Effect of problem size on the performance of constructive algorithms for flow shop scheduling problems with sequence dependent setup times

Rajesh Vanchipura

Department of Mechanical Engineering,
Govt. Engineering College Thrissur,
Thrissur-680009, Kerala, India
E-mail: rajeshvanchipura@gmail.com

R. Sridharan*

Department of Mechanical Engineering,
National Institute of Technology Calicut,
Calicut-673601, Kerala, India
E-mail: sreedhar@nitc.ac.in
*Corresponding author

Abstract: This paper focus on the scheduling problem of a flow shop operating in a sequence dependent setup time (SDST) environment. Two constructive algorithms of contrasting characteristics are analysed for the performance with respect to change in problem size; the first algorithm is processing time-based and the second algorithm is setup time-based. The problem size is characterised by the variables namely, number of jobs and number of machines. An extensive performance analysis of the two constructive algorithms has been carried out using 960 SDST flow shop benchmark problem instances. The graphical analysis of the results reveals the correlation between the relative performance of the algorithms and problem size. The study shows that the performance of the setup time-based algorithm increases with increase in number of jobs and decreases with increase in number of machines. The coefficient of variation analysis is used to investigate the performance variation of the algorithm with change in number of machines. The analysis reveals that as the number machines increases, the coefficient of variation of the summed setup time matrix decreases which causes the change in performance of the setup time-based algorithm.

Keywords: flow shop; sequence dependent setup time; SDST; heuristic algorithm; coefficient of variation.

Reference to this paper should be made as follows: Vanchipura, R. and Sridharan, R. (2014) 'Effect of problem size on the performance of constructive algorithms for flow shop scheduling problems with sequence dependent setup times', *Int. J. Internet Manufacturing and Services*, Vol. 3, No. 3, pp.204–225.

Biographical notes: Rajesh Vanchipura is an Assistant Professor at Government Engineering College, Thrissur, Kerala. He received his MTech degree in Industrial Engineering and Management from the National Institute of Technology Calicut. He also holds MBA degree in Operations and Marketing Management from the Amrita School of Business, Amrita Vishwa Vidyapeetham, Coimbatore. His research interests are in the areas of development of algorithms for scheduling problems, development of

metaheuristics, operations management, and multi-criteria decision making. He has published papers in refereed international journals and proceedings of international and national conferences. Currently, he is pursuing research in the area of development and analysis of algorithms for flow shop scheduling with sequence dependent setup time.

R. Sridharan is a Professor of Industrial Engineering in the Department of Mechanical Engineering at the National Institute of Technology Calicut, India. He received his PhD in 1995 from the Department of Mechanical Engineering at the Indian Institute of Technology Bombay, India. His research interests include modelling and analysis of decision problems in supply chain management, job shop production systems and flexible manufacturing systems. He has published papers in refereed international journals and proceedings of international and national conferences. Currently, he is Professor and Head of the Department of Mechanical Engineering at the National Institute of Technology Calicut, Kerala, India.

This paper is a revised and expanded version of a paper entitled 'Effect of problem size on the performance of constructive algorithms for flow shop scheduling problems with SDSTs', presented at 4th International and, 25th AIMTDR Conference, Jadavpur University, 14–17 December 2012.

1 Introduction

Flow shop scheduling is an extensively researched combinatorial optimisation problem. Generally, all scheduling problems invariably involve setup times apart from the actual processing times of jobs. The setup time includes time required for preparing the machine, tools, process, or the work itself for further processing. For example, a finite amount of time is incurred in positioning work-in-process material, obtaining tools, loading the tool, return tooling, cleanup, etc. In most situations, the setup time is added to the processing time, in order to reduce the complexity of the problem. However, there are many applications which require explicit treatment of setup time (Allahverdi et al., 1999). The present study addresses the problem of scheduling a flow shop operating in a sequence dependent setup time (SDST) environment. Consideration of SDST separately in flow shop scheduling problem certainly makes the problem close to reality. Its benefits can be seen in the production floor in terms of accurate completion time estimate and accurate delivery time promises.

The problem considered in the present research involves scheduling a set of n jobs which are available for processing on m machines with SDST, i.e., the setup time of a job on a machine is dependent on the previously processed job on the same machine. The objective of the problem is to find the permutation schedule which minimises the makespan assuming no pre-emption of operations. In the literature, the problem is known as flow shop scheduling with SDST. The presence of SDST is observed in many practical situations across different industry. Such situations are found in various shop configurations and environments such as:

1 in printing industry, where the machine cleaning depends on the colour of the current and immediately following orders

2 in textile industry applications, where setup for weaving and dying operations depends on the job sequence

3 in the container and bottle industry, where the settings change depending on the sizes and shapes of the container

4 in manufacturing of chemical compounds, where the extent of the cleaning depends on both the chemical most recently processed and the chemical about to be processed (Allahverdi et al., 1999, 2008; Eren, 2010).

Similar situations arise in pharmaceutical, food processing, metal processing, paper manufacturing, and many other industries. In these situations, SDST plays a major role and must be considered explicitly while modelling the problem.

In industrial scheduling, the number of jobs requiring processing and number machines available for processing often vary with the situations based on the typical nature and scale of the manufacturing shops. The number of jobs (n) and the number of machines (m) are the two parameters that define the problem size ($n \times m$). It is very much possible that as the problem size parameters change, the performance of the heuristics used for scheduling also may vary. The motivation of the present study is to find the effect of problem size on the relative performance of heuristics. The size of a problem increases when there is an increase in the number jobs or an increase in the number of machines or an increase in both. Thus, in the present study, these two problem size varibles are analysed seperately with respect to the performance of two existing heuristics. The first constructive heuristic algorithm, NEHRB heuristic developed by Rios-Mercado and Bard (1998) considers only processing time for constructing the sequence. The second constructive heuristic, fictitious job setup ranking algorithm (FJSRA) uses setup time data along with processing time data for constructing the sequence (Vanchipura and Sridharan, 2012). Thus, the two heurstics differ in the methodology of constructing the sequence. The performance of these heuristics is analysed for 12 different size problems at eight different levels of setup time (96 groups of problems). The benchmark problems developed by Vanchipura and Sridharan (2012) for SDST flow shop are used in the present study for experimentation. The results of these exhaustive experiments are used for finding the relative performance improvement. Initially, a graphical analysis is carried out to investigate performance variation. A coefficient of variation analysis of the setup times is used to explain the variation in performance of the two algorithms with increasing number of machines. To the best knowledge of the authors, the performance of different scheduling algorithms for the SDST flow shop under varying problem size parameters is an area unexplored by the researchers. The present study fills this research gap.

