Development and analysis of hybrid genetic algorithms for flow shop scheduling with sequence dependent setup time

Rajesh Vanchipura

Department of Mechanical Engineering,
Government Engineering College,
Thrissur, Kerala − 680 009, India
E-mail: rajeshvanchipura@gmail.com

R. Sridharan*

Department of Mechanical Engineering,
National Institute of Technology Calicut,
Kerala − 673 601, India
E-mail: sreedhar@nitc.ac.in
*Corresponding author

Abstract: This paper deals with the development and analysis of hybrid genetic algorithms for flow shop scheduling problems with sequence dependent setup time. A constructive heuristic called setup ranking algorithm is used for generating the initial population for genetic algorithm. Different variations of genetic algorithm are developed by using combinations of types of initial populations and types of crossover operators. For the purpose of experimentation, 27 group problems are generated with ten instances in each group for flow shop scheduling problems with sequence dependent setup time. An existing constructive algorithm is used for comparing the performance of the algorithms. A full factorial experiment is carried out on the problem instances developed. The best settings of genetic algorithm parameters are identified for each of the groups of problems. The analysis reveals the superior performance of hybrid genetic algorithms for all the problem groups.

Keywords: flow shop; sequence dependent setup time; SDST; hybrid genetic algorithm; parameter tuning.

Reference to this paper should be made as follows: Vanchipura, R. and Sridharan, R. (2014) 'Development and analysis of hybrid genetic algorithms for flow shop scheduling with sequence dependent setup time', *Int. J. Services and Operations Management*, Vol. 17, No. 2, pp.168–193.

Biographical notes: Rajesh Vanchipura is an Assistant Professor at Government Engineering College, Thrissur, Kerala. He received MTech degree in Industrial Engineering and Management from National Institute of Technology Calicut. He also holds an MBA degree in Operations and Marketing Management from 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.

Sridharan R. 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 the refereed international journals and proceedings of international and national conferences. Currently, he is a Professor and the Head of the Department of Mechanical Engineering at the National Institute of Technology Calicut, Kerala, India.

1 Introduction

One of the realistic variations of flow shop scheduling problems is the one with sequence dependent setup time (SDST). In practical scheduling situations, setup times do exist between two consecutive jobs on a machine. It can be observed in various manufacturing, service and information processing operations. Setup times are conveniently added to processing times rather than considering it separately. The importance and benefits of incorporating setup times in scheduling research has been investigated by many researchers (Allahverdi and Soroush, 2008). The setup times considered in this study are dependent on the sequence in which jobs are processed on a machine. The SDSTs are found in chemical, textile, pharmaceutical, food processing, metal processing, paper manufacturing, semiconductor manufacturing, web applications, and many other industries (Eren, 2011; Allahverdi et al., 2008). Some researchers have used SDST consideration in their research works (Sabouni and Logendran, 2013; Tanaka and Araki, 2013; Pargar and Zandieh, 2012; Gómez-Gasquet et al., 2012; Bozorgirad and Logendran, 2012; Bouabda et al., 2011; Magableh and Mason, 2010; Vinod and Sridharan, 2008a, 2008b). The present study also considers the setup time as separable, i.e., the job is not physically required to perform the setup. The problem addressed in the present study involves scheduling a set of n jobs which are available for processing on m machines where the setup time is SDST i.e., the setup time of job on a machine is dependent on the previous 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. The problem is commonly known in the literature as flow shop scheduling problem with SDST or SDST flow shop. This problem is denoted as $F|s_{ijk}, prmu|C_{\max}$, where the first field describes the machine environment (F stands for an m-machine flow shop), the second field provides details of processing characteristics and constraints (s_{ijk} stands for SDSTs and *prmu* means that the order or permutation in which the jobs go through the first machine is maintained throughout the system) and the third field contains the objective to be minimised. Here, s_{ijk} is the known, deterministic and non-negative time span required for setting up machine i when job k is preceded by job j.

Even though, the scheduling area has received much attention from researchers over the years (Amiri et al., 2013; Srinivasan and Srirangacharyulu, 2012; Joseph and Sridharan, 2011; Joo and Min 2011; Boudhar and Tchikou, 2010; Kanagasabapathi et al., 2010; Soroush and Alqallaf, 2009; Kanyalkar and Adil, 2009; Lawrynowicz, 2007), research works carried out in SDST flow shop scheduling problem is very much limited.

The computational complexity is one of the reasons for lesser number of research works in the area. The SDST flow shop scheduling problem is considered as NP complete (Gupta, 1986). The complexity of the problem demands the development and use of computationally less intensive heuristics for the problem. Constructive algorithms with lesser complexity for such problems are difficult to construct and therefore are rare. Rios-Mercado and Bard (1998) propose a constructive heuristic algorithm called NEHRB, which is an extension of the well-known NEH heuristic (Nawaz et al., 1983). On the contrary, improvement algorithms such as metaheuristics and random search algorithms are considered as the popular choice and are found more frequently in the literature. Metaheuristics are general procedure heuristics which can be applied to different problems. The metaheuristics come under the category of improvement algorithms. Metaheuristics have been applied to the general scheduling problems by several researchers (Sobhani and Wong, 2013; Kumar and Sivakumar, 2013; Nguyen and Kachitvichyanukul, 2012; AitZai et al., 2012; Kasemset and Kachitvichyanukul, 2012; Solimanpur and Rastgordani, 2012; Defersha and Chen, 2011; Ramanan et al., 2011; Amuthakkannan et al., 2010; Laha and Mandal, 2007; Arumugam et al., 2007). However, there have been a few applications of metaheuristics in scheduling SDST flow shop. Parthasarathy and Rajendran (1997) develop a simulated annealing algorithm to minimise mean weighted tardiness in a flow shop with SDSTs. Ruiz et al. (2005) develop two heuristics based on genetic algorithm (GA) for flow shop with SDST using makespan criteria. Gajpal et al. (2006) consider the makespan objective and present an ant colony algorithm for flow shop scheduling with sequence dependent setups.

The present study develops and analyses hybrid GAs for scheduling SDST flow shops. A constructive heuristic called setup ranking algorithm (SRA) is used for developing the proposed hybrid GAs (Vanchipura and Sridhran, 2012). To the best knowledge of the authors, this is first application of hybrid metaheuristic for solving SDST flow shop scheduling problem operating in a SDST environment. Generally, a metaheuristic has a drawback that it may end up in local optimum. This drawback can be reduced to some extent by tuning the parameters of the metaheuristic. In the literature, many researchers have used parameter tuning for different metaheuristics (Komarudin and Wong, 2012; Vinay and Sridharan, 2012; Lessmann et al., 2011). Full factorial experimentation is one method for tuning the parameters of the metaheuristic. Here, metaheuristics need to be tested for all possible combinations of parameters at different levels. GA make use of two operators namely, crossover operator and mutation operator. Three different existing crossover operators namely, partially matched crossover (PMX), similar block 2 point order crossover (SB2OX) and similar job 2 point order crossover (SJ2OX) are used as the crossover operators for GA. The performance of metaheuristics is also dependent upon the initial solutions given to the metaheuristic. The present study considers three different initial populations, namely, Random population, SRA population and SRA population + Random population. The possible combinations of crossover operators and population types result in nine different variations of GA. Out of the nine variations; six of them use either SRA population or SRA population + Random population. They are hybrid GAs since they use a different algorithm to obtain input population for the GA. The crossover probability and mutation probability are the other parameters that need to be optimised for GA. In the present study, four different levels of probabilities for both crossover and mutation are considered. Thus, three levels of initial population, three levels of crossover types, four levels of crossover probability and four levels of mutation probability lead to 144 combinations. All these combinations are tested

for 270 problem instances. Statistical analysis based on four-factor ANOVA has been carried out on the results obtained. Using the ANOVA results, the problems are grouped into three cases based on the interactions among the factors. The best settings of GA parameters are identified for each of the groups of problems.

