Chapter 21
Development and Analysis of a Discrete Particle Swarm Optimisation for Bi-criteria Scheduling of a Flow Shop with Sequence-Dependent Setup Time

V. Anjana, R. Sridharan and P. N. Ram Kumar

Abstract Most studies in flow shop scheduling neglect the setup times or consider the setup times along with the processing times. However, in industries that manufacture paint, textiles, ceramic tiles, etc., the setup times are significant and are sequence dependent. This paper addresses the problem of scheduling a flow shop operating in a sequence-dependent setup time (SDST) environment considering the objectives, namely minimisation of makespan and mean tardiness. The evolutionary method of discrete particle swarm optimisation (DPSO) based on weighted approach is developed and applied to SDST benchmark problems of flow shop scheduling. The efficacy of the metaheuristic is compared with that of a hybrid genetic algorithm, and it is observed that on an average, the proposed DPSO provides an improvement of 7.8, 22.3 and 11.3% in the values of mean ideal distance, computational time and diversification matrix, respectively. For most problems, the proposed DPSO performs superior to the hybrid genetic algorithm.

Keywords Permutation flow shop · Sequence-dependent setup time · Discrete particle swarm optimisation · Hybrid genetic algorithm

21.1 Introduction

Flow shop is a shop floor configuration where all the jobs share the same sequence of processing. Every job has a deterministic span of time for completing its operation in each machine known as the processing time [1]. For processing a job, setup activities needed to be performed on each machine and the time incurred for doing these preparations is known as the setup time. Setup includes activities such as obtaining tools, cleaning the machines, setting the machines, fixing and removing jobs and returning

V. Anjana (✉) · R. Sridharan
Mechanical Engineering Department, National Institute of Technology Calicut, Calicut, Kerala 673601, India
e-mail: av.anjanam@gmail.com

P. N. Ram Kumar
Indian Institute of Management Kozhikode, Calicut, Kerala 673570, India

© Springer Nature Singapore Pte Ltd. 2020

M. S. Shunmugam and M. Kanthababu (eds.), *Advances in Simulation, Product Design and Development*, Lecture Notes on Multidisciplinary Industrial Engineering,
https://doi.org/10.1007/978-981-32-9487-5_21

tools. [2]. From the literature, it is observed that most studies in flow shops neglect the setup time or consider the setup time along with the processing time. However, in real-life situations, the presence of setup times cannot be neglected, especially in industries that manufacture paint, tiles, pharmaceuticals, automobiles, drugs and cosmetics, where the setup times are sequence dependent. In a sequence-dependent setup time (SDST) environment, the setup times of jobs depend on the current job to be processed and also on the previous job that has already been processed [3].

The studies on flow shops focus on optimising a single objective such as minimisation of makespan, total flow time, tardiness and number of setups [1, 3–5]. In real-life situations, a decision-maker has to deal with the optimisation of more than one objective. The presence of setup times and the attainment of multiple objectives increase the complexity of the scheduling problem. The solution to the multi-objective problem is obtained as a set of Pareto-optimal solutions or non-dominated solutions. A solution is said to be non-dominating, if it is not dominated by any other solutions of the multi-objective optimisation problem. Each solution in the Pareto-front has a better value for any one of the objectives and is a solution to the problem [6]. The decision-maker has to select a solution from the Pareto-front depending on the importance of the objective to be achieved. Researchers and practitioners adopt the weighted and non-weighted approach for solving the multi-objective problems. In a weighted approach, weights are attached to the objectives such that the multi-objective problem is converted to an equivalent single-objective problem. Hence, the present study focuses on the development of discrete particle swarm optimisation (DPSO) metaheuristic based on the weighted approach for solving the multi-objective SDST flow shop scheduling problem. The efficacy of the proposed metaheuristic is compared with the hybrid genetic algorithm.

The weighted approach has been adopted by many researchers like Rajendran and Zieglar [7], Eren and Guner [8], Eren [9] and Dhingra and Chandna [10] for solving the SDST flow shops with multiple objectives. Rajendran and Zieglar develop heuristics for scheduling an SDST flow shop with the objective of minimising weighted flow time and weighted tardiness. The researchers compare the performance of their heuristic with an existing heuristic, random search procedure and a greedy local search. The performance analysis reveals the better performance of the proposed heuristic. Eren and Guner address the scheduling problem in an SDST flow shop with the objective of minimising the weighted sum of total completion time and total tardiness. The authors develop an integer programming model for solving problems of up to 12 jobs. For solving larger-size problems, special heuristic algorithms and Tabu search are developed. The results from the computational studies indicate that the algorithms are effective for solving problems up to 1000 jobs. Eren considers the scheduling of a two-machine SDST flow shop with the objective of minimising four criteria. An integer programming model and Tabu search are developed for solving the multi-objective problem of up to 1000 jobs. Dhingra and Chandna develop a hybrid genetic algorithm minimising the weighted sum of total weighted squared tardiness and makespan of an SDST flow shop.