The rest of the paper is organised as follows. Section 2 provides a review of the relevant literature. Section 3 describes the problem formulation for SDST flow shop. Section 4 explains the solution methodology. The details of experimentation are provided in Section 5. The results and discussion are provided in Section 6. Concluding remarks and scope for further research are presented in Section 7.

2 Review of relevant literature

Allahverdi et al. (1999) present a review of flow shop scheduling problem with SDST. Later, Allahverdi et al. (2008) provide an extensive review of the scheduling literature on models with setup times. Gupta (1986) proves the NP-completeness of the SDST flow shop scheduling problem. Due to this complexity, the SDST version is relatively less explored compared to the general flow shop scheduling problem. Metaheuristics are found to be the suitable methodologies for this category of problems. Parthasarathy and Rajendran (1997) develop a simulated annealing (SA) algorithm to minimise mean weighted tardiness for a flow shop with SDSTs. They use a method namely, random insertion perturbation scheme for generating the neighbourhood solution sequence for SA algorithm. The SA heuristic is evaluated against existing heuristics and the results of computational evaluation reveal that the proposed heuristic performs much better. Ruiz et al. (2005) proposed a heuristic based on genetic algorithms. They use randomised NEHRB heuristic in two occasions:

1 for initial generating initial population

2 for restarting the population in case of premature convergence which helps the GA to perform better compared to the normal GA.

Gajpal et al. (2006) present an ant colony optimisation algorithm for flow shop scheduling with sequence dependent setups for the makespan objective. An existing ant colony algorithm and the proposed ant colony algorithm are compared with two existing heuristics. Extensive computational investigation reveals that the proposed ant colony algorithm provides promising and better results, as compared to the solutions obtained using the existing ant colony algorithm and the existing heuristics. Apart from these metaheuristics, there are a few research works that develop local-search and greedy heuristics. Ruiz and Stutzle (2008) present two simple local search-based iterated greedy algorithms. Two different optimisation objectives namely, minimisation of makespan and minimisation of total weighted tardiness are considered. Extensive experiments and statistical analyses demonstrate that despite their simplicity, the iterated greedy algorithms emerge as state-of-the-art methods for both the objectives. Tseng et al. (2005) develop a penalty-based heuristic algorithm and compare their heuristic with an existing index heuristic algorithm. Rajendran and Ziegler (2003) proposed two heuristics for the SDST flow shop scheduling problem with a combination of two objectives namely, weighted flow time and weighted tardiness. They present a heuristic in which two single objective constructive algorithms are used to generate two separate sequences and the best sequence is selected.

There are variations of the SDST flow shop namely, flexible flow shop with SDST, no wait flow shop with SDST, which also find application of such intelligent metaheuristics. Ruiz and Morato (2006) present a heuristic based on genetic algorithm for scheduling a hybrid flow shop with SDST. Logendran et al. (2006) develop three tabu search-based algorithms for the same problem. Zandieh and Gholami (2009) present an immune algorithm for a hybrid SDST flow shop with machines that suffer from stochastic breakdowns. Wang et al. (2011) used a SA approach for a hybrid SDST flow shop with the objective of minimising makespan. A water flow-like approach is used by Pargar and Zandieh (2012) in their study to minimise the weighted sum of makespan and total tardiness for a hybrid SDST flow shop. Varmazyar and Salmasi (2012) present

several metaheuristic algorithms based on tabu search and imperialistic competitive algorithm for SDST flow shop scheduling problem with the objective of minimising the number of tardy jobs. Wang et al. (2011) used a SA approach for a hybrid SDST flow shop with the objective of minimising makespan.

Constructive algorithms for SDST flow shop are found rarely in the literature. Rios-Mercado and Bard (1998) present a constructive heuristic for the SDST flow shop scheduling problem with the objective of minimisation of makespan. The heuristic is an extension of the well known constructive algorithm namely, NEH heuristic for general flow shop developed by Nawaz et al. (1983). The heuristic is known as NEHRB heuristic. It uses the processing time data for constructing the sequence. Vanchipura and Sridharan (2012) present a determistic constructive algorithm known as FJSRA which uses setup time data along with processing time data for constructing the sequence. The present study focuses on the analysis of the performance of NEHRB and FJSRA under different size of SDST scheduling problem.

3 Problem formulation

A flow shop scheduling problem involves a set of n jobs to be processed on a set of m machines, with all the jobs processed in the same order. It is assumed that setup time is sequence dependent, i.e., the setup time of a job on a machine is determined by knowing the predecessor of the job. The objective of the problem is to find a sequence for the processing of the jobs on the machines so that the total completion time or makespan of the schedule is minimised.

Notations

n	total number of jobs to be scheduled
i	index of machine
m	total number of machines in the flow shop
j	index of job
p_{ij}	processing time of job j on machine i
s_{ijk}	setup time on machine i, when job k is preceded by job j
σ	ordered set of jobs already scheduled, out of n jobs; partial sequence
n_σ	number of jobs in the partial sequence, σ
$q(\sigma, i)$	completion time of partial sequence σ on machine i
$q(\sigma j, i)$	completion time of job j on machine i, when job j is appended to the partial sequence σ.

For calculating the start and completion times of jobs on machines in the permutation flow shop, recursive equations are used as follows.

The completion time of σj on machine i is determined using the following recursive equation:

$$q(\sigma j, i) = \max\left\{q(\sigma, i) + s_{ijk}, q(\sigma j, i-1)\right\} + p_{ij} \quad (1)$$

where q (Φ, i) = 0 and $q(\sigma, 0) = 0$, for all σ and i, with Φ denoting a null schedule. It is assumed that s_{ijk} exists for all jobs where $j = \Phi$ for all machines. It is also assumed that setup of a machine can be done without the job being available at the machine.

The flow time of job j, C_j, is given by:

$$C_j = q(\sigma j, m) \tag{2}$$

When all the jobs are scheduled, the makespan M is obtained as follows:

$$M = \max\left\{C_j, j = 1, 2, \ldots, n\right\} \tag{3}$$

4 Solution methodology

Two constructive algorithms are considered in this study. These algorithms are NEHRB (Rios-Mercado and Bard, 1998) and FJSRA (Vanchipura and Sridharan, 2012). The FJSRA heuristic uses setup time data along with the processing time data for obtaining the sequence. The algorithms are tested on the benchmark problem developed by Vanchipura and Sridharan (2012) which includes 12 different size problems. The performance variation of the two constructive algorithms are analysed with the problem size parameters (number of jobs and number of machines). The two existing constructive algorithms are described in the following sub-sections.