The rest of the paper is organised as follows. Section 2 provides formulation of the SDST flow shop scheduling problem. Section 3 deals with the solution methodology. Section 4 describes the details of experimentation. Section 5 presents the results and the analyses. Section 6 provides the conclusion.

2 Formulation of the flow shop scheduling problem with SDSTs

The flow shop scheduling problem considered in the present study involves a set of n jobs to be processed on a set of m machines all in the same order. The objective of this problem is to minimise the time at which the last job in the sequence finishes processing on its last machine, i.e., minimise the makespan. The following assumptions are made:

- Each job is available at time zero.
- Each job can be processed at most on one machine at the same time.
- Each machine can process only one job at a time.
- No pre-emption is allowed, i.e., the processing of a job on a machine cannot be interrupted.
- The setup times of the jobs on machines are separable; the job is not necessary to do the setup.
- Setup time is dependent on the sequence in which the jobs are processed.
- The SDST is asymmetric; i.e., $s_{ijk} \neq s_{ikj}$.
- All the processing times and setup times are known in advance.
- In-process inventory is allowed.
- The machines are continuously available.

Notations

n	Total number of jobs to be scheduled
i	Index of machine; $(i - 1)$ indicates the previous machine in the sequence
m	Total number of machines in the flow shop
j	Index of job, $(j - 1)$ indicates the previous job processed in the machine
p_{ij}	Processing time of job j on machine i
s_{ijk}	Setup time on machine i, when job k is preceded by the job j
σ	Ordered set of jobs already scheduled, out of n jobs; partial sequence

$q(\sigma, i)$ Completion time of partial sequence σ on machine i (i.e., the release time of machine i after processing all jobs in the partial sequence σ)

$q(\sigma j, i)$ 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 (Gajpal et al., 2006):

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

where $q(\Phi, i) = 0$ 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. The completion time for the current partial sequence σ on machine i is $q(\sigma, i)$. Now, consider the case when new job j is added to the partial sequence σ. The processing of job j can be started only after the occurrence of the two events namely,

1 completion of processing of previous job on the same machine i and the setup for job j on the machine $q(\sigma, i) + s_{ijk}$

2 completion of the immediately preceding operation of job j on the previous machine $q(\sigma j, i - 1)$.

Hence, equation (1) provides the completion time of job j on machine i. This equation can be applied to find the completion time of job j on the last machine m which is the total completion time of job j and it is given by the following equation.

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

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

$$M = \max\{c_j, j = 1, 2, ..., n\} \tag{3}$$

Example

Consider the following instance of SDST flow shop with four jobs and three machines. Table 1 shows the processing time matrix for the jobs. The SDSTs for the three machines are given in Tables 2, 3 and 4. Using the above formulation, makespan values are obtained for the example problem instance.

Table 1 Processing time matrix

		Job				
		0	*1*	*2*	*3*	*4*
Machine	*1*	0	12	15	10	12
	2	0	2	11	12	11
	3	0	13	10	13	9

Table 2 Setup time matrix for Machine 2

		Job				
		0	*1*	*2*	*3*	*4*
Machine	*0*	-	4	3	1	0
	1	0	-	4	1	3
	2	0	0	-	2	1
	3	0	2	2	-	1
	4	0	0	1	2	-

Table 3 Setup time matrix for Machine 1

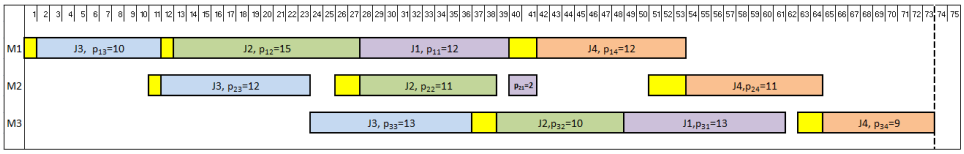
		Job				
		0	*1*	*2*	*3*	*4*
Machine	*0*	-	2	4	1	3
	1	0	-	2	7	2
	2	0	0	-	1	1
	3	0	4	1	-	1
	4	0	1	2	3	-

Table 4 Setup time matrix for Machine 3

		Job				
		0	*1*	*2*	*3*	*4*
Machine	*0*	-	0	1	3	0
	1	0	-	0	1	2
	2	0	0	-	3	2
	3	0	4	2	-	1
	4	0	1	2	2	-

For a sequence S = {3 2 1 4}, a feasible schedule can be prepared as shown in Figure 1 and the corresponding makespan C_{max} = 73.

Figure 1 Gant chart for the schedule for the example (see online version for colours)



The formulation gives the procedure for finding makespan for a given sequence. In flow shop, sequences are possible. Since the present study considers only permutation schedules, there are n sequences. The combinatorial problem becomes more complex by the addition of SDSTs. Clearly, the above problem, even with a moderate size is difficult to solve optimally. Thus, the present study focuses on the use of metaheuristic procedures for solving the problem.

3 Solution methodology

The flow shop with SDST is considered as NP complete (Gupta, 1986). It is well-known that the general flow shop scheduling problem is NP hard and the presence of SDST makes it further hard. The very nature of the problem makes it difficult to get an optimal solution for even small size problems. Metaheuristic is the suitable solution methodology in such situations. But, the literature shows relatively lesser number of applications of such heuristics for SDST flow shop. The present study develops and analyses hybrid GAs for scheduling SDST flow shop. A constructive heuristic called SRA is used in the development of the proposed hybrid GAs. GA is a non-traditional optimisation technique that mimics natural evolution and genetics. It revolves around the natural reproduction process and the survival of the fittest strategies. The GA has been successfully applied for various optimisation problems (Charles and Udhayakumar, 2012; Pal and Gupta, 2012). In the present study, three different types of initial population and three different types of crossover operators are used. The details of the type of crossover operators and the types of population are provided in the following section.

3.1 Three types of initial population

GA is an improvement algorithm and it needs an initial population of solution sequences to start with. The initial populations are very important to GA due to the fact that the performance of GA depends on the quality of this initial population. The size of the population is normally taken as the length of the solution sequence. Since GA is an improvement algorithm, a quick convergence is obtained when it is provided with a good quality initial population. It is quite obvious that better chromosomes or solution sequences are closer to the optimal solution and result in quick convergence of GA. But, quick convergence has a possible risk of the solution getting trapped in local optimum. In the present study, three different types of initial populations are considered. These are

1 SRA population

2 Random population

3 SRA population + Random population.

In random population, chromosomes or solution sequences are constructed in random order. SRA population employs a new heuristic called SRA for the purpose of constructing a population. The solution sequences developed by SRA are better in terms of closeness to optimal solution. On the other hand, the random population is truly random and cannot demonstrate any quality or closeness to optimal solution. The third initial population is a mix of SRA population and random population; therefore, it lies somewhere in between the SRA population and random population in terms of quality. The logic of SRA is explained in the following section.