It is observed from the literature that the works related to SDST flow shops with multiple objectives are less in number. Further, no works have been reported with

discrete particle swarm optimisation (DPSO) based on weighted approach for solving an SDST flow shop scheduling problem with multiple objectives. Thus, the objectives of the present study are as follows.

- Development of a metaheuristic based on discrete particle approach for scheduling an SDST permutation flow shop with the objective of minimising makespan and mean tardiness.
- Determination of the best set of parameters of the proposed metaheuristic.
- Experimentation of the metaheuristic using benchmark problems.
- Comparison of the proposed metaheuristic with a hybrid genetic algorithm.

The rest of the paper is organised as follows. The problem definition and the assumptions related to the study are presented in Sect. 21.2. Section 21.3 provides the detailed description of the proposed metaheuristic. Section 21.4 presents the method of determining the best set of parameters of the proposed metaheuristic. The experimentation details are described in Sect. 21.5. Section 21.6 presents the analysis of the results, and Sect. 21.7 provides the conclusion.

21.2 Problem Definition

The present study addresses the scheduling problem of an n job $\times$ m machine SDST flow shop with the objective of minimising makespan and mean tardiness. The assumptions made in the study are as follows.

- All the jobs are available for processing at time zero.
- The processing times of operations of jobs are known in advance.
- Setup times for operations are considered explicitly from processing time and are sequence dependent.
- Each machine can process only one job at a time.
- No pre-emption is allowed.
- The machines are continuously available, that is no breakdown of machines.

Notations

n	Number of jobs to be scheduled
m	Number of machines in the flow shop
p_{ji}	Processing time of job j on machine i
d_j	Due date of job j
C_j	Completion time of job j
s_{ijk}	Setup time on ith machine if job j is preceded by job k
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 partial sequence σ)
$q\ (\sigma_j, i)$	Completion time of job j on machine i, when the job is appended to partial sequence σ

$f_1(x)$ Makespan of sequence x
$f_2(x)$ Mean tardiness of sequence x
w_1 Weight assigned to makespan
w_2 Weight assigned to mean tardiness.

The objective function for an SDST flow shop scheduling problem is expressed as follows:

$$\text{Min } f(x) = w_1 \times f_1(x) + w_2 \times f_2(x) \tag{21.1}$$

Since a sequence-dependent flow shop is considered, the recursive equation for the completion time of job j on machine i is determined using Eq. 21.2.

$$q(\sigma_j, i) = \max\{q(\sigma, i) + s_{ijk}, q(\sigma_j, i-1)\} + p_{ji}, \tag{21.2}$$

where job k precedes job j; $q(\sigma_j, i)$ is the completion time of job j on machine i, when the job is appended to partial sequence σ; $(q(\sigma, i) + s_{ijk})$ denotes the sum of the completion time of processing of job k on machine i and the setup time for job j on machine i; $q(\sigma_j, i-1)$ is the completion time of the immediately preceding operation of job j on the previous machine; and p_{ji} is the processing time of job j on machine i.

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

$$C_j = q(\sigma_j, m), \tag{21.3}$$

where $q(\sigma_j, m)$ is the completion time of the last operation of job j on machine m.

