Industrial Engineering
& Management Systems
Vol 12, No 2, June 2013, pp.151-160
ISSN 1598-7248 │ EISSN 2234-6473 │

http://dx.doi.org/10.7232/iems.2013.12.2.151
© 2013 KIIE

An Efficient PSO Algorithm for Finding Pareto-Frontier in Multi-Objective Job Shop Scheduling Problems

Warisa Wisittipanich*
Industrial Engineering, Faculty of Engineering, Chiang Mai University, Chiang Mai, Thailand

Voratas Kachitvichyanukul
Industrial and Manufacturing Engineering, School of Engineering and Technology,
Asian Institute of Technology, Pathumthani, Thailand

(Received: August 30, 2012 / Revised: March 22, 2013 / Accepted May 3, 2013)

ABSTRACT

In the past decades, several algorithms based on evolutionary approaches have been proposed for solving job shop scheduling problems (JSP), which is well-known as one of the most difficult combinatorial optimization problems. Most of them have concentrated on finding optimal solutions of a single objective, i.e., makespan, or total weighted tardiness. However, real-world scheduling problems generally involve multiple objectives which must be considered simultaneously. This paper proposes an efficient particle swarm optimization based approach to find a Pareto front for multi-objective JSP. The objective is to simultaneously minimize makespan and total tardiness of jobs. The proposed algorithm employs an Elite group to store the updated non-dominated solutions found by the whole swarm and utilizes those solutions as the guidance for particle movement. A single swarm with a mixture of four groups of particles with different movement strategies is adopted to search for Pareto solutions. The performance of the proposed method is evaluated on a set of benchmark problems and compared with the results from the existing algorithms. The experimental results demonstrate that the proposed algorithm is capable of providing a set of diverse and high-quality non-dominated solutions.

Keywords: Particle Swarm Optimization, Pareto Front, Multi-Objective Optimization, Job Shop Scheduling Problems

* Corresponding Author, E-mail: warisa_o@hotmail.com

1. INTRODUCTION

The job shop scheduling problem (JSP) is well-known as one of the hardest combinatorial optimization problems, and it has been the subject of many research efforts throughout several decades. The classical JSP allocates a set of jobs on a set of machines in order to optimize one or more objectives, subjected to the constraints that each job has a pre-defined processing operations sequence through all machines. Garey *et al.* (1976) have proven that JSP is NP-hard. The traditional exact algorithms such as mathematical programming and branch and bound methods are ineffective for practitioners since they can only solve small problems and computational time grows exponentially when the problem size increases. Therefore, metaheuristic approaches are more preferable in practice to deal with such high computational complexity of JSP in order to find high-quality or near-optimal solutions in a reasonable time.

Many researchers have studied and proposed algorithms based on metaheuristics to solve single objective JSP. These approaches include, but not limited to, tabu search (Nowicki and Smutnicky, 1996), simulated annealing (Matsuo *et al.*, 1988; Van Laarhoven *et al.*, 1992), genetic algorithm (Goncalves *et al.*, 2005; Yamada and Nakano, 1995), ant colony optimization (Udomsakdigool and Kachitvichyanukul, 2006), particle swarm optimization (PSO) (Pongchairerks and Kachitvichyanukul, 2009a; Rookkapibal and Kachitvichyanukul, 2006), and differential evolution (Liu *et al.*, 2009; Wisittipanich and Kachitvichyanukul, 2012). The number of research works on multi-objective JSP is much less than that for single objective JSP. Generally, the methods for solving multi-objective problems can be classified into two types: non-Pareto-based and Pareto-based approaches. Non-Pareto methods are mostly based on an aggregated weighted approach where multiple objectives functions are given pre-determined weights and combined into a single objective function. However, these methods have two major drawbacks. First, the decision makers are required to determine the suitable weight of each objective which is mainly based on their subjective judgment. As a result, the requirement of prior preference of the decision makers may not lead to a satisfactory result. Second, the decision maker's knowledge about the range of each objective value may be limited. Thus, even with a preference in mind, the single solution obtained provides no possibility for decision tradeoffs. In order to be less subjective, the approach based on a single aggregated objective function needs to be run several times to find a set of solutions based on varying weights. On the other hand, Pareto-based approaches offer an advantage for decision makers to simultaneously find the tradeoff by identifying the non-dominated solutions or Pareto front in a single run.