4.1 NEHRB heuristic algorithm

This heuristic is based on NEH heuristic (Nawaz et al., 1983) for flow shop scheduling problem. The procedure adopted in NEH heuristic consists of inserting a job into the best available position of the partial sequence. Rios-Mercado and Bard (1998) extend the NEH heuristic to solve the SDST flow shop scheduling problem. This procedure is known as NEHRB heuristic. This algorithm uses multiple iterations in which there is a partial schedule at each iteration. The largest processing time rule is applied to select the next job to be inserted into the partial sequence. The makespan obtained is compared at each stage and the best sequence is selected. This procedure is continued till the final sequence is obtained. An illustration NEHRB heuristic is provided in Appendix 1.

4.2 Fictitious job setup ranking algorithm

FJSRA is developed by giving importance to setup time along with processing time (Vanchipura and Sridharan, 2012). The principle behind this algorithm is that as the setup time between jobs is reduced, the makespan also will get reduced. FJSRA is based on the concept of fictitious jobs. A pair of jobs with minimum setup time between them is treated as a fictitious job and the processing time of the fictitious job is the sum of the processing time of the pair of jobs involved in the fictitious job. The algorithm starts with the summed setup time matrix which is the sum of setup time matrices for all the machines. The summed setup time matrix shows the sum of all setup time on all machines of all two-job combinations. This matrix gives a measure of setup time between different combinations of jobs. From the setup time matrix, the pairs of jobs with minimum value of summed setup time are identified as the fictitious jobs. The first two

fictitious jobs with maximum fictitious processing time form the partial sequence. Then, the fictitious job with the next highest summed up processing time is considered and inserted in all possible positions in the partial sequence. Partial makespan values of all sequences are found out and the sequence with minimum makespan value is made the next partial sequence. The algorithm repeats these steps to get the best sequence. An illustration of the heuristic is provided in Appendix 2.

5 Experimentation

Experimentation is carried out on the benchmark problems developed for flow shop scheduling problems with SDST (Vanchipura and Sridharan, 2012). The benchmark problems involve 12 different sizes of problems at eight different levels of setup time which forms 960 problem instances (96 groups of problems with 10 instances in each group). The different sizes of problem are 20 jobs × 5 machines, 20 × 10, 20 × 20, 50 × 5, 50 × 10, 50 × 20, 100 × 5, 100 × 20, 100 × 20, 200 × 10, 200 × 20 and 500 × 20. The setup time data has been generated for each problem size using eight different proportions of maximum setup time to maximum processing time. These proportions are 5%, 10%, 25%, 50%, 75%, 100%, 125% and 150%. Corresponding to these proportions, setup times are generated using eight different uniform distributions viz., U(1, 5), U(1, 10), U(1, 25), U(1, 50), U(1, 75), U(1, 100), U(1, 125) and U(1, 150). The constructive algorithms (NEHRB and FJSRA) are coded in MATLAB and tested using the 960 problem instances developed.

6 Results and discussion

The makespan values are obtained for all the 12 problem sizes at eight different levels of setup time. Average makespan obtained using each of the heuristics, NEHRB and FJSRA, for each of the problem sizes are presented in Table 1.

Table 1 clearly shows that for smaller problems, NEHRB is found to be the better algorithm and FJSRA is better for larger problems. The purpose of this study is to analyse this performance variation with problem size. The makespan results obtained shows that the makespan values are found to vary drastically with change in problem size and setup time level. This necessitates a normalisation of the makespan results for the purpose of comparison. This can be done by determining the Relative Performance Improvement (RPI) for each of the problem instances. Since RPI is the percentage improvement in makespan, it can be used to compare groups of problems which have entirely different makespan values. The Relative Performance Improvement used for the comparison of the heuristic algorithms among groups is computed as follows.

$$\text{Relative Performance Improvement} = \frac{NEHRB\ C_{\max} - FJSRA\ C_{\max}}{EHRB\ C_{\max}}$$

where

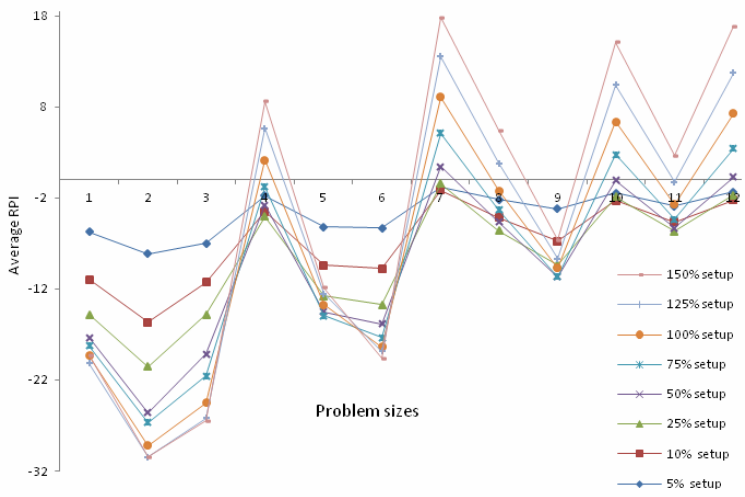
$NEHRB\ C_{\max}$ makespan found using NEHRB heuristic

$FJSRA\ C_{\max}$ makespan found using FJSRA heuristic.