The makespan for a sequence of jobs is given by

$$f_1 = \max(C_j, \quad j = 1, 2, \ldots, n) \tag{21.4}$$

The tardiness of a job is given by

$$t_j = \max\{0, (C_j - d_j)\} \tag{21.5}$$

The mean tardiness of a sequence of jobs is given by

$$f_2 = \frac{\sum_{j=1}^{n} t_j}{n} \tag{21.6}$$

21.3 Discrete Particle Swarm Optimisation

Particle swarm optimisation (PSO) developed by Kennedy and Eberhart [11] for optimising continuous linear functions mimics the social behaviour of birds gathering their food. PSO optimises a problem by having a population of candidate solutions and moving these particles around in the search space over the particle's position and velocity. The continuous PSO is not sufficient to solve the real-life problems with discrete problem features. Thus, the developers of PSO modified it to address the discrete problem, namely flow shop scheduling [12]. Discrete particle swarm optimisation (DPSO) is the discrete version of particle swarm optimisation. The difference between PSO and DPSO occurs in the representation of the solution. When PSO is applied for solving discrete optimisation problems (scheduling problems), the solution representation of PSO is modified to represent the discrete solutions [13]. In the present study, a variant of DPSO based on the weighted approach is developed for scheduling an SDST flow shop with the objective of minimising makespan and mean tardiness. The proposed metaheuristic is described in detail in Sect. 21.3.1.

21.3.1 The Proposed DPSO Metaheuristic

In DPSO, the initial population is considered as the swarm and each solution in the swarm is termed as the particle. The initial swarm for the present research is generated using the NEH heuristic [14] and the pair-wise interchange method. The generation of the initial swarm is followed by the computation of the objective function values of the particles in the swarm. Each particle in the swarm is represented as $X_1, X_2, X_3, \ldots,$ $X_{N,}$ where N denotes the number of particles in the swarm. The personal best matrix corresponds to the objective function values of each particle in swarm. The objective function values are determined as the weighted sum of the objective function values. The lowest value among the personal best values is considered as the global best. Once the personal best matrix is formed, the position of the particles is updated. In PSO, every move of the particle to the next position is influenced by its own previous position, the position of the neighbouring particles and the particle in the leading position. The position of the particles is obtained from the velocity components. The position update is performed by two types of crossover and a mutation operation. The types of crossover involved are social crossover and cognition crossover. The previous position, the position of the neighbouring particles and the particle in the leading position are given by the mutation operation, the cognition velocity component and the social velocity component, respectively. The three components are determined using the following relations.

The position update equation consists of three components:

$$\lambda_i^t = w \otimes F_1(X_i^{t-1}) \tag{21.7}$$

$$\delta_i^t = c_1 \otimes F_2(\lambda_i^t, p_i^{t-1}) \tag{21.8}$$

$$\mu_i^t = c_2 \otimes F_3(\delta_i^t, g_i^{t-1}) \tag{21.9}$$

The first component given by Eq. 21.7 represents the velocity of the particle. In Eq. 21.7, F_1 represents the mutation operator with a probability of w. A uniform random number r is generated between 0 and 1. If r is less than w, then the mutation operator is applied to generate a perturbed permutation of the particle, otherwise the particle is kept without any change. The second component obtained using Eq. 21.8 represents the cognition part of the particle. F_2 in Eq. 21.8 represents the cognition crossover operator with a crossover probability c_1. Here, λ_i^t and p_i^{t-1} are the two parents for crossover where λ_i^t is the particle obtained from mutation and p_i^{t-1} is the particle in the personal best matrix. The occurrence of this crossover operation depends on the random number generated. The third or social component is provided by Eq. 21.9 where F_3 and c_2 represent the crossover operator and social crossover probability, respectively. The parents for crossover are δ_i^t and g_i^{t-1} where δ_i^t is the particle obtained from the cognition crossover and g_i^{t-1} is the global best solution. The crossover operation depends on the random number generated. The objective function values of the velocity components are determined. Since the problem is of minimisation type, the least value among the three components is considered and the position is updated. A local search is performed on these solutions, which increases the diversity of the solutions. A non-dominant sorting procedure is applied to the solutions obtained from the local search, and the set of non-dominated solutions are updated in each generation. The solutions obtained from the local search become the swarm for the next generation. The procedure is repeated until it reaches the specified termination criteria.

21.3.2 Hybrid Genetic Algorithm

The hybrid genetic algorithm (HGA) is developed by combining the evolutionary method of genetic algorithm with a local search method. The initial population is generated using NEH and the pair-wise interchange method. The population is then subjected to genetic operators of selection, crossover and mutation. A local search is applied to the solutions obtained from mutation. The set of Pareto-optimal solutions is obtained by applying the non-dominant sorting algorithm to the offspring of mutation. The procedure is repeated for a specified number of generations. The hybrid GA is applied to the benchmark problems with the best set of parameters obtained from Taguchi's robust design and the utility index concept.