3.1.1 Setup ranking algorithm

SRA is a sequence construction procedure primarily based on the setup time between jobs (Vanchipura and Sridharan, 2012). Since the processing time remains constant, decrease in setup time could lead to a reduced makespan. Usually, constructive algorithms are designed to give one sequence as output. But, SRA is a constructive

algorithm specially designed to create multiple number of sequences. In constructing the sequence of jobs, SRA considers the SDST of consecutive jobs and not the processing time of jobs. The fundamental principle behind this algorithm is that as the setup time between jobs is reduced, the makespan also will get reduced. The algorithm starts with the summed setup time matrix, SST. The matrix SST is obtained by the matrix addition operation on the setup time matrices of all the machines. This 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 jobs and can be used for ranking combinations of jobs from the lowest setup time sum to the maximum setup time sum to get the sorted summed setup time matrix, SSST. The SSST matrix shows the best combinations of jobs in terms of setup time. Each job combinations of the matrix SSST can be used as seed for new sequence which will lead to multiple sequences with reduced setup time. If n jobs are considered, there will be n^2 seed sequences. For example, the first job combination in SSST is taken as the seed of the sequence. The complete sequence is built by adding the lowest possible combination on either side of the seed or partial sequence. The sorting of summed setup time is the major computation of the algorithm with computational efficiency of the order of n^2. Summing the setup time matrix operation has computational efficiency of the order of n. Hence, the computational complexity of this algorithm is $O(n^2)$. The details of the algorithm are presented in the following section.

3.1.2 Procedure of SRA

Input: Processing time and setup time matrices.

Output: Feasible schedule and makespan.

Step 1: Set $\sigma = 0$

Step 2: $sst_{jk} = \sum_{i=1}^{m} s_{ijk}$, for $\forall j, k$

where sst_{jk} represents jth row kth column element of *SST*.

Step 3: Sort the elements of *SS*T in ascending order to get *SSST*.

Step 4: Select the Min (*SSST*)

Step 5: Select the jobs associated with Min (*SSST*) as seed sequence.

While $n\sigma < n$, do

Step 5a: Find eligible jobs in the partial sequence.

Step 5b: List all possible combinations for eligible jobs.

Step 5c: Find the job with minimum rank.

Step 5d: Append to partial sequence.

Step 6: Output σ

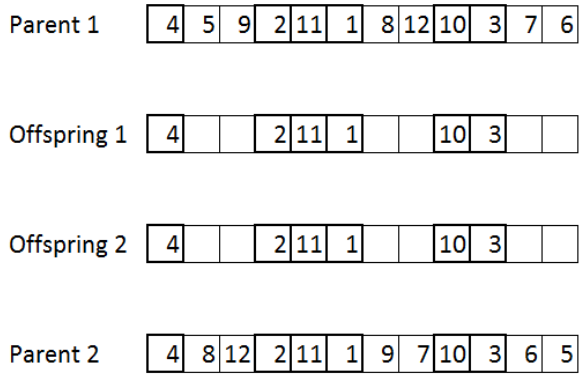
Step 7: Stop

3.2 Types of crossover operators

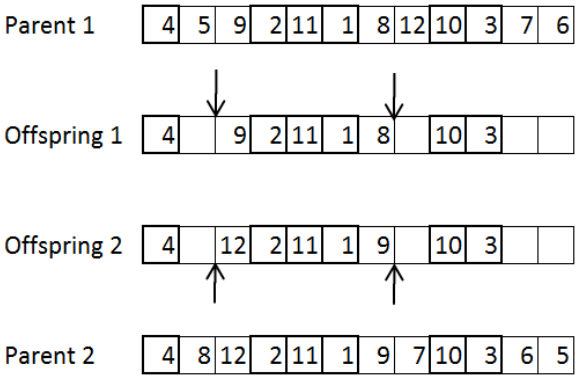
Crossover operation is the mechanism employed in GA for producing new chromosomes or solution sequences from the mating pool (the existing set of solutions). The new

strings are created by exchanging information among chromosomes in the mating pool. In most crossover operators, two strings are picked from the mating pool at random and some portions of the strings are exchanged between the strings. Simply exchanging portions between strings does not work with flow shop problems because this will lead to duplication of jobs in the strings, thus making the strings infeasible. Hence, special types of crossover operators are generally designed for flow shop scheduling problems. PMX is the commonly used crossover operator in flow shop problems. Other crossover operators that are specifically designed for these problems are SJ2OX, and SB2OX (Ruiz and Maroto, 2006). The SJ2OX operator proceeds as follows: At first, both parents are examined on a position-by-position basis and the building blocks of jobs are directly copied to the offspring [Figure 2(a)]. Then, two random cut points are drawn and section between these is directly copied to the offspring and finally the missing elements are copied in the relative order as it appears in the other parent [Figure 2(b) and Figure 2(c)].

Figure 2 SJ2OX procedure (a) the common jobs in both parents are copied over to the offspring (b) jobs between two randomly chosen cut point are inherited from the parent (c) the missing elements in the offspring are copied in the relative order of the other parent

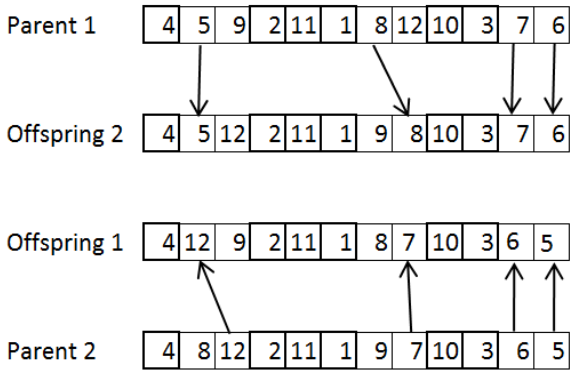


(a)



(b)

Figure 2 SJ2OX procedure (a) the common jobs in both parents are copied over to the offspring (b) jobs between two randomly chosen cut point are inherited from the parent (c) the missing elements in the offspring are copied in the relative order of the other parent (continued)



(c)

In SB2OX, the first step of SJ2OX is modified by considering blocks of at least two consecutive identical jobs. In SJ2OX, individual jobs in identical positions are copied to offspring but, in SB2OX at least two jobs are required in the identical positions so that it can be copied.

3.3 Mutation and crossover probabilities

The probabilities of mutation and crossover are the other two important factors that will determine the performance of genetic algorithm. Convergence is generally considered as a performance criteria for GA. The improvement in the solution value of the problem is negligible for each of the iteration after convergence. But, a quick convergence does not necessarily mean that the solution is close to optimum. It is also possible that the solution is converging to a local optimum. Both crossover and mutation try to reduce the chance of the algorithm getting trapped in a local optimum. When the crossover and mutation probability is high, there will be lesser chances for a quick convergence and getting trapped in a local optimum. But, increasing the probability of crossover and mutation will extend the possible termination of the GA at specified convergence criteria. Hence, it is very much essential to find the best settings for the levels of probability of crossover and mutation. In the present study, both crossover probability and mutation probability are tested at four different levels each.

4 Experimentation

The experimentation is carried out in two stages:

1 identification of benchmark problems for SDST flow shop scheduling problem

2 testing of hybrid GA.

The benchmark problems for SDST flow shop presented in research paper by Vanchipura and Sridhran (2012) are considered for experiments. These problems are developed by augmenting setup time matrices for each machine in the Taillard benchmark problems (Taillard, 1993). Three different levels of setup times namely, 50%, 100% and 150% are developed using uniform distributions, U(1, 50), U(1, 100), U(1, 150) respectively. Nine different sizes of Taillard flow shop benchmark problems, 20 × 5, 20 × 10, 20 × 20, 50 × 5, 50 × 10, 50 × 20, 100 × 5, 100 × 10, and 100 × 20 are considered for the development of benchmark SDST flow shop scheduling problems. These problem groups are formed by taking different combinations of machines (5, 10 and 20) and jobs (20, 50 and 100). Ten instances are developed for each of the 27 groups of problems (three levels of setup time × three levels of number of machines × three levels of number of jobs) thus constituting 270 problem instances.