Table 1 Average makespan for NEHRB and FJSRA for different problem sizes

Problem size	*Average makespan at different setup time percentages*															
	5%		*10%*		*25%*		*50%*		*75%*		*100%*		*125%*		*150%*	
	NEHRB	*FJSRA*	*NEHRB*	*FJSRA*	*NEHRB*	*FJSRA*	*NEHRB*	*FJSRA*	*NEHRB*	*FJSRA*	*NEHRB*	*FJSRA*	*NEHRB*	*FJSRA*	*NEHRB*	*FJSRA*
SDST 20 × 5	1,295	1,368	1,346	1,416	1,462	1,659	1,651	1,692	1,865	1,883	2,065	2,087	2,267	2,285	2,448	2,448
SDST 20 × 10	1,632	1,763	1,683	1,808	1,818	1,904	2,053	2,153	2,310	2,333	2,547	2,611	2,817	2,850	3,086	3,086
SDST 20 × 20	2,367	2,531	2,427	2,529	2,589	2,681	2,829	2,950	3,106	3,179	3,392	3,493	3,703	3,766	4,025	4,035
SDST 50 × 5	2,843	2,901	2,931	2,991	3,200	3,216	3,644	3,591	4,083	3,994	4,489	4,342	4,978	4,780	5,390	5,210
SDST 50 × 10	3,241	3,407	3,341	3,481	3,664	3,787	4,225	4,298	4,742	4,761	5,344	5,278	5,855	5,784	6,369	6,327
SDST50 × 20	4,069	4,284	4,202	4,387	4,547	4,729	5,167	5,276	5,786	5,871	6,471	6,539	7,126	7,160	7,864	7,792
SDST 100 × 5	5,430	5,474	5,594	5,615	6,095	6,042	6,895	6,745	7,724	7,409	8,512	8,149	9,275	8,811	10,050	9,579
SDST 100 × 10	5,961	6,108	6,158	6,289	6,735	6,839	7,769	7,690	8,773	8,627	9,810	9,589	10,844	10,487	11,891	11,435
SDST 100 × 20	6,889	7,129	7,115	7,384	7,784	8,005	8,951	9,078	10,122	10,134	11,340	11,229	12,618	12,446	13,895	13,572
SDST 200 × 10	11,150	11,311	11,557	11,679	12,739	12,670	14,647	14,342	16,568	16,031	18,416	17,691	20,392	19,434	22,317	21,164
SDST 200 × 20	12,183	12,562	12,688	12,920	13,995	14,142	16,215	16,140	18,461	18,274	20,685	20,314	23,072	22,438	25,391	24,552
SDST 500 × 20	28,047	28,409	29,124	29,391	32,388	32,213	37,676	36,927	42,976	41,634	48,396	46,531	53,742	51,361	59,106	56,075

Figure 1 Average relative performance improvement for different problem sizes (see online version for colours)



The RPI values are determined for each problem instance and the average RPI values for each group are calculated by taking the average of the RPI values of the respective problem instances in the group. The study is focused on finding the effect of problem size on the performance of the two algorithms. Hence, average RPI values are obtained for each of the problem sizes for the comparison of the performance variation of the heuristic algorithms. Figure 1 shows the average RPI values for 12 problem sizes.

Figure 1 shows that for larger problems, the improvement is in the positive range indicating a relatively better performance of the setup time-based algorithm FJSRA. Figure 1 also shows that the RPI of the setup time-based algorithm increases as the problem size increases. The relative improvement with respect to problem size shows a zigzag pattern. This is due to the fact that the problem size parameters, namely, number of jobs and number of machines, are not increasing simultaneously and continuously. For example, for the first three problems, the number of jobs remains constant and the number of machines increases. Figure 1 shows that for every increase in the number of jobs, there is a sudden jump in improvement. As the number of machines increases, there is a gradual decrease in improvement. The above observation shows that the relative performance is correlated with the problem size parameters, namely, number of jobs and number of machines. This aspect of the problem necessitates further graphical analysis based on RPI values which is presented under the follwing cases.

1 relative performance with increasing number of machines

2 relative performance with increasing number of jobs.

6.1 Relative performance with increasing number of machines

Here, the different problem sizes in the benchmark set are segregated in such a way that problem groups with the same number of machines form a set of problem groups. For example, the three problems with size such as 20 × 5, 20 × 10 and 20 × 20 form a set of 20 job problems. In a similar manner, there are sets of 50 job problems, 100 job problems

and 200 job problems. The average RPI values are plotted for different sets of problem groups as shown in Figures 2 to 5.

Figure 2 RPI plot for 20 jobs with increasing number of machine (see online version for colours)

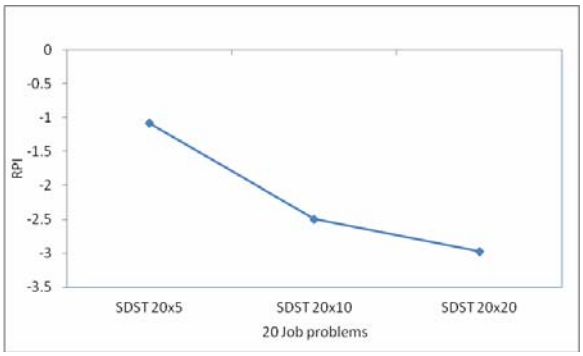


Figure 3 RPI plot for 50 jobs with increasing number of machines (see online version for colours)

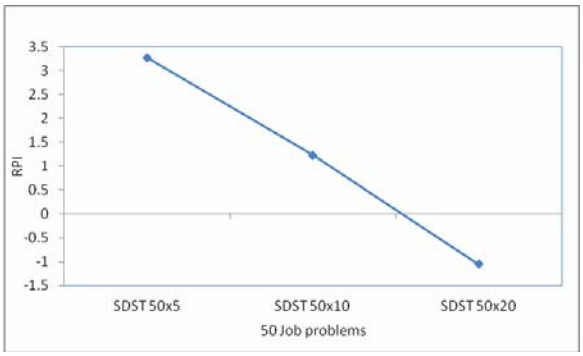


Figure 4 RPI plot for 100 jobs with increasing number of machines (see online version for colours)

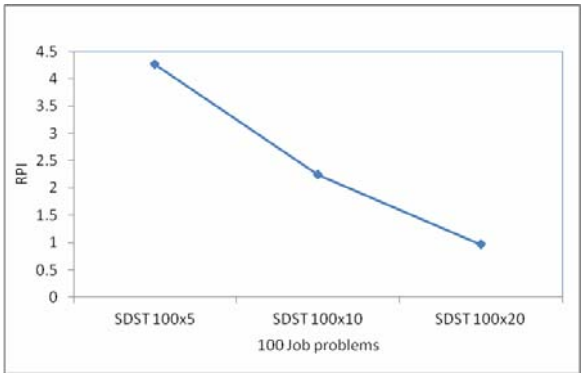
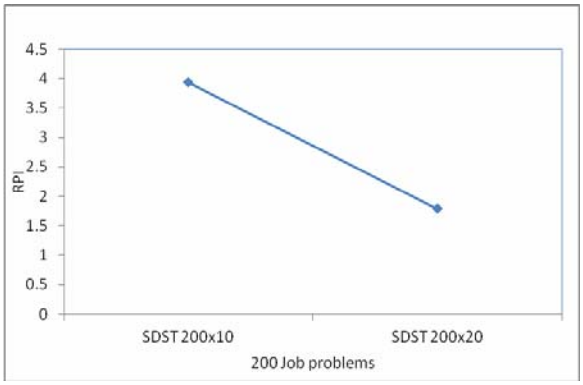


Figure 5 RPI plot for 200 jobs with increasing number of machines (see online version for colours)