21.4 Parameter Configuration of DPSO

The parameters of the proposed DPSO include type of mutation, probability of mutation, type of cognition crossover, probability of cognition crossover, type of social crossover, probability of social crossover and the swarm size. The parameters of the DPSO metaheuristic are determined using Taguchi's robust design in combination with the concept of utility index [15, 16]. The parameters and their different levels are shown in Table 21.1. The L18 orthogonal array is selected, and the objective function values are determined for each trial. Once the objective function values are computed, the average response value of each objective function for each factor level is determined. The average values of the objectives for each factor level are presented in Table 21.2. The average response value corresponding to each level of the parameters is computed from the objective function values and is shown in Table 21.3.

The preference number for each objective function is obtained using Eq. 21.10.

$$P_b = Z \log \frac{y_b}{y_b'}, \tag{21.10}$$

where y_b is the value of the objective b, y_b' is the maximum or minimum acceptable value of the objective and Z is a constant. The value of Z has to be determined for computing the preference number for each objective. It is assumed that at optimum, value of the objective $P_b = 9$, and hence, the value of Z is computed as follows.

$$\text{At optimum, } y_b' \text{of objective } b,\ P_b = 9;\ \ Z = \frac{9}{\log \frac{y_b^*}{y_b'}}, \tag{21.11}$$

where y_b^* is the predicted optimal value of attribute b.

The predicted optimal value of makespan $= 1651.87 + 1645.\ 39 + 1643.81 + 1638.61 + 1645.72 + 1640.81 + 1642.64 - 3 \times 1655.48 = 6838.3$.

Table 21.1 Parameters and their levels

Sl. No.	Parameter	Code	Level		
			1	2	3
1	Mutation type	*A*	Shift	Swap	–
2	Mutation probability	*B*	0.1	0.2	0.3
3	Type of cognition	*C*	Single point	Two point	PMX
4	Cognition probability	*D*	0.7	0.8	0.9
5	Type of social crossover	*E*	Single point	Two point	PMX
6	Social crossover probability	*F*	0.7	0.8	0.9
7	Swarm size	*G*	20	30	50

Table 21.2 Average response value by factor levels

Sl. No.	Parameter							Average makespan	Average mean tardiness
	Type of mutation	Probability of mutation	Type of cognition crossover	Probability of cognition crossover	Type of social crossover	Probability of social crossover	Swarm size		
1	Shift	0.1	Single point	0.7	Single point	0.7	20	1663.67	308.23
2	Shift	0.1	Two point	0.8	Two point	0.8	30	1667.67	296.77
3	Shift	0.1	PMX	0.9	PMX	0.9	50	1682.50	288.40
4	Shift	0.2	Single point	0.7	Two point	0.8	50	1654.67	289.00
5	Shift	0.2	Two point	0.8	PMX	0.9	20	1659.33	292.80
6	Shift	0.2	PMX	0.9	Single point	0.7	30	1628.50	291.15
7	Shift	0.3	Single point	0.8	Single point	0.9	30	1669.67	299.63
8	Shift	0.3	Two point	0.9	Two point	0.7	50	1682.50	289.10
9	Shift	0.3	PMX	0.7	PMX	0.8	20	1623.33	307.60
10	Swap	0.1	Single point	0.9	PMX	0.8	30	1685.00	299.30
11	Swap	0.1	Two point	0.7	Single point	0.9	50	1685.00	292.65
12	Swap	0.1	PMX	0.8	Two point	0.7	20	1646.00	301.53
13	Swap	0.2	Single point	0.8	PMX	0.7	50	1654.67	301.97
14	Swap	0.2	Two point	0.9	Single point	0.8	20	1663.67	308.20
15	Swap	0.2	PMX	0.7	Two point	0.9	30	1635.50	315.68
16	Swap	0.3	Single point	0.9	Two point	0.9	20	1680.50	285.35
17	Swap	0.3	Two point	0.7	PMX	0.7	30	1569.50	281.05
18	Swap	0.3	PMX	0.8	Single point	0.8	50	1647.00	299.13

Table 21.3 Average objective value corresponding to each level

Level	Makespan							Mean tardiness						
	A	*B*	*C*	*D*	*E*	*F*	*G*	*A*	*B*	*C*	*D*	*E*	*F*	*G*
1	1659.09	1671.64	1668.03	1638.61	1659.58	1640.81	1656.08	295.85	297.81	297.25	299.04	299.83	295.51	300.62
2	1651.87	1649.39	1654.61	1657.39	1661.14	1656.89	1642.64	298.32	299.80	293.43	298.64	296.24	300.00	297.26
3		1645.42	1643.81	1670.44	1645.72	1668.75	1667.72		293.64	300.58	251.64	295.19	253.50	293.38
	Cell mean		1655.481					Cell mean		292.877				

The predicted optimal value of mean tardiness = 295.85 + 293.64 + 293.43 + 251.64 + 295.19 + 253.50 + 293.38 − 3 × 292.88 = 1098.