The next stage is the development of the metaheuristics based on GA. The main objective at this stage is the development of hybrid GAs with the best settings of the parameters. Three different types of initial populations and three different types of crossover operators are used in the study. The possible combinations of population types and crossover operators result in nine different variations of GA. Out of the nine variations, six of them use either SRA population or SRA population + Random population. They are hybrid GAs since they use a different algorithm to obtain initial population for the GA. The remaining three variations of GA are the GA based metaheuristic without hybridisation. The SRA heuristic combined with GA forms the structure of the hybrid GAs. SRA is a constructive heuristic used for constructing multiple solution sequences as explained in the preceding section. The factors such as the initial population type set at three levels, crossover operator type set at three levels, crossover probability set at four levels and the mutation probability set at four levels result in 144 combinations. A comprehensive full factorial experiment is carried out for all 144 combinations on 27 groups of problems. All these 144 combinations are tested on 270 problem instances developed and the corresponding makespan values are obtained. Four-factor ANOVA is carried out on the results obtained to determine the significant factors for each of the 27 problem groups. It is obvious that the makespan values of different problem in these problem groups will differ. This implies that there is no need to compare and prove the statistical difference of makespan values of different problems in these problem groups. Hence, it is more appropriate to test the significance of each of the factors for each of the 27 groups of problems separately. The different factors and their levels used in the analysis are provided in Table 5.

Table 5 Experimental factors and their levels

Factor	*Levels*			
Population type, A	Random	SRA	SRA+ Random	-
Crossover type, B	PMX	SJ2OX	SB2OX	-
Probability of crossover, P_c, C	0.1	0.2	0.3	0.4
Probability of mutation, p_m, D	0.1	0.2	0.3	0.4

All the factors are used for optimising the GA developed. But, the first factor, namely, the population type can also be used for validating the hybrid GA. If the analysis reveals that population type is significant, it means changing the initial population will affect the performance of the algorithm. Moreover, if SRA is the best setting for the initial

population, it validates the superiority of the hybrid GAs. Thus, the full factorial statistical experiment is carried out with two objectives:

1 to validate the superiority of the hybrid algorithms

2 to optimise the parameters of GA.

5 Results and discussion

Nine different variations of GA at 16 different combinations of mutation and crossover probabilities, presented in the study are tested for 27 groups of problems with ten problem instances in each. The makespan results obtained are analysed statistically. Since the experiment is carried out on problem sets of varying sizes, the makespan values vary considerably from problem to problem. Normalisation of parameters will be more appropriate for the comparison of performance. Hence, a percentage deviation measure, relative performance index (RPI) is used for the purpose of normalisation. For calculating RPI, a base makespan sequence is required. The constructive algorithm NEHRB (Rios-Mercado and Bard, 1998) is used for this purpose. GAs are improvement algorithms whereas NEHRB is a constructive algorithm. It may be noted that NEHRB is used not for comparison purpose but only as a base on which improvement of GA can be measured. A positive deviation indicates that there is an improvement over NEHRB; otherwise there is no improvement. The percentage deviation from NEHRB can be expressed as follows

$$\text{Relative performance index} = \frac{C_{\max} NEHRB - C_{\max} GA}{C_{\max} NEHRB}$$

where,

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

$C_{\max}GAB$ makespan values found using GA.

RPI values are calculated for all the problem groups. Each problem group is determined by the number of machines, number of jobs and the setup time percentage. The analysis is carried out for each problem group separately and the best parameter values are found out for each. The analysis is carried out in two stages. In the first stage, a four-factor ANOVA is carried out and the significant factors are found out for each of the 27 problem groups. In the second stage, the best parameter settings are identified for the significant factors found in the first stage. The analyses are performed using STATGRAPHICS 5.1.

5.1 Stage 1: four-factor ANOVA

A four-factor ANOVA is carried out for all the 27 groups of problems separately to determine the significant factors and interactions. Statistically, a factor/interaction is significant implies that the factor/interaction has a significant effect on the response variable. The F-ratios for each factor and for each interaction up to three factors are found out. This forms the basis for deciding the significance. The level of significance for the

test has been chosen as 5% ($\alpha = 0.05$). All the 27 instances are analysed separately using this method. The significant factors and interaction are found out for each case. The results obtained for the problem set (20 × 5 at 50% setup time) are shown in Table 6 as an example. Due to space limitations, the outputs for the remaining sets are not provided.

Table 6 ANOVA results for the problem with 20 jobs, 5 machines at 50% setup time

Source of variation		*DOF*	*SUMSQR*	*MEANSQR*	*F value*	*P Value*
Main effects	A: Initial population	2	0.263541	0.131770	113.2744*	0.0001
	B: Crossover operator	2	0.000691	0.000345	0.2972	0.7429
	C: Crossover probability	3	0.004681	0.001560	1.3412	0.2594
	D: Mutation probability	3	0.084729	0.028243	24.2788*	0.0001
Interactions	AB	4	0.000300	0.000075	0.0644	0.9924
	AC	6	0.001516	0.000253	0.2173	0.9714
	AD	6	0.018012	0.003000	2.5809*	0.0173
	BC	6	0.000903	0.000150	0.1294	0.9927
	BD	6	0.001756	0.000292	0.2516	0.9587
	CD	9	0.003244	0.000360	0.3098	0.9719
	ABC	12	0.002334	0.000194	0.1672	0.9994
	ABD	12	0.004814	0.000401	0.3448	0.9806
	ACD	18	0.009059	0.000503	0.4326	0.9813
	BCD	18	0.014119	0.000784	0.6743	0.8391
	ABCD	36	0.017480	0.000485	0.4174	0.9991

Note: *indicates the F-ratio significant at 5% significance level.

From Table 6, it can be inferred that the initial population type (A), the probability of mutation (D) and the interaction between these factors (AD) are significant. Interestingly, the crossover type, a prominent factor in GA, does not come out as a significant factor in the analysis. The main reason for this is the similarity between the crossover operators PMX, SJ2OX and SB2OX. The solution similarity between the crossover operators, SJOX and SBOX has been explained in Section 3.2. Moreover, if same jobs are not found in the parents, SJ2OX and SB2OX function like PMX. Even though same jobs or job blocks can be found in the two sequences selected as parents, it is quite unlikely to find them in exactly identical positions. These explanations lead to the conclusion that in most of the situations, the crossover operator will function like PMX even when the crossover operator selected is SJ2OX or SB2OX. The other factor, probability of mutation is also not significant for this particular group of problem. In a similar manner, all the 27 groups of problems are analysed. Table 7 shows the consolidated summary of the significant factors and the significant interactions for all the 27 groups considered in the experiment.

The 'NS' entry in Table 7 indicates that the factor is not significant. For each problem group, the significant factors are ranked according to their relative order of significance. The number entry in Table 7 shows the ranking of factors for that particular problem. Three factor interactions are not shown in Table 7 as they are not found significant in any of the cases considered. The type of crossover operator is found to be not significant for any of the problem groups considered. The reason is the similarity between the crossover operators considered. Two of the two-factor interactions AC and AD are found to be

significant for some of the problems. AC and AD represent the interaction of probability of crossover (C) and probability of mutation (D) with the population type (A) respectively.