For over two decades, evolutionary algorithms (EAs) have been mostly used as solution techniques to search for the Pareto front. Nevertheless, only a few research works based on Pareto-based approaches have been proposed for solving multi-objective JSP (MOJSP). Zitzler and Thiele (1999) proposed strength Pareto evolutionary algorithm (SPEA) to emphasize the importance of elitism which has strongly influenced the design of subsequent developed algorithms. Lei and Wu (2006) developed a crowding measure-based multi-objective evolutionary algorithm (CMOEA) for MOJSP. The proposed algorithm makes use of crowding measure to adjust the external population and assign different fitness for individuals in order to simultaneously minimize makespan and the total tardiness of jobs. Ripon *et al.*, 2006 presented a jumping genes genetic algorithm (JGGA) for solving MOJSP. The jumping operations proposed in JGGA exploit scheduling solutions around the chromosomes whereas the general genetic operators globally explore solution from the population. The experimental results demonstrated that the JGGA is capable of searching a set of well diverse solution near the Pareto-optimal front while maintaining consistency and convergence of non-dominated solutions. Chiang and Fu (2006) proposed a genetic algorithm with cyclic fitness assignment (CFGA) which effectively combines three existing fitness assignment mechanisms: MOGA (Horn *et al.*, 1994), SPEA2 (Zitzler *et al.*, 2001), and NSGA-II (Deb *et al.*, 2002) to obtain better performance and help to avoid rapid loss of population diversity. The numerical results showed that the CFGA outperforms MOGA, SPEA2, and NSGA-II. Lei and Xiong (2007) presented an EA for multi-objective stochastic JSP which archive maintenance and fitness assignment were performed based on crowding distance.

PSO proposed by Kennedy and Eberhart (1995), is one of the latest EAs for continuous optimization. PSO has been successfully applied in many fields due to its ease of implementation and computational efficiency. It also demonstrated its efficiency in solving single objective JSP (Pongchairerks and Kachitvichyanukul, 2009a; Pratchayaborirak and Kachitvichyanukul, 2011; Rookkapibal and Kachitvichyanukul, 2006). However, the applications of PSO for solving MOJSP are still very limited. Recently, Lei (2008) designed a Pareto archive particle swarm optimization (PAPSO) to simultaneously minimize makespan and total tardiness of jobs. PAPSO combines the selection of global best position with the crowding measure-based archive maintenance. The proposed algorithm is capable of producing a number of high-quality Pareto optimal scheduling plans.

This paper presents an implementation of an effective multi-objective PSO-based approach (MOPSO), proposed by Nguyen and Kachitvichyanukul (2010). MOPSO has been successfully applied to solving many continuous optimization problems. In this study, the MOPSO is first applied to deal with MOJSP by using the particle representation based on that given in Pratchayaborirak and Kachitvichyanukul (2011). The objective is to simultaneously minimize makespan and total tardiness of jobs.

The remainder of this paper is organized as follows. The job shop problem description is provided in Section 2. Section 3 describes the PSO algorithm and its application to JSP. Section 4 presents the framework of MOPSO algorithm and its movement strategies. Experimental results are reported in Section 5. Finally, our conclusion is provided in Section 6.

2. PROBLEM DESCRIPTION

The classical JSP schedules a set of n jobs on m different machines, subjected to two main sets of constraints; the precedence and conflict constraints. Each job has a set of sequential operations, and each operation must be processed on a specified machine with deterministic processing time known in advance. Each machine is independent from one another. Machine setup time and transfer time between operations are negligible. In addition, machine breakdown and pre-emption are not considered in this problem. The precedence constraints ensure that each operation of a certain job is processed sequentially, and the conflict constraints guarantee that each machine can process only one operation at a time. The goal is to sequence all operations on machines and determine the starting time and the ending time of each operation in order to optimize certain objectives while satisfying all constraints.

In this paper, the objective is to simultaneously minimize two objectives; makespan and total tardiness. The variables and notations used in the JSP model are provided as follows.

$p_{j,k}$	: the process time of job j on machine k
$s_{j,k}$	: the start time of job j on machine k
M	: a large positive number
r_j	: the ready time of job j
D_j	: the due date of job j
$y_{j,j'k}$	: a binary variable defined as

$$y_{j,j'k} = \begin{cases} 1, \text{ if job } j \text{ is before job } j' \text{ on machine } k \\ 0, \text{ otherwise} \end{cases}$$

The mathematical model of the problem can be formulated as follows.

Minimization of makespan:

$$f_1 : minimize\ max\{s_{j,k} + p_{j,k}\} \quad (1)$$

Minimization of total tardiness:

$$f_2 : minimize\ \Sigma\ max\ \{0, (s_{j,k} + p_{j,k}) - D_j\} \quad (2)$$

Subjected to

$$s_{j,k} + p_{j,k} \le s_{j,k'} \qquad \forall j, k, k' \quad (3)$$

$$s_{j,k} + p_{j,k} \le s_{j,k'} + M \cdot (1 - y_{j,j',k}) \qquad \forall j, j', k \quad (4)$$

$$s_{j,k} + p_{j,k} \le s_{j,k'} + M \cdot y_{j,j',k} \qquad \forall j, j', k \quad (5)$$

$$s_{j,k} \ge r_j \ge 0 \qquad \forall j, k \quad (6)$$

where $j, j' = \{1, 2, \cdots, n\}$ and $k, k' = \{1, 2, \cdots, m\}$

Eqs. (1) and (2) are the objective functions. Eq. (3) is the precedence constraint ensuring that each operation of a job is processed sequentially. Eqs. (4) and (5) are the conflict constraints ensuring that each machine can process only one operation at a time. Eq. (6) ensures that no job starts before its ready time.

