5)	69903	85418	103194	112571
ta032 (50 x 5)	73318	90360	108652	117198
ta033 (50 x 5)	68714	84102	102162	111219
ta034 (50 x 5)	74157	90329	107981	115943
ta035 (50 x 5)	74515	89160	106639	115121
ta036 (50 x 5)	72286	88902	107760	116973

ta037 (50 x 5)	71784	87879	106336	116011
ta038 (50 x 5)	69802	86132	104585	113005
ta039 (50 x 5)	68123	83998	102457	111372
ta040 (50 x 5)	74171	89003	107093	116298
ta041 (50 x 10)	93052	113780	138106	150780
ta042 (50 x 10)	89228	109376	134454	146253
ta043 (50 x 10)	85978	106836	131904	145264
ta044 (50 x 10)	92596	113710	138063	150532
ta045 (50 x 10)	92754	113856	138972	150913
ta046 (50 x 10)	92815	112872	137306	150399
ta047 (50 x 10)	94929	114871	139221	150749
ta048 (50 x 10)	93124	113130	136888	149844
ta049 (50 x 10)	91837	111361	135462	147495
ta050 (50 x 10)	94232	114575	138901	151539
ta051 (50 x 20)	132614	155781	185963	201756
ta052 (50 x 20)	125415	148907	179817	195646
ta053 (50 x 20)	122846	147175	178175	194389
ta054 (50 x 20)	127044	151147	181739	196666
ta055 (50 x 20)	124894	148717	179699	195707
ta056 (50 x 20)	127046	149901	180146	196185
ta057 (50 x 20)	129466	152607	182702	199223
ta058 (50 x 20)	129205	152358	182830	198377
ta059 (50 x 20)	128117	151497	180053	196638
ta060 (50 x 20)	130561	154273	184256	201101
ta061 (100 x 5)	275439	331503	401657	433446
ta062 (100 x 5)	264943	322771	383133	419408
ta063 (100 x 5)	259815	316880	383644	420447
ta064 (100 x 5)	248923	307457	378426	414200
ta065 (100 x 5)	262116	319807	390014	421523
ta066 (100 x 5)	254835	313885	382968	415877
ta067 (100 x 5)	262375	321259	388349	425422
ta068 (100 x 5)	253692	312864	382403	416271
ta069 (100 x 5)	270141	328563	397009	430184
ta070 (100 x 5)	264365	322097	390145	425174
ta071 (100 x 10)	325603	400663	497754	542145
ta072 (100 x 10)	300505	379595	474716	522170
ta073 (100 x 10)	313833	391991	485623	533185
ta074 (100 x 10)	328630	407329	504444	552165
ta075 (100 x 10)	310948	391514	488541	528558

ta076 (100 x 10)	296344	374211	472840	520832
ta077 (100 x 10)	305371	385063	478189	529056
ta078 (100 x 10)	317042	394881	491548	534020
ta079 (100 x 10)	327521	402094	496114	546294
ta080 (100 x 10)	316921	397252	493400	540241
ta081 (100 x 20)	394418	484783	600212	661519
ta082 (100 x 20)	401716	494877	609845	666897
ta083 (100 x 20)	399318	492412	608238	669137
ta084 (100 x 20)	402653	492503	604868	663480
ta085 (100 x 20)	397236	487625	604096	666136
ta086 (100 x 20)	400348	492664	606046	671086
ta087 (100 x 20)	403511	494385	608838	672860
ta088 (100 x 20)	412752	502498	617446	679579
ta089 (100 x 20)	402837	492970	607588	669486
ta090 (100 x 20)	407864	499018	607456	671957
ta091 (200 x 10)	1146682	1441896	1807352	1986298
ta092 (200 x 10)	1138406	1429058	1792087	1991074
ta093 (200 x 10)	1149346	1445466	1813283	2004644
ta094 (200 x 10)	1131227	1429536	1785374	1976164
ta095 (200 x 10)	1137508	1432359	1812037	1981399
ta096 (200 x 10)	1112680	1415515	1774099	1954304
ta097 (200 x 10)	1158724	1460000	1836110	2006684
ta098 (200 x 10)	1147599	1451753	1816781	2004683
ta099 (200 x 10)	1126810	1427237	1789820	1971411
ta100 (200 x 10)	1132527	1435648	1802622	1984071
ta101 (200 x 20)	1341485	1694819	2136787	2378331
ta102 (200 x 20)	1359134	1719013	2162691	2393149
ta103 (200 x 20)	1380766	1735784	2179383	2406760
ta104 (200 x 20)	1351768	1710121	2168961	2398177
ta105 (200 x 20)	1340365	1695159	2145547	2364488
ta106 (200 x 20)	1344617	1698398	2151639	2369067
ta107 (200 x 20)	1357790	1706425	2163931	2397594
ta108 (200 x 20)	1354229	1712888	2166485	2398013
ta109 (200 x 20)	1345479	1705958	2155966	2379548
ta110 (200 x 20)	1363190	1719065	2167126	2394268
ta111 (500 x 20)	7356002	9512013	12201447	13581758
ta112 (500 x 20)	7457075	9614520	12314837	13660282
ta113 (500 x 20)	7394925	9560189	12266597	13628437
ta114 (500 x 20)	7443908	9584384	12231835	13641375

ta115 (500 x 20)	7419538	9573874	12228518	13558514
ta116 (500 x 20)	7400282	9572786	12229388	13583976
ta117 (500 x 20)	7339338	9532073	12179786	13587496
ta118 (500 x 20)	7429258	9610969	12299046	13647700
ta119 (500 x 20)	7359471	9549719	12195424	13531394
ta120 (500 x 20)	7398971	9564768	12312740	13646101