Table 7 Consolidated summary of the significant factors and interactions for the problem groups

Problem size $n \times m$	*Setup time level*	*A*	*B*	*C*	*D*	*AB*	*AC*	*AD*	*BC*	*BD*	*CD*
20 × 5	50%	1	NS	-	2	NS	NS	3	NS	NS	NS
	100%	1	NS	-	2	NS	NS	3	NS	NS	NS
	150%	1	NS	-	2	NS	NS	3	NS	NS	NS
20 × 10	50%	2	NS	3	1	NS	NS	NS	NS	NS	NS
	100%	1	NS	3	2	NS	NS	NS	NS	NS	NS
	150%	1	NS	3	2	NS	NS	NS	NS	NS	NS
20 × 20	50%	3	NS	2	1	NS	NS	NS	NS	NS	NS
	100%	1	NS	3	2	NS	NS	NS	NS	NS	NS
	150%	1	NS	NS	2	NS	NS	3	NS	NS	NS
50 × 5	50%	1	NS	3	2	NS	5	4	NS	NS	NS
	100%	1	NS	3	2	NS	5	4	NS	NS	NS
	150%	1	NS	3	2	NS	5	4	NS	NS	NS
50 × 10	50%	1	NS	3	2	NS	NS	4	NS	NS	NS
	100%	1	NS	3	2	NS	5	4	NS	NS	NS
	150%	1	NS	3	2	NS	5	4	NS	NS	NS
50 × 20	50%	1	NS	3	2	NS	NS	NS	NS	NS	NS
	100%	1	NS	3	2	NS	NS	4	NS	NS	NS
	150%	1	NS	3	2	NS	5	4	NS	NS	NS
100 × 5	50%	1	NS	3	2	NS	4	5	NS	NS	NS
	100%	1	NS	3	2	NS	5	4	NS	NS	NS
	150%	1	NS	5	2	NS	4	3	NS	NS	NS
100 × 10	50%	1	NS	3	2	NS	4	5	NS	NS	NS
	100%	1	NS	3	2	NS	5	4	NS	NS	NS
	150%	1	NS	4	2	NS	3	5	NS	NS	NS
100 × 20	50%	1	NS	3	2	NS	4	5	NS	NS	NS
	100%	1	NS	3	2	NS	5	4	NS	NS	NS
	150%	1	NS	3	2	NS	4	5	NS	NS	NS

Note: NS indicates the factor/interaction is not significant at 5% significance level

5.2 Stage 2: finding the best parameter settings for the significant factors

The second stage involves finding the best settings of the significant factors which are identified for all the problem groups. Best setting refers to the levels at which each significant factor is set so that the best results are obtained with the GA. The preceding section has already found out the significant factors and interactions for each problem

group. All the problems are categorised into three cases based on the number of interactions as shown in Table 8. The analysis for each of these three cases is provided in following sub-sections.

Table 8 Problems grouped into three cases based on the number of significant interactions

Cases	*Problems*		*Significant factors and interactions*	*Remarks*
	Problem size n × m	*% setup time*		
1	20 × 10	50	D, A, C	Two-factor interactions are not significant
	20 × 10	100	A, D, C	
	20 × 10	150	A, D, C	
	20 × 20	50	D, C, A	
	20 × 20	100	A, D, C	
	50 × 20	50	A, D, C	
2	20 × 5	50	A, D, AD	One two-factor interaction effect is significant
	20 × 5	100	A, D, AD	
	20 × 5	150	A, D, AD	
	20 × 20	150	A, D, AD	
	50 × 10	50	A, D, C, AD	
	50 × 20	100	A, D, C, AD	
3	50 × 5	50	A, D, C, AD, AC	Two two-factor interaction effects are significant
	50 × 5	100	A, D, C, AD, AC	
	50 × 5	150	A, D, C, AD, AC	
	50 × 10	100	A, D, C, AD, AC	
	50 × 10	150	A, D, C, AD, AC	
	50 × 20	150	A, D, C, AD, AC	
	100 × 5	50	A, D, C, AC, AD	
	100 × 5	100	A, D, C, AD, AC	
	100 × 5	150	A, D, AD, AC, C	
	100 × 10	50	A, D, C, AC, AD	
	100 × 10	100	A, D, C, AD, AC	
	100 × 10	150	A, D, AC, C, AD	
	100 × 20	50	A, D, C, AC, AD	
	100 × 20	100	A, D, C, AD, AC	
	100 × 20	150	A, D, C, AC, AD	

Note: Factors: A − initial population; B − crossover operator: C − crossover probability: D − mutation probability.

5.2.1 Case 1: no two-factor interaction is significant

There are six problem groups that come under this category. These are three 20 × 10 size problems at 50%, 100% and 150% setup time proportions, two 20 × 20 size problems at 50% and 100% setup time proportions and one 50 × 20 size problem at 50% setup time

proportion. When no two-factor interaction is significant, each factor can be set independently at the best level. This can be done using a means plot. A means plot is a graph of response variable against factor levels. The response variable value used is an average of all the responses for that level at all combinations of other factors. Here, RPI values are taken as response variable in this experiment. For example, the procedure is explained with the help of the problem, 20 × 10 at 50% setup time proportion. For the problem set considered, factors A, C, and D representing initial population type, crossover probability, and mutation probability respectively are significant with no significant interaction among these factors.

Figure 3 Means plot of factor A (population type) for 20 × 10 problem at 50% setup time

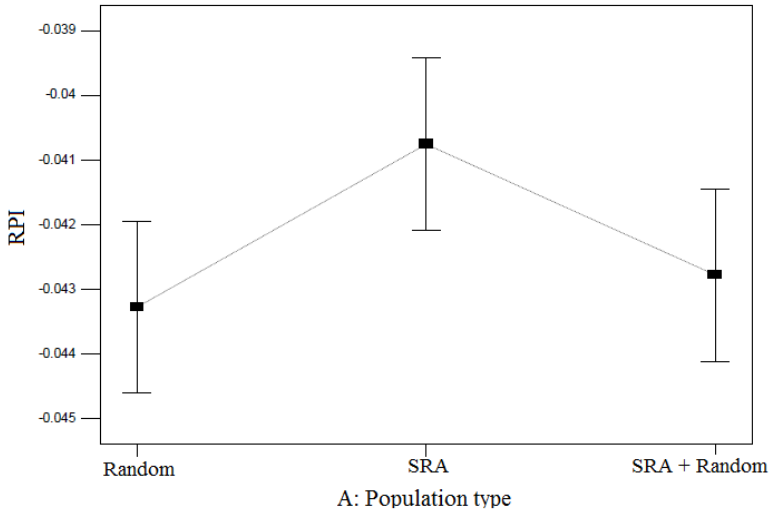


Figure 4 Means plot of factor C (Crossover probability) for 20 × 10 problem at 50% setup time