3. PARTICLE SWARM OPTIMIZAIOTN

3.1 Standard PSO

PSO is a population-based random search approach that imitates the behavior of bird flocking or fish schooling. The original version of PSO was proposed by Kennedy and Eberhart in 1995. In this model, each individual particle in a swarm learns and adapts its search behavior based on its own experience (cognitive term) and global experience (social term) during the search.

The standard PSO is formulated as the following equations.

$$\omega_{id}(t+1) = \omega_{id}(t) + c_p u\left(\psi_{id}^p - q_{id}(t)\right) + c_g u\left(\psi_{id}^g - q_{id}(t)\right) \quad (7)$$

$$q_{id}(t+1) = q_{id}(t) + \omega_{id}(t) \quad (8)$$

where

q_{id}	: current position of d^{th} dimension of i^{th} particle
ω_{id}	: velocity of d^{th} dimension of i^{th} particle
ψ_{id}^p	: personal best position of d^{th} dimension of i^{th} particle
ψ_{id}^g	: global best position of d^{th} dimension of i^{th} particle
c_p	: weight of personal best position term
c_g	: weight of global best position term
u	: uniform random number in range [0, 1]

3.2 GLNPSO

One major drawback of the standard PSO is that the swarm tends to converge prematurely. A swarm is frequently trapped at a local optimum and can no longer move. Several approaches have been proposed to deal with this problem. One of the approaches, introduced by Pongchairerks and Kachitvichyanukul (2009b), utilizes multiple social learning terms and extends the concept of the standard PSO in GLNPSO. Instead of using the global best particle only as reference, GLNPSO also incorporates the local best and near-neighbor best (Veeramachaneni *et al.,* 2003) as additional social learning reference terms. In GLNPSO, velocity (ω_{id}) and position (q_{id}) in the d^{th} dimension of particle i are updated by the following formulas.

$$\omega id(t+1) = w\omega_{id}(t) + c_p u\left(\psi_{id}^p - q_{id}(t)\right) + c_g u\left(\psi_{id}^g - q_{id}(t)\right) \quad (9)$$
$$+ c_l u\left(\psi_{id}^l - q_{id}(t)\right) + c_n u\left(\psi_{id}^n - q_{id}(t)\right)$$

$$q_{id}(t+1) = q_{id}(t) + \omega_{id}(t) \quad (10)$$

The inertia weight, w, introduced by Shi and Eberhart (1998), is used to improve the search ability. The value of w is linearly decreasing so that the swarm can search the whole space aggressively at the early stage and gradually reduce the search space at the later stage. ψ_{id}^p, ψ_{id}^q, ψ_{id}^l, ψ_{id}^n, represent the d^{th} dimension of the personal best, the global best, the local best, the near neighbor best, respectively, and $q_{id}(t)$ is the current position of particle i at t^{th} generation. The local best position of particle i, $\psi_i^l = \{\psi_{i0}^l, \psi_{i1}^l, \cdots, \psi_{iD}^l\}$ can be determined as the best particle among K neighboring particles. It is equivalent to dividing the whole swarm into multiple sub-swarms with population of size K, the local best particle is the best particle among the K neighbors, i.e., the particle gives the best fitness value in the sub-swarms.

The near neighbor best position of particle i, $\psi_i^n = \left\{\psi_{i0}^n, \psi_{i1}^n, \cdots, \psi_{iD}^n\right\}$ represents the interaction between particles to achieve a better value of the objective function. The neighborhood is determined by the direct linear distance from the best particle. Each element of ψ_i^n is determined by the following formula:

$$FDR_{id} = \frac{Fitness(\Theta_i) - Fitness(\Psi_i)}{q_{id} - \psi_{id}} \qquad (11)$$

The equations for updating position and inertia weight w in GLNPSO are the same as standard PSO formulas. The other parameters and working procedures are also similar to those in the standard PSO.

3.3 Solution Representation

Since PSO is first designed for continuous domain, in order to apply PSO to combinatorial problem, i.e., scheduling problem, particles need to be transformed into a schedule. This study implements the solution mapping procedure presented in the paper work of Pratchayaborirak and Kachitvichyanukul (2011). These procedures use the random key representation encoding scheme (Bean, 1994), and permutation of job-repetition (Bierwirth, 1995) to generate the sequence of all operations. Then, the operation-based approach (Cheng *et al.*, 1996) is employed to generate an active schedule.

A solution of a job shop scheduling problem is represented using a particle with dimensions equal to the total number of operations processed on all machines. Consider an example of three jobs and three machines in JSP in Table 1.

According to this example, the total number of dimensions, equal to the total number of operations, is 9. Fig. 1 illustrates the encoding scheme for the random key representation where each value in a dimension is initially generated with a uniform random number in range [0, 1].

Table 1. An example of job shop scheduling problem with 3 jobs and 3 machines

Job	Machine sequence			Processing time		
A	M1	M3	M2	2	1	4
B	M2	M1	M3	3	6	5
C	M3	M2	M1	7	4	3

Next, the permutation of 3-repetition of 3 jobs is applied with a sorting list rule to decode an individual vector into a sequence of operations as shown in Figure 2. The advantage of this approach is that any permutation of this representation always provides a feasible schedule. Then, the operation-based approach is employed to generate a schedule. The decoded individual is transformed into a schedule by taking the first operation from the list, then the second operation, and so on. During the process of generating a schedule, each operation is allocated to a required machine in the best available position without delaying other scheduled operations. This procedure results in an active schedule as shown in Figure 3.