From Figures 2 to 5, it is evident that there is a decrease in RPI with an increase in the number of machines. This shows that as the problem size increases, the relative improvement obtained for FJSRA decreases. For larger size problems, it is observed that the improvement is in the positive range even though the improvement is decreasing (Figures 4 and 5). The decreasing relative performance of FJSRA algorithm is due to the decreased significance of setup time as the number of machines increases. This can be further explained with help of coefficient of variation of the setup time matrices. The FJSRA algorithm starts with summed setup time matrix which is the basis for determining the best sequence. In other words, if there is more variation in summed setup time, the FJSRA algorithm performs well and a lesser variation in summed setup time matrix results in the inferior performance of the algorithm. Hence, it is required to find out the variation involved in the summed setup time matrix. However, as number of machines increases, the number of setup time matrices also increases, resulting in the higher means of elements of setup time matrix. Here, it is obvious that as the problem size increases, the mean of the elements of summed setup time matrix also increases. The coefficient of variation is a more useful measure in order to study the variation involved in summed setup time matrix. The coefficient of variation is a dimensionless number. Hence, when comparing between data sets with different units or widely different means, the coefficient of variation is recommended for comparison instead of the standard deviation. In the present study, it is found that the coefficient of variation decreases with increasing number of machines. Figure 6 shows the decreasing trend of coefficient of variation for problem sizes with 20 jobs and 5, 10, and 20 machines (i.e., for SDST 20 × 5, SDST 20 × 10 and SDST 20 × 20 problem sizes).

The FJSRA heuristic uses summed setup time matrix for determining the best pairs of jobs. As the number of machines increases, the coefficient of variation of setup time in the summed setup time matrix decreases. Figure 6 shows the decreasing trend of coefficient of variation for problem sizes with 20 jobs. The decreasing coefficient of variation reflects the decreasing variation in summed setup time matrix elements which in turn results in the decreased performance of the FJSRA algorithm as shown in Figure 2. Similarly, coefficient of variation is computed for problem sizes having 50 jobs and 100 jobs as shown in Figures 7 and 8.

Figure 6 Coefficient of variation for 20 jobs problems with increasing number of machines (see online version for colours)

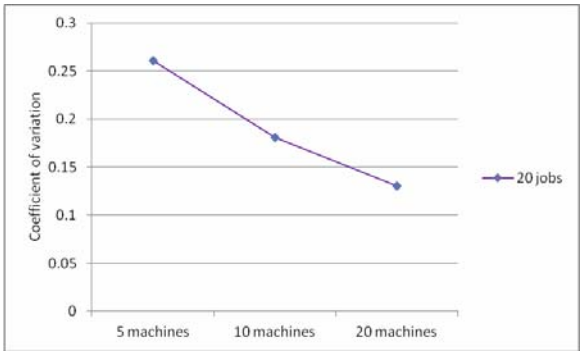


Figure 7 Coefficient of variation for 50 jobs problems with increasing number of machines (see online version for colours)

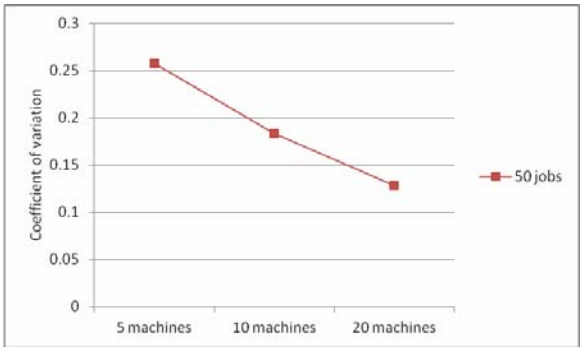
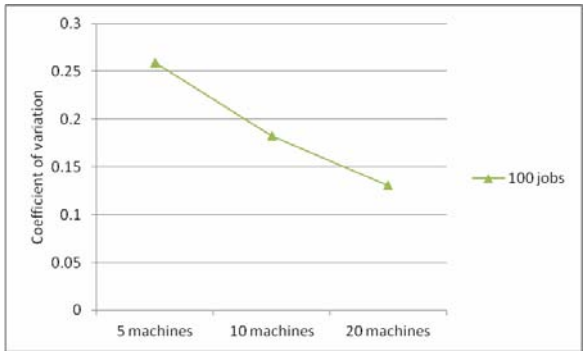


Figure 8 Coefficient of variation for 100 jobs problems with increasing number of machines (see online version for colours)



Both the Figures 7 and 8 show the decreasing trend of coefficient of variation which result in the decreasing performance of FJSRA as observed in Figures 3 and 4 respectively.

6.2 Relative performance with increasing number of jobs

For this analysis, the problems in the benchmark set are segregated in such a way that problems with the same number of machines form a set of problem group. Thus, the three groups of problems formed are five machine problems (SDST 20 × 5, SDST 50 × 5 and SDST 100 × 5), ten machines problems (SDST 20 × 10, SDST 50 × 10, SDST 100 × 10 and SDST 200 × 10) and 20 machines problems (SDST 20 × 20, SDST 50 × 20, SDST 100 × 20, SDST 200 × 20 and SDST 500 × 20). The average RPI values are ploted for each of the problems groups seperately as shown in Figures 9, 10 and 11 for the 5 machines, 10 machines and 20 machines problem groups respectively.

Figures 9, 10 and 11 show the increasing trend of the average RPI values indicating the better performance of the FJSRA with increasing number of jobs. When the number of jobs increases, the setup time becomes more significant which leads to an improved performance of setup time-based algorithms. This is evident from the pattern of variation depicted in Figures 9 to 11. Since FJSRA gives more prominence to setup time, the algorithm provides better performance. The existing algorithm NEHRB gives relatively less importance to setup time and this is the reason for the inferior performance of the NEHRB algorithm in comparison with FJSRA for problems with lesser number of jobs

Figure 9 RPI plot for five machines with increasing number of job (see online version for colours)

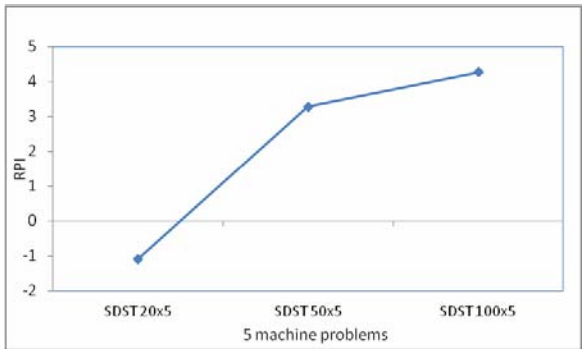


Figure 10 RPI plot for ten machines with increasing number of job (see online version for colours)