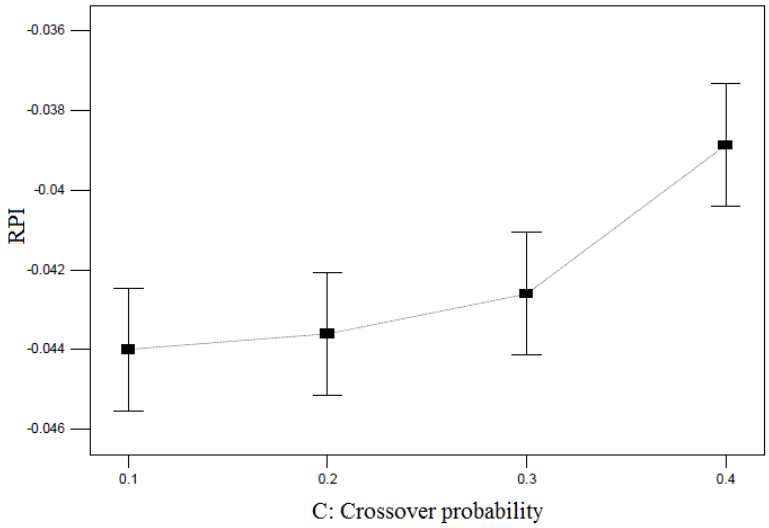
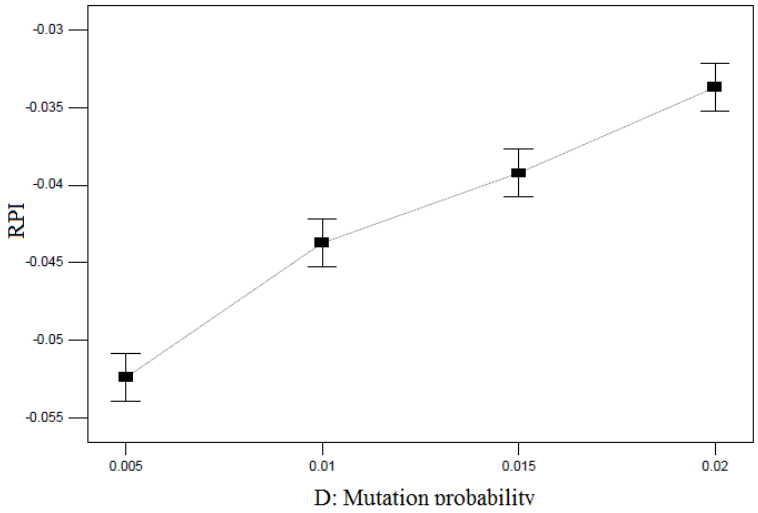


Figure 5 Means plot of factor D (Mutation probability) for 20 × 10 problem at 50% setup time



Figures 3 to 5 show the means plot for the three factors. The highest level of percentage improvement is identified from the respective plots, and the levels corresponding to the highest point is chosen as the best parameter setting for the factor. The best parameter values identified for the 20 × 10 problem at 50% setup time are as follows.

- Population type: SRA
- Crossover probability: 0.4
- Mutation probability: 0.02

The same procedure is repeated for the other problems in Case 1 and the results are presented in Table 9.

Table 9 Best setting of parameters for Case 1 problems

Problem $n \times m$	*% setup time*	*Best settings for factor*		
		A	*C*	*D*
20 × 10	50	SRA	0.4	0.02
20 × 10	100	SRA	0.4	0.02
20 × 10	150	SRA	0.3	0.02
20 × 20	50	SRA	0.4	0.02
20 × 20	100	SRA	0.4	0.02
50 × 20	50	SRA	0.4	0.02

Table 9 shows that for the factor A (the population type), SRA is the best setting for all the problems. The crossover probability is different for different problems in the set but, the mutation probability remains the same.

5.2.2 Case 2: one two-factor interaction is significant

The problems that come under this category are the 20 × 5 size problem at 50%, 100% and 150% setup time proportions, 20 × 20 size problem at 150% setup time proportion, 50 × 10 size problem at 50% setup time proportion and 50 × 20 size problem at 100% setup time proportion. Here, one two-factor interaction is significant and hence the levels of factors cannot be set independently. A multiple comparison test is required to find the best settings for the significant factors. STATGRAPHICS 5.1 software is used to carry out the multiple comparison tests. The test uses the Fishers least significant difference (LSD) test. The test compares the means obtained using different settings and checks whether there exists a significant difference between them. In the test, the difference between means and pairs of settings are computed and the difference is said to be significant if it exceeds the LSD value. Based on this test, homogeneous groups are identified. Homogeneous groups are groups within which there is no statistically significant difference between means. The factor levels can be set at any level corresponding to the levels of the most prominent homogeneous group, i.e., the one in which mean makespan is minimum. The procedure is explained with the 20 × 5 problem set at 50% setup time proportion. Here, A (population type), D (mutation probability) and AD interactions are significant. The factor A has three levels and factor D has four levels. Thus, 12 different combinations of A and D are possible for the interaction AD. For example, A1D1 represents both A and D at first level, i.e., population type is Random and probability of mutation is 0.005 and A3D4 represents both at last levels, i.e., population type is SRA+ Random and probability of mutation is 0.02. The inputs for the analysis are the makespan values found out at each combination. Each combination has ten makespan values as input (ten problem instances in each combination). The average of these makespan values at each combination is used in the LSD procedure. The results of statistical multiple comparison test carried out for the above problem are presented in Table 10.

Table 10 Multiple comparison test results for 20 × 5 size Case 2 problem at 50% setup time level

Factor level combinations	*Mean*	*Homogeneous groups*							
A2D4	1700.05	a							
A2D3	1708.95	a	b						
A3D4	1710.97	a	b	c					
A2D2	1712.60	a	b	c					
A3D3	1718.05		b	c	d				
A2D1	1725.52		b	c	d	e			
A3D2	1726.85			c	d	e			
A3D1	1733.19				d	e	f		
A1D4	1740.86					e	f		
A1D3	1749.43						f		
A1D2	1769.55							g	
A1D1	1794.83								h

Table 10 shows the factor level combinations, the average makespan values and the homogenous groups. The combinations with the same lower case letters represent homogeneous groups. Here, A2D4, A2D3, A3D4, and A2D2 belong to the same group labelled 'a' and A2D3, A3D4, A2D2, A3D3, and A2D1 belong to the same group labelled 'b' and so on. Any combination in group 'a' can be set as the best factor level, but for the sake of uniformity, the combination with the least average makespan in group 'a' is taken as the best setting. In this case, A2D4 turns out to be the best. So, the best setting is factor A at second level and factor D at fourth level as follows.

- Population type: SRA
- Mutation probability: 0.02

The procedure is repeated for obtaining the factor levels for the remaining problem sizes. The consolidated results are presented in Table 11.

Table 11 Best setting of parameters for Case 2 problems

Problem	*% setup time*	*Best settings for factor*		
		A	*C*	*D*
20 × 5	50	SRA	NS	0.02
20 × 5	100	SRA	NS	0.02
20 × 5	150	SRA	NS	0.02
20 × 20	150	SRA	NS	0.02
50 × 10	50	SRA	0.4	0.02
50 × 20	100	SRA	0.4	0.02

Note: NS indicates the factor/interaction is not significant

Table 11 shows that for factor A (population type), SRA is the best setting for all the problems. The crossover probability is significant only for some of the problems in the set. The mutation probability is significant for all the problems with the probability value set at 0.02.

5.2.3 Case 3: two two-factor interactions are significant

The third case involves problems which have two two-factor interactions significant. The problems that come under this category are 50 × 5 size problem at 50%, 100% and 150% setup time proportions, 50 × 10 size problem at 100% and 150% setup time proportions, 50 × 20 size problem at 150% setup time proportion, and 100 × 5, 100 × 10 and 100 × 20 size problem at 50%, 100% and 150% setup time proportions. Since two factor interactions are significant, the procedure of Case 2 can be applied here. The procedure is described below with the help of 100 × 5 problem set at 50% setup time level. The two interactions AC and AD are significant in the problem considered while the interaction AC is slightly more significant than the interaction AD. The levels for A and C are set first by analysing AC interaction, and then the corresponding level for D is set by analyzing AD interaction. The results of the multiple comparison tests are provided in Tables 12 and 13.