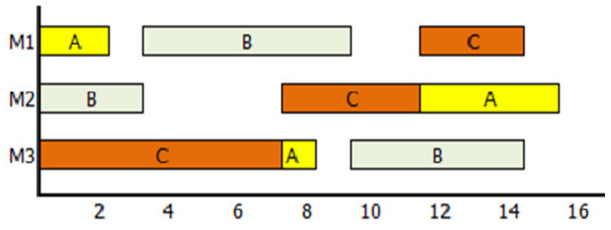

Figure 3. An active schedule after the decoding process.

Dimension j	1	2	3	4	5	6	7	8	9
	0.78	0.38	0.14	0.46	0.98	0.72	0.25	0.66	0.87

Figure 1. Random key representation.

Dimension j	3	7	2	4	8	6	1	9	5
	0.14	0.25	0.38	0.46	0.66	0.72	0.78	0.87	0.98
	A	A	A	B	B	B	C	C	C

Dimension j	1	2	3	4	5	6	7	8	9
	0.78	0.38	0.14	0.46	0.98	0.72	0.25	0.66	0.87
	C	A	A	B	C	B	A	B	C

Figure 2. m-repetition of job number permutation and operation-based representation.

4. MOPSO ALGORITHM

4.1 MOPSO Framework

As mentioned earlier, this study employs an MOPSO algorithm proposed by Nguyen and Kachitvichyanukul (2010) to solve multi-objective JSP. In MOPSO framework, the flying experience of a swarm is stored in an external archive, called Elite group, as a set of non-dominated solutions. After each movement, each particle position is updated, and the objective functions that correspond to the position of each particle are re-evaluated. Then, the dominated sorting procedure is activated to identify the group of new non-dominated solutions. This sorting procedure applies to the group of new solutions and current solutions in the external archive and store only non-dominated solutions into an archive for the Elite group. Then, solutions in the Elite group are screened to eliminate inferior solutions. As a result, the Elite group in the archive contains the best non-dominated solutions found so far in the searching process of the swarm.

It is important to mention that the movement of particles is a very critical attribute of the algorithm to enhance the quality of the Pareto front. In multi-objective problems, the existence of multiple candidates in the archive provides a large number of reference options for the movement of particles. Several potential movement strategies are proposed in MOPSO framework to utilize the non-dominated solutions in Elite group as the movement guidance. This study adopts Ms*, one of MOPSO movement strategies, as the movement guidance of particles since the Ms* has demonstrated its effectiveness by providing high quality solutions in many standard tested problems. This strategy is discussed in the next section.

4.2 Movement Strategy

The movement strategy (Ms*) uses a single swarm with mixture of particles with different movement behaviors. In Ms*, four groups of particles (Ms1, Ms2, Ms3, and Ms4) co-exist in the same swarm, and all of their flying experiences are stored in a common Elite archive. In the first group, Ms1, a particle will not directly use the global knowledge but will explore the search space gradually based on its own experience. As a result, these particles do not change their movement abruptly every time the global trend changes. This feature helps them to better explore the local region. The formula for updating the position of a particle in Ms1 is shown in Eq. (12).

Ms1:
$$w\omega_{i,d} + c_p u(\psi_{id}^{p} - q_{i,d}) + c_l u(\psi_{id}^{l} - q_{i,d}) + c_n u(\psi_{id}^{n} - q_{i,d}) \quad (12)$$

On the other hand, the remaining groups of particles utilize the global knowledge from the Elite group as the guidance to move to new positions. In the second group, the particles are guided to the less crowded areas of the non-dominated front. Thus, the movement of those particles will mainly depend on how the solutions are distributed in the current Elite group. Although the first two groups have tried to explore the search in many different directions, they may still leave some unexplored gaps because of their convergence at some segments on the Pareto front. For that reason, the task of particles in the third group is to fill these gaps so that the final front can have a better distribution. Finally, the responsibility of particles in the fourth group is to explore around the endpoints of the front to increase the spread of the non-dominated fronts. The formula for updating new position with the global term of a particle in Ms2, Ms3 and Ms4 are show in the following equations.

$$\text{Ms2:} \quad c_g u(y_{R1,d} - q_{i,d}) \quad (13)$$

$$\text{Ms3:} \quad c_g u\left[(E_{1,d} - q_{i,d}) + (E_{1,d} - E_{2,d})\right] \quad (14)$$

$$\text{Ms4:} \quad c_g u(y_{R1,d} - y_{R2,d}) \quad (15)$$

where $y_{R1,d}$ and $y_{R2,d}$ are the position of particle *R1* and *R2* from the top (%) and bottom (%) of Elite group respectively, *r* is uniform random number in range (0, 1), $E_{1,d}$ and $E_{2,d}$ are a pair of particle position selected form unexplored list in Elite group.