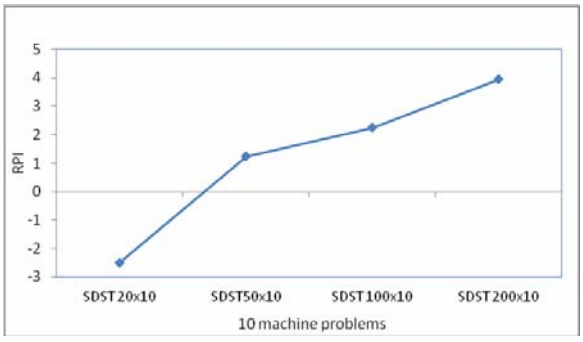
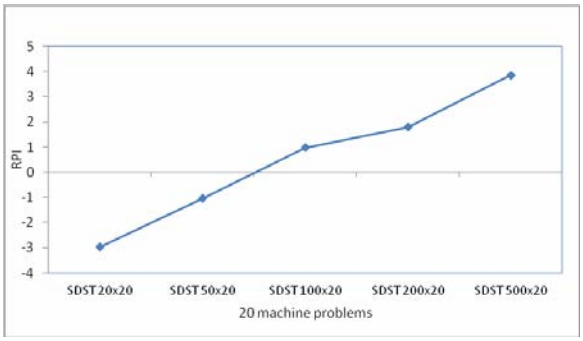


Figure 11 RPI plot for 20 machines with increasing number of job (see online version for colours)



7 Conclusions

This paper presents the analysis of two different constructive algorithms for SDST flow shop scheduling. The first constructive heuristic algorithm, NEHRB heuristic considers only processing time for constructing the sequence. The second constructive heuristic, FJSRA uses setup time data along with processing time data for constructing the sequence. The objective of the study is to find the relative performance of these two algorithms with increasing problem size. The problem size is characterised by the varibles namely, number of jobs and number of machines analysed. Eexperimentation is carried out on benchmark problems involving 960 problem instances. Relative performance improvement analysis and statistical analysis are carried out. The analysis reveals that FJSRA emerges as the better algorithm for larger-size problems and for smaller-size problems with higher level of setup time.

It is a well-known fact that the solution quality to flow shop scheduling problem depends on number of jobs and number of machines. However, the present study analyses the performance variation of two constructive algorithms. Moreover, the present study also analyses the problem size parameters (number of jobs, n and number of machines, m). The study shows that the performance of the setup time-based algorithm FJSRA increases with increase in number of jobs. On the other hand, the performance of FJSRA decreases with increase in number of machines. A coefficient of variation analysis carried out in the present study investigates the reason for the decreased performance of FJSRA, which is a major contribution of this paper. The analysis reveals that as the number machines increases, the coefficient of variation of the summed setup time matrix decreases which causes the reduced performance of FJSRA.

SDST flow shop situation can be observed in various manufacturing and service industries. The size of the scheduling problem involved in such industries and situations often vary. The analysis carried out in the present research brings out the fact that the performance of the two constructive algorithms changes with change in problem size. As the number of machines in a problem situation increases, the performance of the setup time-based algorithm decreases. The processing time-based algorithm, NEHRB is suited for scheduling jobs in such situations. When the number jobs are relatively high, the setup time-based algorithm FJSRA is more appropriate. The performance variation of the two algorithms with respect to the number of jobs and the number machines identified in

the present study is the major contribution of the paper. Moreover, the coefficient of variation analysis used for the purpose of investigating the effect of increasing number of machines is a novel approach. The present study provides guidelines for choosing the correct algorithm for different situations so as to develop a better schedule that minimises the total completion time of jobs or makespan. In an increasingly competitive world of manufacturing, improving the schedules by as little as 1% can have a significant financial impact.

Considering the importance of due-date fulfilment in determining service level, the industry demands research with due-date consideration. Hence, attention needs to be directed towards customer-driven performance measures such as minimising the total weighted earliness and tardiness, weighted number of tardy jobs, etc. The high computational requirements of the problem considered in the present research make metaheuristics as a promising solution methodology. The possibility of non-permutation schedule in flow shop with SDST can also be explored.

Acknowledgements

The authors express their sincere thanks to the editor and the reviewers for their constructive comments and suggestions which have immensely helped to bring this paper to the present form.

References

Allahverdi, A., Gupta, J.N.D. and Aldowaisan, T. (1999) 'A review of scheduling research involving setup considerations', *Omega, International Journal of Management Science*, Vol. 27, No. 2, pp.219–239.

Allahverdi, A., Ng, C.T., Cheng, T.C.E. and Kovalyov, M.Y. (2008) 'A survey of scheduling problems with setup times or costs', *European Journal of Operational Research*, Vol. 187, No. 3, pp.985–1032.

Eren, T. (2010) 'A bicriteria *m*-machine flowshop scheduling with sequence-dependentsetuptimes', *Applied Mathematical Modelling*, Vol. 34, No. 2, pp.284–293.

Gajpal, Y., Rajendran, C. and Ziegler, H. (2006) 'An ant colony algorithm for scheduling in flow shops with sequence-dependent setup times of jobs', *International Journal of Advanced Manufacturing Technology*, Vol. 30, Nos. 5–6, pp.416–424.

Gupta, J.N.D. (1986) 'Flow shop schedules with sequence dependent setup times', *Journal of Operations Research Society Japan*, Vol. 29, No. 3, pp.206–219.

Logendran, R., deSzoeke, P. and Barnard, F. (2006) 'Sequence-dependent group scheduling problems in flexible flow shops', *International Journal of Production Economics*, Vol. 102, No. 1, pp.66–86.

Nawaz, M., Enscore, E.E. and Ham, I. (1983) 'A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem', *Omega*, Vol. 11, No. 1, pp.91–97.

Pargar, F. and Zandieh, M. (2012) 'Bi-criteria SDST hybrid flow shop scheduling with learning effect of setup times: water flow-like algorithm approach', *International Journal of Production Research*, Vol. 50, No. 10, pp.2609–2623, doi:10.1080/00207543.2010.546380.

Parthasarathy, S. and Rajendran, C. (1997) 'A simulated annealing heuristic for scheduling to minimize weighted tardiness in a flowshop with sequence dependent setup times of jobs – a case study', *Production Planning and Control*, Vol. 8, No. 5, pp.475–483.

Rajendran, C. and Ziegler, H. (2003) 'Scheduling to minimize the sum of weighted flow time and weighted tardiness of jobs in a flow shop with sequence-dependent setup times', *European Journal of Operational Research*, Vol. 149, No. 3, pp.513–522.

Rios-Mercado, R.Z. and Bard, J.F. (1998) 'Heuristics for the flow line problem with setup costs', *European Journal of Operational Research*, Vol. 110, No. 1, pp.76–98.