Table 12 Multiple comparison test results for Case III problem − AC factor level combination

Factor level combinations, AC	*Mean*	*Homogeneous groups*				
A2C3	6789.39	a				
A2C4	6729.93	a				
A2C1	6798.27	a				
A2C2	6801.45	a				
A3C4	6805.72	a				
A3C3	6817.85	a				
A3C2	6820.02	a				
A3C1	6828.63	a				
A1C4	7409.63		b			
A1C3	7436.48		b	c		
A1C2	7475.73			c		
A1C1	7529.05					d

Table 13 Multiple comparison test results for Case 3 problem − AD factor level combination

Factor level combinations, AD	*Mean*	*Homogeneous groups*				
A2D4	6778.06	a				
A2D3	6785.43	a				
A2D2	6800.88	a				
A3D4	6805.16	a	b			
A3D3	6809.02	a	b			
A3D2	6815.52	a	b			
A2D1	6817.68	a	b			
A3D1	6842.52		b			
A1D4	7401.38			c		
A1D3	7434.97			c	d	
A1D2	7475.48				d	
A1D1	7540.07					e

From Table 12, the levels of A and C are set as A2C3, i.e., initial population type is SRA corresponding to A2 and the crossover probability is 0.3 corresponding to C3. Now, the level of D has to be set such that the initial population type is A2. From Table 13, it can be found that A2D4 is the best setting. Hence, the level of D is set at D4, i.e., mutation probability is 0.02. Thus, the best settings for the example problem are as follows.

- Population type: SRA
- Crossover probability: 0.3
- Mutation probability: 0.02

The above procedure is repeated for all the problem sets of this category. Table 14 summarises the results for all the problems in Case 3.

Table 14 Best setting of parameters for Case 3 problems

Problem $n \times m$	*% setup time*	*Best settings for factor*		
		A	*C*	*D*
50 × 5	50	SRA	0.4	0.02
50 × 5	100	SRA	0.1	0.02
50× 5	150	SRA	0.3	0.02
50 × 10	100	SRA	0.4	0.02
50 × 10	150	SRA	0.3	0.02
50 × 20	150	SRA	0.4	0.02
100 × 5	50	SRA	0.3	0.02
100 × 5	100	SRA	0.1	0.02
100 × 5	150	SRA	0.2	0.02
100 × 10	50	SRA	0.4	0.02
100 × 10	100	SRA	0.4	0.02
100 × 10	150	SRA	0.1	0.02
100 × 20	50	SRA	0.4	0.02
100 × 20	100	SRA	0.4	0.02
100 × 20	150	SRA	0.2	0.02

Table 14 shows that for factor A (the population type), SRA is the best setting for all the problems. The crossover probability is different for different problems in the set but, the mutation probability remains the same. The statistical analyses are carried out for all the problems and the best parameter settings are identified for each problem. The summary of the results are presented in Table 15.

Table 15 Summary of results

Factor	*Levels*	*Number of problem groups for which the factor level is the best setting*
Type of population	Random	0
	SRA	27
	SRA+ Random	0
Crossover probability	0.100	3
	0.200	2
	0.300	4
	0.400	14
Mutation Probability	0.005	0
	0.010	0
	0.015	0
	0.020	27

For the population type, regardless of the problem set, SRA population always gives better results compared to random initialization or a combination of the two. This leads to the major finding that the hybrid GA (genetic algorithm which uses SRA heuristic based

initial population) performs better in all the problem sets. This proves the superiority of the hybrid genetic algorithm. The other two factors that are found to be significant are the mutation probability and the crossover probability. The study also identifies the best crossover probability and the best mutation probability for each problem group. Interestingly, crossover operator type is not found to be significant for any of the problem groups. The best mutation probability is 0.02 in every problem instance. But, crossover probability varies with problem sets and percentage of setup time, with 0.4 being the best value in the majority of the problem sets.

6 Conclusions

This paper has dealt with the development and analysis of hybrid GAs for flow shop scheduling problem in a SDST environment. Twenty-seven different sizes of SDST flow shop scheduling problems are used for experimentation. Different combinations of jobs (20, 50 and 100) and machines (5, 10 and 20) are used at three different setup time levels (50%, 100% and 150%). The factors (parameters of GA) considered in the experimental analysis are the population type, the crossover operator type, the crossover probability and the mutation probability. Three different types of initial population and three different types' crossover operators are used in the study. The three different initial populations are Random population, SRA population and SRA population + Random population. The SRA heuristic used to provide better initial population for genetic algorithm. The possible combinations of population types and crossover operators result in nine different variations of GA. The GAs which uses either SRA population or SRA population + Random population are called hybrid genetic algorithms.

A four-factor ANOVA is carried out initially for each of the 27 problem groups separately to identify the significant factors and their interactions. Based on the results obtained from ANOVA, the 27 problems are grouped into three categories. The best parameters for the Case 1 are determined from the means plot of the significant parameters. For Case 2 and Case 3, a multiple comparison test is employed to arrive at the best settings for the significant factors. The best settings for the genetic algorithm parameters are identified for each of the groups of problems.

The development of the SRA-based hybrid genetic algorithm is the major contribution of the present study. Another relevant finding from the results obtained is that the best parameter settings of the GA changes with the variation in problem size and also with setup time level. The non-significance of the type of crossover operators in the performance of GA-based algorithms is another relevant finding from the present study. Moreover, GAs based on SRA population always provide better performance regardless of the problem set. This proves the superiority of the hybrid GA (genetic algorithm which uses SRA heuristic-based initial population) for all the problem sets.

One of the standard assumptions in operations scheduling is that setup times are included in the processing times. However, setup time is encountered in manufacturing firms such as printing, plastics manufacturing, metal and chemical processing, paper industry etc. The trend in manufacturing towards production of batches or unit production to satisfy demand and avoid inventory has made more relevant the scheduling problem with SDSTs. In such cases, improving the schedules by as little as one percent can have a significant financial impact. The present study proposes a hybrid genetic algorithm with

the objective minimising the total completion time of jobs (makespan) which in many ways influences the lead time. Reductions in manufacturing lead time can generate numerous benefits including lower inventory levels, improved quality, lower costs, and lesser forecasting error. Hence, the present study has significant implications for operations managers.

The high computational requirements of the SDST flow shop scheduling problem make the metaheuristics as a practical and effective solution methodology. Since the hybrid algorithms emerge as more effective in the present study, different types of hybrid metaheuristics can be experimented for the SDST flow shop scheduling problem. Considering the importance of due-date fulfilment in determining the service level, more studies are required using the due-date consideration. The customer-driven performance measures such as minimising the total earliness and tardiness, number of tardy jobs, and maximum tardiness can be investigated. Since SDST environment is not restricted to manufacturing, various SDST scheduling problems encountered in service operations can be analysed.

Acknowledgements

The authors express their 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

AitZai, A., Benmedjdoub, B. and Boudhar, M. (2012) 'A branch and bound and parallel genetic algorithm for the job shop scheduling problem with blocking', *International Journal of Operational Research*, Vol. 14, No. 3, pp.343−365.

Allahverdi, A. and Soroush, H.M. (2008) 'The significance of reducing setup times/setup costs', *European Journal of Operational Research*, Vol. 187, No. 3, pp.978–984.

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.

Amiri, M., Abtahi, A-M. and Damghani, K. (2013) 'Solving a generalised precedence multi-objective multi-mode time-cost-quality trade-off project scheduling problem using a modified NSGA-II algorithm', *International Journal of Services and Operations Management*, Vol. 14, No. 3, pp.355−372.

Amuthakkannan, R., Babu, C.K. and Kannan, S.M. (2010) 'An approach to the minimisation of makespan in the textile industry using ant colony optimisation', *International Journal of Services and Operations Management*, Vol. 7, No. 2, pp. 215−230.

Arumugam, C., Raja, K. and Selladurai, V. (2007) 'Agility in two-stage hybrid flow shop parallel machine scheduling through simulated annealing', *International Journal of Services and Operations Management*, Vol. 3 No. 3, pp. 332−354.