5. EEPERIMENTAL RESULTS

The experiments are implemented using the C# language of the Microsoft Visual Studio 9.0. The program runs on the platform of Intel Core 2 Duo CPU 1.67 GHz with 3062 MRAM. The performance of MOPSO is evaluated using fifteen benchmark job shop problems. The due date data for FT06 is set according to Ponnambalam *et al.* (2001). For the problems with 10 and 20 jobs, the due date data are set according to Lei and Wu (2006). The MOPSO parameters used in this experiment are shown in Table 2.

Table 2. Parameters of multi-objective particle swarm optimization based approach

Inertia weight	Linearly decrease from 0.9 to 0.4
Constant acceleration	$c_p, c_g, c_l, c_n = 1$
Swarm size (particles)	1,200
Particles in Elite group	100
Top members (%)	10
Bottom member (%)	20
Potential gap (%)	5
Number of iterations	2,000

The ratio of number of particles in each group is 1:1:1:1, respectively. Ten replications are performed in each tested instance. The movement behavior of the Pareto front found by the particles in the MOPSO algorithm is illustrated in Figure 4 for problem instance FT10 and in Figure 5 for problem instance LA28.

It can be seen from Figures 4 and 5 that as the search progresses, particles in a swarm can gradually find better non-dominated solutions in the less crowded area, on the border of the front, and between the potential gaps; resulting in a well-spread and better quality of non-dominated fronts. It is worth noting that JSP is a combinatorial problem, thus an obtained Pareto front may leave some gaps since the solutions between those gaps might not exist.

The results of the MOPSO algorithm are compared to those obtained from other existing algorithms. Table 3 shows the comparison results on non-dominated solutions obtained from MOPSO with those from SPEA and CMOEA given in Lei and Wu (2006). The notation (x, y) represents (makespan, total tardiness) in Table 3.

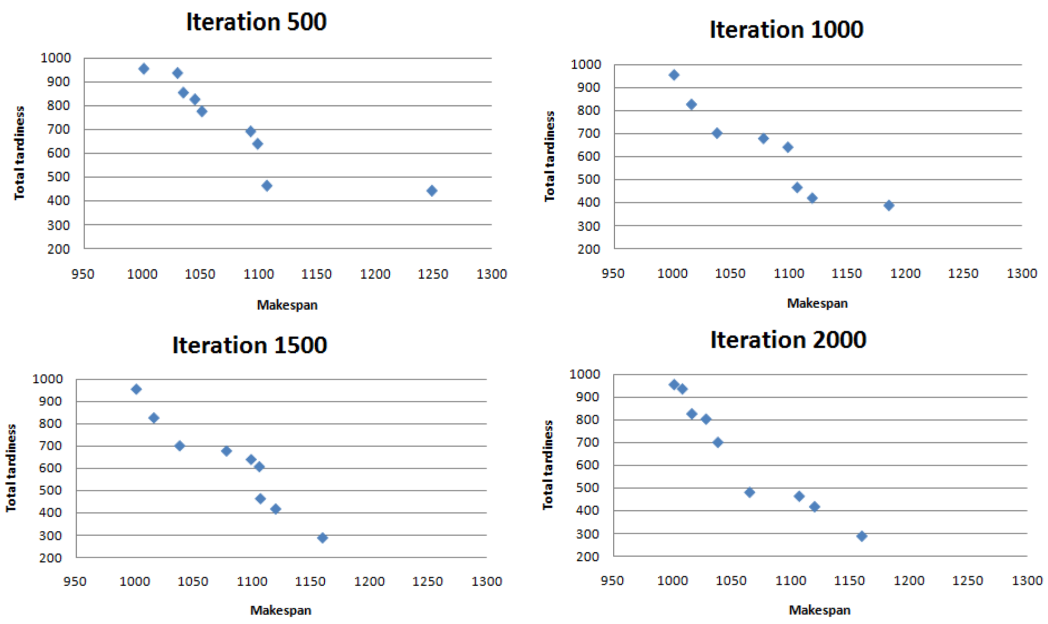

Figure 4. Movement behavior of the Pareto front for instance FT10 (10 jobs 10 machines).