Ruiz, R. and Maroto, C. (2006) 'A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility', *European Journal of Operational Research*, Vol. 169, No. 3, pp.781–800.

Ruiz, R. and Stutzle, T. (2008) 'An iterated greedy heuristic for the sequence dependent setup times flow shop with makespan and weighted tardiness objectives', *European Journal of Operational Research*, Vol. 187, No. 3, pp.1143–1159.

Ruiz, R., Maroto, C. and Alcatraz, J. (2005) 'Solving the flow shop scheduling problem with sequence dependent setup times using advanced meta heuristic', *European Journal of Operational Research*, Vol. 165, No. 1, pp.34–54.

Tseng, F.T., Gupta, J.N.D. and Stafford, E.F. (2005) 'A penalty-based heuristic algorithm for the permutation flow shop scheduling problem with sequence-dependent set-up times', *Journal of the Operational Research Society*, Vol. 57, No. 5, pp.541–551.

Vanchipura, R. and Sridharan, R. (2012) 'Development and analysis of constructive heuristic algorithms for flow shop scheduling problems with sequence-dependent setup times', *International Journal of Advanced Manufacturing Technology*, Vol. 67, Nos. 5–8, pp.1337–1353, doi: 10.1007/s00170-012-4571-8.

Varmazyar, M. and Salmasi, N. (2012) 'Sequence-dependent flow shop scheduling problem minimising the number of tardy jobs', *International Journal of Production Research*, Vol. 50, No. 20, pp.5843–5858, doi:10.1080/00207543.2011.632385.

Wang, H., Chou, F. and Wu, F. (2011) 'A simulated annealing for hybrid flow shop scheduling with multiprocessor tasks to minimize makespan', *International Journal of Advanced Manufacturing Technology*, Vol. 53, Nos. 5–8, pp.761–776.

Zandieh, M. and Gholami, M. (2009) 'An immune algorithm for scheduling a hybrid flow shop with sequence-dependent setup times and machines with random breakdowns', *International Journal of Production Research*, Vol. 47, No. 24, pp.6999–7027.

Appendix 1

Illustration of NEHRB heuristic

An example problem in which six jobs need processing on four different machines is considered. Table A1 shows the processing time matrix. The setup time matrices for machines 1, 2, 3 and 4 are provided in Tables A2, A3, A4 and A5 respectively.

Table A1 Processing time matrix

		Jobs					
		J_1	J_2	J_3	J_4	J_5	J_6
Machines	M_1	5	7	0	1	1	5
	M_2	2	4	9	3	7	8
	M_3	5	4	9	3	1	7
	M_4	6	10	8	7	7	9

Table A2 Setup time matrix for machine 1

	J_0	J_1	J_2	J_3	J_4	J_5	J_6
J_0	-	9	3	7	2	0	7
J_1	0	-	5	5	9	6	6
J_2	0	9	-	8	6	2	2
J_3	0	9	0	-	5	2	10
J_i	0	7	5	5	-	1	7
J_i	0	0	1	5	1	-	8
J_6	0	8	7	1	7	5	-

Table A3 Setup time matrix for machine 2

	J_0	J_1	J_2	J_3	J_4	J_5	J_6
J_0	-	10	6	8	5	4	8
J_1	0	-	1	1	2	4	8
J_2	0	8	-	1	4	5	4
J_3	0	7	6	-	3	4	0
J_i	0	10	2	1	-	4	2
J_i	0	5	3	10	9	-	1
J_6	0	7	3	4	5	9	-

Table A4 Setup time matrix for machine 3

	J_0	J_1	J_2	J_3	J_4	J_5	J_6
J_0	0	4	10	3	7	7	5
J_1	0	-	7	7	2	1	10
J_2	0	2	-	0	6	9	7
J_3	0	2	4	-	5	10	2
J_i	0	9	6	4	-	2	4
J_i	0	5	1	6	2	-	4
J_6	0	6	3	3	6	3	-

Table A5 Setup time matrix for machine 4

	J_0	J_1	J_2	J_3	J_4	J_5	J_6
J_0	0	8	10	7	3	6	1
J_1	0	-	9	9	8	3	6
J_2	0	0	-	4	3	2	2
J_3	0	4	1	-	6	5	7
J_i	0	7	6	0	-	1	3
J_i	0	5	7	4	8	-	7
J_6	0	10	5	3	1	6	-

The NEHRB heuristic uses the processing time data to construct the sequence. The algorithms starts with finding summed processing time for each jobs. The sums of the processing time are calculated shown in Table A6.

Table A6 Sum of processing time for job

Job	*1*	*2*	*3*	*4*	*5*	*6*
Sum of processing time	18	25	26	14	16	29

The jobs are sorted based on the summed processing time and presented in Table A7.

Table A7 Sorted sum of processing time for jobs

Job	*6*	*3*	*2*	*1*	*5*	*4*
Sum of processing time	29	26	25	18	16	14

Initialisation

Set of unscheduled jobs, u = {1, 2, 3, 4, 5, 6}; Partial sequence, σ = {0}; Makespan = 0.

1st iteration

The job with maximum processing time, i.e., job 6 with processing time 29, is removed from u and added to the partial sequence, σ.

Set of unscheduled jobs, u = {1, 2, 3, 4, 5}; Partial sequence σ = {0 – 6}; Makespan = 29.

2nd iteration

The next job with the highest processing time in the set of unscheduled jobs is removed, i.e., job 3, and added to the partial sequence, σ. The different possibilities of inserting the job in the partial sequence are evaluated in the Table A8. Here, both the sequences are found to have the same makespan value. Hence, arbitrarily {0 – 6 – 3} is chosen as the partial sequence.

Table A8 possible sequences and the corresponding makespan

Sequence	*Makespan*
{0 – 3 – 6}	50
{0 – 6 – 3}	50

Set of unscheduled jobs, u = {1, 2, 4, 5}; partial sequence, σ = {0 – 6 – 3}; makespan = 50.

3rd iteration

The next job with the highest processing time from the set of unscheduled jobs is removed and added to the partial sequence σ. The different possibilities of inserting the job in the partial sequence are evaluated in the Table A9. The best sequence is selected as the partial sequence.

Table A9 possible sequences and corresponding makespan

Sequence	*Makespan*
{0 – 2 – 6 – 3}	56
{0 – 6 – 2 – 3}	63
{0 – 6 – 3 – 2}	61

4th iteration

Set of unscheduled jobs, u = {1, 4, 5}; partial sequence σ = {0 – 2 – 6 – 3}; makespan = 56. The new partial sequence σ is {0 – 2 – 6 – 3} with makespan value as 56. The next job with the highest processing time from the set of unscheduled jobs is removed and added to the partial sequence, σ. The different possibilities of inserting the job in the partial sequence are evaluated in Table A10. The best sequence is selected as the partial sequence.