The preference number is determined for makespan and mean tardiness from the predicted optimal values using Eq. 21.10. The utility index is computed using Eq. 21.12.

$$U_d = \sum_{b=1}^{l} a_b P_b, \tag{21.12}$$

where a_b is the weight assigned to the objective b, P_b denotes the preference number of objective b, l is the number of objectives and d is the experiment number. The preference number and the utility index corresponding to each experiment of the orthogonal array are shown in Table 21.4. The average utility index corresponding to each level of parameters is determined and is provided in Table 21.5. From Table 21.5, it is observed that parameter B, that is the mutation probability, has the highest range, and hence, it is the influencing factor on the performance characteristics of the algorithm. The order of importance of the parameters on the performance of the algorithm can be listed as follows: mutation probability, probability of social

Table 21.4 Preference number and utility index

Experiment No.	Preference number		Utility index
	Makespan	Mean tardiness	
1	0.082	0.180	0.131
2	0.066	0.455	0.260
3	0.010	0.662	0.336
4	0.116	0.647	0.381
5	0.098	0.552	0.325
6	0.218	0.593	0.406
7	0.059	0.385	0.222
8	0.010	0.644	0.327
9	0.239	0.195	0.217
10	0.000	0.393	0.197
11	0.000	0.556	0.278
12	0.150	0.339	0.245
13	0.116	0.329	0.223
14	0.082	0.181	0.131
15	0.191	0.007	0.099
16	0.017	0.739	0.378
17	0.455	0.849	0.652
18	0.146	0.397	0.272

Table 21.5 Average utility value for each level of parameters

Parameter	Utility value			Max–Min (range)
	Level 1	Level 2	Level 3	
A	0.289	0.275	–	0.015
B	0.241	0.261	0.344	0.103
C	0.255	0.329	0.262	0.074
D	0.293	0.258	0.296	0.038
E	0.240	0.282	0.325	0.085
F	0.330	0.243	0.325	0.087
G	0.238	0.306	0.303	0.068

crossover, type of social crossover, type of cognition crossover, swarm size, probability of cognition crossover and type of mutation. The different levels of parameters are plotted with the average utility index, and the parameter with the highest utility value is selected as the best parameter. The utility index value for each parameter at different levels is shown in Fig. 21.1. The parameter values with the highest utility values are *A*1 *B*3 *C*2 *D*3 *E*3 *F*1 *G*2.

The best set of parameters of the proposed DPSO is obtained from Taguchi's method, and the concept of utility value is shown in Table 21.6.

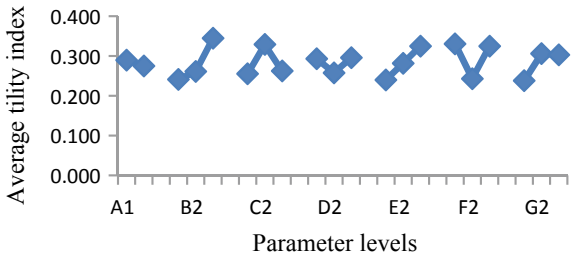

Fig. 21.1 Utility index value for each parameter at various levels

Table 21.6 Best set of parameters of DPSO

Sl. No.	Code	Parameter	The best setting level
1	*A*	Mutation type	Shift
2	*B*	Mutation probability	0.3
3	*C*	Type of cognition crossover	Two point
4	*D*	Probability of cognition crossover	0.9
5	*E*	Type of social crossover	PMX
6	*F*	Probability of social crossover	0.7
7	*G*	Swarm size	30

21.5 Experimentation

Experimentation of the proposed DPSO is carried out on the SDST benchmark problems of flow shop scheduling. The study is conducted on 20 jobs, 50 jobs and 100 jobs with the machine size as 5, 10 and 20. The method for generating the setup times and the due dates required for the study are provided in the following subsection. The algorithms are applied with the best set of parameters determined using Taguchi's orthogonal array and utility index as described in the preceding section. The DPSO metaheuristic terminates after examining 1000 solutions for 20 jobs and 50 jobs, whereas the termination occurs after examining 1500 solutions for 100 jobs. All the problem instances are solved using MATLAB software on a desktop computer that runs on an Intel Core Processor with 3 GHz RAM.