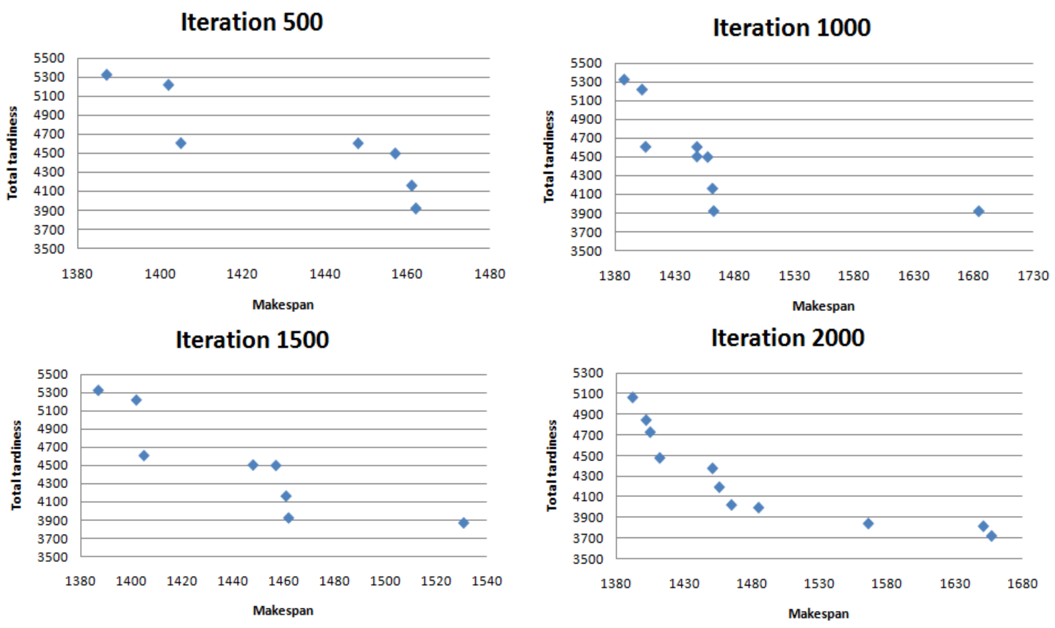

Figure 5. Movement behavior of the Pareto front for instance LA28 (20 jobs 10 machines).

Table 3. The objective function of non-dominated solutions provided by SPEA, CMOEA, and MOPSO

Instance	SPEA	CMOEA	MOPSO
FT06	(58, 4)	(58, 4)	(55, 28)
	(60, 1)	(60, 1)	(56, 27)
	(62, 0)	(62, 0)	(57, 19)
			(58, 4)
			(60, 1)
			(62, 0)
FT10	(1059, 180)	(1057, 274)	(966, 843)
	(1076, 196)	(1072, 187)	(969, 725)
	(1093, 267)	(1085, 156)	(1057, 167)
			(1286, 116)
FT20	(1269, 6629)	(1283, 6858)	(1234, 8502)
	(1276, 6928)	(1287, 6668)	(1246, 8211)
	(1287, 6376)	(1306, 6644)	(1258, 7981)
			(1271, 7776)
			(1304, 7374)
			(1310, 7304)
			(1333, 7089)
			(1454, 6796)
ABZ5	(1306, 439)	(1277, 422)	(1284, 350)
	(1316, 486)	(1296, 360)	(1270, 460.5)
			(1276, 436)
			(1317, 200)
			(1331, 133)
			(1337, 0)
ABZ6	(981, 212)	(979, 348)	(978, 301)
	(994, 216)	(988, 155)	(984, 251)
	(1002, 166)	(993, 309)	(994, 196)
			(1028, 62.5)
			(1054, 43)
			(1084, 30)
			(1088, 1.5)
			(1025, 0)
ABZ7	(790, 487)	(786, 350)	(757, 868)
	(783, 396)	(789, 465)	(761, 761)
	(794, 354)	(792, 353)	(782, 753)
			(786, 747)
			(789, 665)
			(790, 650)
			(802, 617)
			(807, 582)
			(813, 546)
			(841, 514)
ABZ8	(805, 707.5)	(817, 293)	(780, 912.5)
	(808, 624.5)	(819, 552.5)	(781, 911.5)
	(810, 582.5)	(824, 251)	(782, 892.5)
			(791, 829.5)
			(801, 818.5)
			(810, 766.6)
			(817, 668.5)
			(854, 652.5)
			(864, 596.5)
			(886, 584)
ORB1	(1160, 718)	(1160, 858)	(1133, 670.5)
	(1193, 413)	(1161, 393.5)	(1154, 819.5)
	(1191, 469)	(1188, 381.5)	(1159, 796.5)
			(1160, 753)
			(1166, 665)
			(1173, 379)
			(1180, 257)
			(1227, 247.5
			(1290, 268.5)
ORB2	(949, 192)	(941, 36)	(920, 276)
	(952, 190)	(952, 41)	(921, 263)
	(956, 130)	(956, 31)	(931, 173)
			(943, 131)
			(946, 66)
			(952, 32)
			(955, 28)
			(971, 9)
			(974, 0)
ORB3	(1164, 890)	(1167, 367)	(1103, 944.5)
	(1165, 793)	(1173, 364)	(1139, 883)
	(1158, 799)	(1191, 596)	(1144, 638)
			(1153, 591)
			(1155, 411)
			(1231, 361)
			(1268, 288.5)
ORB4	(1130, 709.5)	(1138, 314.5)	(1089, 570.5)
	(1148, 616.5)	(1139, 690.5)	(1110, 392.5)
	(1168, 483.5)	(1154, 569.5)	(1134, 322.5)
			(1150, 312)
			(1180, 304.5)
			(1185, 271.5)
			(1217, 206.5)
			(1223, 180.5)
			(1262, 121.5)
ORB5	(1002, 1)	(988, 23)	(971, 214)
	(1012, 5)	(989, 45)	(979, 190)
		(994, 18)	(981, 80)
			(992, 61)
			(995, 14)
			(1012, 0)
LA26	(1405, 4436)	(1366, 3539)	(1333, 5259)
	(1428, 4346)	(1375, 3537)	(1345, 5245)
		(1394, 4063)	(1346, 4620)
			(1349, 4392)
			(1375, 4323)
			(1389, 4205)
			(1676, 3551)
LA27	(1451, 2960.5)	(1451, 2968.5)	(1386, 4736.5)
	(1452, 2512.5)	(1452, 2611.5)	(1389, 4458.5)
			(1401, 4197.5)
			(1407, 4151.5)
			(1416, 3974.5)
			(1434, 3756.5)
			(1483, 3739.5)
			(1508, 3662.5)
LA28	(1400, 3111)	(1398, 3553)	(1358, 4825)
	(1414, 2926)	(1410, 3339)	(1359, 4785)
			(1362, 4773)
			(1369, 4158)
			(1379, 4088)
			(1386, 4019)
			(1393, 3501)
			(1599, 3311)