Bouabda, R., Jarboui, B., Eddaly, M. and Rebaï, A. (2011) 'A branch and bound enhanced genetic algorithm for scheduling a flowline manufacturing cell with sequence dependent family setup times', *Computers & Operations Research*, Vol. 38, No. 1, pp.387–393.

Boudhar, M. and Tchikou, H. (2010) 'Scheduling with arranged multi-purpose machines', *International Journal of Operational Research*, Vol. 8, No. 3, pp.379−397.

Bozorgirad, M.A. and Logendran, R. (2012) 'Sequence-dependent group scheduling problem on unrelated-parallel machines', *Expert Systems with Applications*, Vol. 39, No. 10, pp.9021–9030.

Charles, V. and Udhayakumar, A. (2012) 'Genetic algorithm for chance constrained reliability stochastic optimisation problems', *International Journal of Operational Research*, Vol. 14, No. 4, pp.417−432.

Defersha, F.M. and Chen, M. (2011) 'A genetic algorithm for one-job m-machine flowshop lot streaming with variable sublots', *International Journal of Operational Research*, Vol. 10, No. 4, pp.458−468.

Eren T.A. (2011) 'Bi-criteria m-machine flow shop scheduling with sequence-dependent setup times', *Applied Mathematical Modeling*, 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.

Gómez-Gasquet, P., Andrés, C. and Larioa, F.C. (2012) 'An agent-based genetic algorithm for hybrid flowshops with sequencedependentsetup times to minimise makespan', *Expert Systems with Applications*, Vol. 39, No. 9, pp.8095−8107.

Gupta, J.N.D. (1986) 'Flow shop schedules with sequence dependent setup times', *Journal of Operations Research Society Japan*, Vol. 29, No. 3, pp.206−19.

Joo, S. and Min, H. (2011) 'A multiple objective approach to scheduling the preventive maintenance of modular aircraft components', *International Journal of Services and Operations Management*, Vol. 9, No. 1, pp.18−31.

Joseph, O.A. and Sridharan, R. (2011) 'Development of simulation-based metamodels for the analysis of routing flexibility, sequencing flexibility and scheduling decision rules on the performance of an FMS', *International Journal of Operational Research*, Vol. 12, No. 3, pp.333−361.

Kanagasabapathi, B., Rajendran, C. and Ananthanarayanan, K. (2010) 'Scheduling in resource-constrained multiple projects to minimise the weighted tardiness and weighted earliness of projects', *International Journal of Operational Research*, Vol. 7, No. 3, pp.334−386.

Kanyalkar, A.P. and Adil, G.K. (2009) 'Determining the optimum safety stock under rolling schedules for capacitated multi-item production systems', *International Journal of Services and Operations Management*, Vol. 5, No. 4, pp.498−519.

Kasemset, C. and Kachitvichyanukul, V. (2012) 'A PSO-based procedure for a bi-level multi-objective TOC-based job-shop scheduling problem', *International Journal of Operational Research*, Vol. 14, No. 1, pp.50−69.

Komarudin and Wong, K.Y. (2012) 'Parameter tuning of ant system using fuzzy logic controller', *International Journal of Operational Research*, Vol. 15, No. 2, pp.125−135.

Kumar, P.N.R. and Sivakumar, A.I. (2013) 'Simulated annealing−based algorithm for the capacitated hub routing problem', *International Journal of Services and Operations Management*, Vol. 14, No. 2, pp.221−235.

Laha, D. and Mandal, P. (2007) 'Improved heuristically guided genetic algorithm for the flow shop scheduling problem', *International Journal of Services and Operations Management*, Vol. 3, No. 3, pp.316−331.

Lawrynowicz, A. (2007) 'Production planning and control with outsourcing using artificial intelligence', *International Journal of Services and Operations Management*, Vol. 3, No. 2 pp.193−209.

Lessmann, S., Caserta, M. And Arango, I.M. (2011) 'Tuning metaheuristics: A data mining based approach for particle swarm optimization', *Expert Systems with Applications*, Vol. 38, No. 10, pp.12826–12838.

Magableh, G.M. and Mason, S.J. (2010) 'Minimising earliness and tardiness on parallel machines with sequence−dependent setups', *International Journal of Operational Research*, Vol. 8, No. 1, pp.42−61.

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.

Nguyen, S. and Kachitvichyanukul, V. (2012) 'An efficient differential evolution algorithm for multi-mode resource-constrained project scheduling problems', *International Journal of Operational Research*, Vol. 15, No. 4, pp.466−481.

Pal, B.B. and Gupta, S. (2012) 'A genetic algorithm-based fuzzy goal programming approach for solving fractional bilevel programming problems', *International Journal of Operational Research*, Vol. 14, No. 4, pp.453−471.

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.2609−23.

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.475−83.

Ramanan, T.R., Sridharan, R., Shashikant, K.S. and Haq, A.N. (2011) 'An artificial neural network based heuristic for flow shop scheduling problems', *Journal of Intelligent Manufacturing*, Vol. 22, No. 2, pp.279−88.

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 flow shops with sequence dependent setup times and machine eligibility', *European Journal of Operational Research*, Vol. 169, No. 3, pp.781–800.

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.

Sabouni, M.T.Y. and Logendran, R. (2013) 'Carryover sequence-dependent group scheduling with the integration of internal and external setup times', *European Journal of Operational Research*, Vol. 224, No. 1, pp.8−22.

Sobhani, F. and Wong, K.Y. (2013) 'Optimisation of distribution quantity in a multi-product multi-period supply chain using genetic algorithm', *International Journal of Services and Operations Management*, Vol. 14, No. 3, pp.277− 297.

Solimanpur, M. and Rastgordani, R. (2012) 'Minimising tool switching and indexing times by ant colony optimisation in automatic machining centres', *International Journal of Operational Research*, Vol. 13, No. 4, pp.465−479.

Soroush, H.M. and Alqallaf, F.A. (2009) 'Minimising a weighted quadratic function of job lateness in the stochastic single machine scheduling problem', *International Journal of Operational Research*, Vol. 6, No. 4, pp.538−572.

Srinivasan, G. and Srirangacharyulu, B. (2012) 'Minimising variance of job completion times in a single machine', *International Journal of Operational Research*, Vol. 13, No. 1, pp.110−127.

Taillard, E. (1993) 'Benchmarks for basic scheduling problems', *European Journal of Operational Research*, Vol. 64, No. 2, pp.278−85.

Tanaka, S. and Araki, M. (2013) 'An exact algorithm for the single-machine total weighted tardiness problem with sequence-dependent setup times', *Computers & Operations Research*, Vol. 40, No. 1, pp.344−352.

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*, pp.1−17, DOI: 10.1007/s00170-012-4571-8.

Vinay, V.P. and Sridharan, R. (2012) 'Taguchi method for parameter design in ACO algorithm for distribution–allocation in a two-stage supply chain', *International Journal of Advanced Manufacturing Technology* online first, pp.1−11, DOI: 10.1007/s00170-012-4104-5.

Vinod, V. and Sridharan, R. (2008) 'Dynamic job-shop scheduling with sequence-dependent setup times: simulation modeling and analysis', *International Journal of Advanced Manufacturing Technology*, Vol. 36, Nos. 3−4, pp.355−72.

Vinod, V. and Sridharan, R. (2008) 'Scheduling a dynamic job shop production system with sequence-dependent setups: an experimental study', *Robotics and Computer-Integrated Manufacturing*, Vol. 24, No. 3, pp.435–449.