21.5.1 Data Generation for the Problems for Computational Studies

In the present study, the setup times of jobs are considered explicitly. Hence, the setup times of jobs are generated using the setup time level concept. The setup time level is expressed as the ratio of maximum setup time to the maximum processing time. The setup time for the jobs is expressed using the following relation.

$$\text{Setup time level} = \frac{\max s_{ijk}}{\max p_{ijk}} \times 100$$
$$\text{for all } i = 1, 2, \ldots, m; \quad j = 1, 2, \ldots, n; \quad k = 1, 2, \ldots, n, \tag{21.13}$$

where p_{ijk} is the maximum time element of the processing time matrix and s_{ijk} is the maximum time element of the setup time matrix. The setup time level is assumed to be 25%, and hence, the setup time of jobs is generated in a uniform distribution in the interval between 1 and 25. The processing times are obtained from the benchmark problems of flow shop scheduling [17]. The due dates of jobs required for the study are generated using the method of total work content. The due date of a job is expressed as follows.

$$\text{Due date of a job} = \text{arrival time} + r \times (\text{processing time of the job} + \text{setup time of the job}), \tag{21.14}$$

where r is the allowance factor and it is set equal to 2.

$$\text{Setup time of a job} = \text{number of operations of the job} \times \text{average setup time of an operation} \tag{21.15}$$

21.6 Results and Discussion

The Pareto-optimal solutions are determined for the nine problem sizes using the DPSO metaheuristic. The efficacy of the proposed metaheuristic is compared with the hybrid genetic algorithm based on the performance measures such as mean ideal distance (MID), computational time, diversification matrix (DM), average objective values and minimum objective values. The obtained Pareto-optimal solutions for the nine SDST benchmark problem sizes are shown in Table 21.7. It is observed from the values in Table 21.7 that for most of the problem sizes, the values of makespan and mean tardiness obtained from DPSO are better than the hybrid genetic algorithm. In the 20 job × 20 machine problem, both the metaheuristics provide mean tardiness values as zero for one of the solutions in the Pareto-optimal set. However, a better makespan value is obtained from DPSO for the zero mean tardiness value.

21.6.1 Performance Analysis of the Proposed Metaheuristic Based on the Mean Ideal Distance, Computational Time and the Diversification Matrix

The MID values, computational time and DM values of the proposed metaheuristic are shown in Table 21.8. From Table 21.8, it is observed that for all the problem

Table 21.7 MID, computational time and the diversification matrix for the SDST benchmark problems

Sl. No.	Problem size $n \times m$	Mean ideal distance		Computational time (s)		Diversification matrix	
		Hybrid genetic algorithm	DPSO	Hybrid genetic algorithm	DPSO	Hybrid genetic algorithm	DPSO
1	20 × 5	1612.24	1634.97	3.12	**0.83**	22.74	**95.13**
2	20 × 10	2059.68	**1891.51**	3.27	**0.81**	47.58	41.23
3	20 × 20	2834.00	**2426.01**	3.61	**1.46**	174.03	**408.08**
4	50 × 5	3747.80	**3695.62**	6.37	14.97	61.94	**198.46**
5	50 × 10	4160.81	**3735.86**	6.96	15.85	58.33	**105.89**
6	50 × 20	4917.66	**3940.43**	13.53	15.86	79.96	**165.13**
7	100 × 5	7421.30	**7398.17**	24.13	**1.93**	25.00	**55.25**
8	100 × 10	7460.86	**7136.28**	24.15	**1.77**	61.19	**93.02**
9	100 × 20	8863.25	**7654.33**	35.75	**2.08**	48.71	**48.96**

n—Number of jobs; *m*—number of machines
The values are provided in bold to show the better performance of the discrete particle swarm optimisation