SPEA: strength Pareto evolutionary algorithm, CMOEA: crowding measure-based multi-objective evolutionary algorithm, MOPSO: multi-objective particle swarm optimization based approach.

As shown in Table 3, SPEA, CMOEA, and MOPSO perform well in providing a set of non-dominated solutions. The makespan is just about 3% to 20% bigger than the optimal makespan, and some makespan values are very close to the optimal results; while the total tardiness of these solutions is reasonable. The movement strategy used in MOPSO is capable of generating a larger set of non-dominated solutions. In addition, it is clear that the spread of the fronts generated via the MOPSO is much wider than those obtained from the other two methods since the minimum values of make-span and total tardiness found by the MOPSO are lower. More specifically, MOPSO clearly outperforms SPEA and CMOEA for instance FT10, ABZ5, ABZ6, ORB1, ORB2, ORB3, and ORB4 since several solutions obtained by SPEA and CMOEA are dominated by those from MOPSO. However, the solutions obtained from MOPSO are inferior to SPEA and CMOEA in instance FT20, ABZ7, ABZ8, and LA27.

To further evaluate the performance of the MOPSO, this study uses $\tilde{C}$ metric to compare the set of non-dominated solutions obtained from the MOPSO with those obtained from existing algorithms; SPEA, and CMOEA. $\tilde{C}(A, B)$ measures the fractions of members of B that are dominated by members of A.

$$\tilde{C}(A, B) = \frac{\left|\{b \in B : \exists a \in A, a \succ b\}\right|}{|B|}$$

where $|B|$ is the number of solutions in B. Therefore, $\tilde{C}(A, B) = 1$ means that each solution in B is dominated by some solutions in A. On the other hand, $\tilde{C}(A, B) = 0$ represents that all solutions in B are non-dominated by any solution in A. The lower the ratio $\tilde{C}(A, B)$ is, the better the solution set in B is.

Table 4 shows the comparison results between MOPSO, SPEA, and CMOEA. It is noted that *S*, *C*, and *P* are represented for SPEA, CMOEA, and MOPSO algorithm correspondingly.

As can be seen in Table 4, the MOPSO algorithm outperforms SPEA and CMOEA in most cases. MOPSO obtains lower value of $\tilde{C}$ metric than SPEA and CMOEA for nine and eight out of fifteen instances, respectively. It is worth noting that the performance of MOPSO is inferior to SPEA and CMOEA for instances FT20, ABZ7, and ABZ8 since the value of $\tilde{C}$ metric obtained from MOPSO is higher than those obtained from the other two algorithms. However, the sum of fraction of solutions obtained from SPEA and CMOEA is much higher than MOPSO. This indicates that, in general, the solutions obtained from SPEA and CMOEA are mostly dominated by solutions from MOPSO. In most cases, when the $\tilde{C}$ metric values of the MOPSO are lower, the difference value of $\tilde{C}$ between MOPSO and other two algorithms is significant. In contrast, SPEA and CMOEA just slightly outperform MOPSO on only two and three instances, respectively, and the difference on $\tilde{C}$ is trivial.

Table 4. The comparison between SPEA, CMOEA, and MOPSO

Instance	$\tilde{C}(P, S)$	$\tilde{C}(S, P)$	$\tilde{C}(P, C)$	$\tilde{C}(C, P)$
FT06	0	0	0	0
FT10	1	0	0.667	0
FT20	0	0.625	0	0.5
ABZ5	1	0	0.5	0
ABZ6	0.333	0.125	0.667	0.125
ABZ7	0	0.7	0	0.7
ABZ8	0	0.5	0	0.4
ORB1	1	0.111	0.667	0.111
ORB2	1	0	0.667	0.222
ORB3	1	0	0.333	0
ORB4	1	0	0.667	0
ORB5	0.5	0	0	0.167
LA26	1	0	0	0.428
LA27	0	0.25	0	0.25
LA28	0	0.125	0.5	0
Sum of dominated fraction	*7.833*	*2.436*	*4.668*	*2.903*

SPEA: strength Pareto evolutionary algorithm, CMOEA: crowding measure-based multi-objective evolutionary algorithm, MOPSO: multi-objective particle swarm optimization based approach.