Table A10 possible sequences and corresponding makespan

Sequence	*Makespan*
{0 – 1 – 2 – 6 – 3}	72
{0 – 2 – 1 – 6 – 3}	73
{0 – 2 – 6 – 1 – 3}	75
{0 – 2 – 6 – 3 – 1}	66

Set of unscheduled jobs, u = {4, 5}; partial sequence, σ = {0 – 2 – 6 – 3 – 1}; makespan = 66.

5th iteration

The sequence with the lowest makespan, {0 – 2 – 6 – 3 – 1} is taken as the new partial sequence, σ, with makespan value as 66. The next job with the highest processing time from the set of unscheduled job is removed added to the partial sequence, σ. The different possibilities of inserting the job in the partial sequence are evaluated in Table A11. The best sequence is selected as the partial sequence.

Table A11 possible sequences and corresponding makespan

Sequence	*Makespan*
{0 – 5 – 2 – 6 – 3 – 1}	70
{0 – 2 – 5 – 6 – 3 – 1}	75
{0 – 2 – 6 – 5 – 3 – 1}	80
{0 – 2 – 6 – 3 – 5 – 1}	79
{0 – 2 – 6 – 3 – 1 – 5}	76

Set of unscheduled jobs, u = {5}; partial sequence, σ = {0 – 2 – 6 – 3 – 1 – 5}; makespan = 70.

6th iteration

The sequence with the lowest makespan, {0 – 2 – 6 – 3 – 1 – 5} is taken as the new partial sequence, σ, with makespan value as 70. Now, the remaining job is added to the partial sequence, σ. The different possibilities of inserting the job in the partial sequence are evaluated in the Table A12. The best sequence is selected as the partial sequence.

Table A12 Possible sequences and corresponding makespan

Sequence	*Makespan*
{0 – 4 – 5 – 2 – 6 – 3 – 1}	78
{0 – 5 – 4 – 2 – 6 – 3 – 1}	82
{0 – 5 – 2 – 4 – 6 – 3 – 1}	79
{0 – 5 – 2 – 6 – 4 – 3 – 1}	77
{0 – 5 – 2 – 6 – 3 – 4 – 1}	86
{0 – 5 – 2 – 6 – 3 – 1 – 4}	85

Set of unscheduled jobs, u = { }; partial sequence, σ = {0 – 5 – 2 – 6 – 4 – 3 – 1}; makespan = 77.

There are no more jobs to be inserted into the partial sequence. Hence, {0 – 5 – 2 – 6 – 4 – 3 – 1} forms the final sequence with makespan value as 77.

Appendix 2

Illustration of FJSRA heuristic

For the illustration of FJSRA heuristic, the example problem provided in Appendix 1 is used. The algorithm starts with summing the setup time matrices of each machine. In the present example, there are four machines and four setup time matrices as shown in Tables A2, A3, A4 and A5. The setup time matrices are added to get summed setup time matrix as shown in Table A1.

Table A1 Summed setup time matrix

	J_0	J_1	J_2	J_3	J_4	J_5	J_6
J_0	0	31	29	25	17	17	21
J_1	0	-	22	22	21	14	30
J_2	0	19	-	13	19	18	15
J_3	0	22	11	-	19	21	19
J_i	0	33	19	10	-	8	16
J_i	0	15	12	25	20	-	20
J_6	0	31	18	11	19	23	-

The summed setup time matrix shows the total setup time between each combination of jobs. FJSRA heuristic uses this information to construct the sequence. The elements of the summed setup time matrix are ranked according to the total setup time to get sorted summed setup time matrix.

Table A2 Sorted summed setup time matrix

Sl. no.	*Job 1*	*Job 2*	*Total setup time*	*Sl. no.*	*Job 1*	*Job 2*	*Total setup time*
1	4	5	8	19	6	4	19
2	4	3	10	20	3	6	19
3	3	2	11	21	5	4	20
4	6	3	11	22	5	6	20
5	5	2	12	23	1	4	21
6	2	3	13	24	3	5	21
7	1	5	14	25	0	6	21
8	5	1	15	26	3	1	22
9	2	6	15	27	1	2	22
10	4	6	16	28	1	3	22
11	0	4	17	29	6	5	23
12	0	5	17	30	0	3	25
13	6	2	18	31	5	3	25
14	2	5	18	32	0	2	29
15	2	1	19	33	1	6	30
16	4	2	19	34	0	1	31
17	2	4	19	35	6	1	31
18	3	4	19	36	4	1	33

From Table A2, it can be found that the smallest total setup time is between jobs 4 and 5. The key concept involved in FJSRA algorithm is fictitious job concept. A pair of jobs with minimum setup time between them is treated as a fictitious job and the processing time of the fictitious job is the sum of the processing time of the pair of jobs involved in the fictitious job. The first fictitious job identified is [4 – 5] which has the lowest setup time between them. Similarly, other fictitious jobs are identified and their respective processing times are obtained as shown in Table A3.

Table A3 Fictitious jobs and processing times

Fictitious jobs	*Processing time*
[4 – 5]	30
[3 – 2]	51
[1 – 6]	47

The partial sequence is initialised as $\sigma = \{0\}$. Then, the fictitious job with maximum processing time [3 – 2] is inserted into the partial sequence, σ. Now, the partial sequence is {0 – 3 – 2} with makespan value as 26. The next fictitious job to be inserted is [1 – 6]. It can be inserted in two ways as follows:

- 1st insertion: {0 – 3 – 2 – *1* – *6*}; makespan = 69
- 2nd insertion: {0 – *1* – *6* – 3 – 2}; makespan = 74.

Now, the best sequence is {0 – 3 – 2 – 1 – 6} with makespan value as 69 is fixed as new partial sequence σ. Next, fictitious job, [4 – 5] is inserted into the sequence. The three different possible insertions are given below.

- 1st insertion: {0 – *4* – *5* – 3 – 2 – 1 – 6}; makespan = 90
- 2nd insertion; {0 – 3 – 2 – *4* – *5* – 1 – 6}; makespan = 89
- 3rd insertion; {0 – 3 –2 – 1 – 6 – *4* – *5*}; makespan = 85.

So, the best sequence is {0 –3 –2 –1 –6 – 4 – 5} with makespan value as 85. Since there are no more jobs remaining to be inserted, this forms the final sequence.