Table 21.8 Minimum and average makespan values for the SDST benchmark problems

Problem size $n \times m$	Hybrid genetic algorithm		DPSO	
	Minimum	Average	Minimum	Average
20 × 5	1581	**1589.33**	**1563**	1609.50
20 × 10	2042	2053.33	**1871**	**1890.00**
20 × 20	2740	2834.00	**2222**	**2426.00**
50 × 5	3534	3552.20	**3432**	**3508.50**
50 × 10	4016	4027.25	**3619**	**3664.00**
50 × 20	4839	4869.25	**3881**	**3929.50**
100 × 5	6893	6895.50	**6852**	**6879.00**
100 × 10	7049	7067.67	**6745**	**6791.50**
100 × 20	8527	8551.00	**7431**	**7454.50**

n—Number of jobs; *m*—number of machines
The values are provided in bold to show the better performance of the discrete particle swarm optimisation

sizes except the 20 job × 5 machine problem instance, the MID values are lower for the DPSO metaheuristic. Lower MID values indicate better performance of the metaheuristic. Hence, it is evident from the MID values that the DPSO metaheuristic performs better than the hybrid genetic algorithm. When the computational time is considered, the time taken for the DPSO metaheuristic to provide the Pareto-optimal solutions is lower for most of the problem sizes when compared to the hybrid genetic algorithm. The DM values also reveal the superior performance of DPSO metaheuristic to hybrid genetic algorithm. The higher values of DM indicate the better performance of the metaheuristic. Thus, in terms of MID values, computational time and DM values, the DPSO metaheuristic outperforms the hybrid genetic algorithm.

21.6.2 Performance Analysis Based on the Average and Minimum Objective Function Values

The average values and the minimum values of makespan and mean tardiness obtained for the proposed algorithms are shown in Tables 21.9 and 21.10, respectively. From Table 21.10, it is observed that the minimum value of makespan is obtained from the DPSO metaheuristic for all the problem instances. Further, the average makespan provided by the DPSO metaheuristic has lower values for all the problem sizes except the 20 job × 5 machine problem. In that problem instance, though the average makespan value is better for HGA, the minimum makespan is provided by the DPSO metaheuristic. Similarly, the minimum and average values of mean tardiness have better values for the DPSO metaheuristic except the 20 job ×

Table 21.9 Minimum and average mean tardiness values for the SDST benchmark problems

Problem size $n \times m$	Hybrid genetic algorithm		DPSO	
	Minimum	Average	Minimum	Average
20 × 5	**265.20**	**270.73**	276.90	286.90
20 × 10	148.30	160.57	**67.10**	**75.10**
20 × 20	0.00	1.47	**0.00**	**4.15**
50 × 5	1175.50	1194.78	**1100.70**	**1158.90**
50 × 10	1021.80	1045.55	**700.40**	**728.30**
50 × 20	673.10	688.14	**227.20**	**290.48**
100 × 5	2731.40	2743.65	**2716.70**	**2722.55**
100 × 10	2372.50	2390.02	**2190.40**	**2191.30**
100 × 20	2327.70	2331.85	**1730.70**	**1737.55**

n—Number of jobs; *m*—number of machines
The values are provided in bold to show the better performance of the discrete particle swarm optimisation

Table 21.10 Pareto-optimal solutions for the SDST benchmark problems

Sl. No.	Problem size $n \times m$	Hybrid genetic algorithm		DPSO	
		Makespan	Mean tardiness	Makespan	Mean tardiness
1	20 × 5	1599	265.20	1563	296.9
		1588	267.90	1656	276.9
		1581	279.10		
2	20 × 10	2073	148.30	1909	67.1
		2045	149.00	1871	83.1
		2042	184.40		
3	20 × 20	2914	0.00	2630	0
		2848	1.40	2222	8.3
		2740	3.00		
4	50 × 5	3569	1175.50	3590	1100.7
		3557	1181.30	3557	1113.7
		3554	1193.30	3432	1220.8
		3547	1197.20	3455	1200.4
		3534	1226.60		
5	50 × 10	4041	1021.80	3709	700.4
		4030	1035.10	3619	756.2

(continued)

Table 21.10 (continued)

Sl. No.	Problem size $n \times m$	Hybrid genetic algorithm		DPSO	
		Makespan	Mean tardiness	Makespan	Mean tardiness
		4022	1050.80		
		4016	1074.50		
6	50 × 20	4901	673.10	3881	323.7
		4888	677.00	4015	227.2
		4884	679.80	3902	310
		4880	682.00	3920	301
		4876	683.20		
		4875	683.50		
		4870	685.40		
		4867	689.40		
		4854	691.30		
		4849	694.60		
		4848	694.80		
		4839	723.60		
7	100 × 5	6898	2731.40	6852	2716.7
		6893	2755.90	6906	2728.4
8	100 × 10	7091	2372.50	6745	2192.2
		7075	2374.30	6838	2190.4
		7070	2382.30		
		7065	2394.00		
		7056	2400.00		
		7049	2417.00		
9	100 × 20	8575	2327.70	7431	1744.4
		8527	2336.00	7478	1730.7

n—Number of jobs; *m*—number of machines

5 machine case. Hence, it is evident from the results that the DPSO metaheuristic performs better when compared to the hybrid genetic algorithm.

21.7 Conclusions

The present study proposes a DPSO for solving the bi-objective scheduling problem of an SDST flow shop. Computational studies using the SDST benchmark problems reveal that on an average, the proposed DPSO provides an improvement of 7.8, 22.3

and 11.3% when compared with HGA for the measures such as MID values, computational time and diversification matrix, respectively. In the present study, continuous availability of the machines is assumed. However, in real-life situations, we may encounter breakdown and repair of machines. Thus, the work can be extended by integrating appropriate scheduling and maintenance policies. Other methods for generating the initial population and due dates can be examined. Performance measures other than makespan and mean tardiness can also be considered.

Acknowledgements The authors express their sincere thanks to the reviewers for their suggestions which helped in improving the initial version of the paper.

References

1. Ruiz, R., Maroto, C., Alcaraz, J.: Solving the flow shop scheduling problem with sequence dependent setup times using advanced metaheuristic. Eur. J. Oper. Res. **165**(1), 34–54 (2005)
2. Ciavotta, M., Minella, G., Ruiz, R.: Multi-objective sequence dependent setup times permutation flow shop: a new algorithm and comprehensive study. Eur. J. Oper. Res. **227**(2), 301–313 (2013)
3. Vanchipura, R., Sridharan, R.: Development and analysis of constructive heuristic algorithms for flow shop scheduling problems with sequence dependent setup times. Int. J. Adv. Manuf. Technol. **67**(5), 1337–1353 (2013)
4. Roger, Z., Mercado, R., Bard, J.: Computational experience with a branch and cut algorithm for flow shop scheduling with setups. Comput. Oper. Res. **25**(5), 351–366 (1998)
5. Roger, Z., Mercado, R., Bard, J.: A branch and bound algorithm for permutation flow shops with sequence dependent setup times. IIE Trans. **31**, 721–731 (1999)
6. Deb, K.: Multi-objective Optimisation Using Evolutionary Algorithms, Student ed. Wiley, Hoboken (2005)
7. Rajendran, C., Ziegler, H.: Scheduling to minimise the sum of weighted flow time and weighted tardiness of jobs in a flow shop with sequence dependent setup time. Eur. J. Oper. Res. **149**(3), 513–522 (2003)
8. Eren, T., Guner, E.: A bi-criteria scheduling with sequence dependent setup time. Appl. Math. Comput. **179**(1), 378–385 (2006)
9. Eren, T.: A multi-criteria flow shop scheduling problem with setup times. J. Mater. Process. Technol. **186**(1–3), 60–65 (2007)
10. Dhingra, A., Chandna, P.: A bi-criteria *m*-machine sequence dependent setup time flow shop using modified heuristic genetic algorithm. Int. J. Eng. Sci. Technol. **2**(5), 216–225 (2010)
11. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948. Piscataway, NJ (1995)
12. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. IEEE 4104–4108
13. Pan, Q., Tasgetiren, F., Liang, Y.C.: A discrete particle swarm optimisation algorithm for no-wait flow shop scheduling problem. Eur. J. Oper. Res. **35**(9), 2807–2839 (2008)
14. Enscore Jr., E., Ham, I., Nawaz, M.: A heuristic algorithm for the *m*-machine, *n*-job flow shop sequencing problem. Omega Int. J. Manage. Sci. **11**(1), 91–97 (1983)
15. Ross, P.J.: Taguchi Techniques for Quality Engineering, 2nd ed. McGraw Hill International Editions (1996)

16. Kaladhar, M., Subbaiah, K.V., Rao, S., Rao, K.N.: Application of Taguchi approach and utility concept in solving the multi-objective problem when turning AISI 202 Austenitic Stainless Steel. J. Eng. Sci. Technol. Rev. **4**(1), 55–61 (2011)
17. Taillard, E.: Benchmarks for basic scheduling problems. Eur. J. Oper. Res. **64**(2), 278–285 (1993)