The main structural difference between MOPSO, SPEA and CMOEA is that SPEA and CMOEA focus on the use of external population, fitness assignment techniques and preservation of population while MOPSO emphasizes the movement direction of particular particles to fulfill the Pareto front by utilizing multiple learning terms. The successful implementation of MOPSO is attributed to the use of various movement strategies in MOPSO. Each group of particles performs different movements with its own advantage to explore different potential areas. In addition, the use of a common Elite group helps particles to utilize the global information, thus the search procedure is faster and a better quality of solutions is obtained.

6. CONCLUSION

This study presented an application of the MOPSO algorithm for solving MOJSP with the objective to simultaneously minimize makespan and total tardiness of jobs. The MOPSO framework uses Elite group to store solutions and utilizes those solutions as the guidance of particle movement. A single swarm with a mixture of four groups of particles with different movement strategies is adopted to search for Pareto front. While particles in the first group explore solutions based on their own experience, particles in the second group utilize the global knowledge by adopting crowding distance as the guidance of their movement to better explore the less crowded area. The task of particles in the third group is to fill the unexplored gaps between non-dominated solutions in order to obtain a better Pareto distribution. Finally, particles in the fourth group aim to search around the border of the Pareto front to increase the spread of the front. The performance of the MOPSO approach is evaluated on a set of benchmark JSP problems and compared with the results from the other existing evolutionary algorithms (SPEA and CMOEA). The results demonstrate that the MOPSO algorithm is a competitive approach, and it is capable of finding a set of diverse and high quality non-dominated solutions on Pareto front.

REFERENCES

Bean, J. C. (1994), Genetic algorithms and random keys for sequencing and optimization, *INFORMS Journal on Computing*, **6**(2), 154-160.

Bierwirth, C. (1995), A generalized permutation approach to job shop scheduling with genetic algorithms, *Operations-Research-Spektrum*, **17**(2/3), 87-92.

Cheng, R., Gen, M., and Tsujimura, Y. (1996), A tutorial survey of job-shop scheduling problems using genetic algorithms: I. representation, *Computers and Industrial Engineering*, **30**(4), 983-997.

Chiang, T. C. and Fu, L. C. (2006), Multiobjective job shop scheduling using genetic algorithm with cyclic fitness assignment, *Proceedings of the IEEE Congress on Evolutionary Computation*, Vancouver, Canada, 3266-3273.

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

Garey, M. R., Johnson, D. S., and Sethi, R. (1976), The complexity of flowshop and jobshop scheduling, *Mathematics of Operations Research*, **1**(2), 117-129.

Goncalves, J. F., de Magalhaes Mendes, J. J., and Resende, M. G. C. (2005), A hybrid genetic algorithm for the job shop scheduling problem, *European Journal of Operational Research*, **167**(1), 77-95.

Horn, J., Nafpliotis, N., and Goldberg, D. E. (1994), A niched Pareto genetic algorithm for multiobjective optimization, *Proceedings of the 1st IEEE Congress on Evolutionary Computation*, Orlando, FL, 82-87.

Kennedy, J. and Eberhart, R. C. (1995), Particle swarm optimization, *Proceedings of the IEEE International Conference on Neural Networks*, Perth, WA, 1942-1948.

Lei, D. (2008), A Pareto archive particle swarm optimization for multi-objective job shop scheduling, *Computers and Industrial Engineering*, **54**(4), 960-971.

Lei, D. and Wu, Z. (2006), Crowding-measure-based multiobjective evolutionary algorithm for job shop scheduling, *International Journal of Advanced Manufacturing Technology*, **30**(1-2), 112-117.

Lei, D. M. and Xiong, H. J. (2007), An efficient evolutionary algorithm for multi-objective stochastic job shop scheduling, *Proceedings of the 6th International Conference on Machine Learning Cybernetics*, Hong Kong, 867-872.

Liu, F., Qi, Y., Xia, Z., and Hao, H. (2009), Discrete differential evolutionary algorithm for the job shop scheduling problem, *Proceedings of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, Shanghai, China, 879-882.

Matsuo, H., Suh, C. J., and Sullivan, R. S. (1988), A controlled search simulated annealing method for the general job-shop scheduling problem, *Working Paper #03-04-88*, Graduate School of Business, The University of Texas at Austin, Austin, TX.

Nguyen, S. and Kachitvichyanukul, V. (2010), Movement strategies for multi-objective particle swarm optimization, *International Journal of Applied Metaheuristic Computing*, **1**(3), 59-79.

Nowicki, E. and Smutnicki, C. (1996), A fast taboo search algorithm for the job shop problem, *Management Science*, **42**(6), 797-813.

Pongchairerks, P. and Kachitvichyanukul, V. (2009a), A two-level particle swarm optimisation algorithm on job-shop scheduling problems, *International Journal of Operational Research*, **4**(4), 390-411.

Pongchairerks, P. and Kachitvichyanukul, V. (2009b), Particle swarm optimization algorithm with multiple social learning structures, *International Journal of Operational Research*, **6**(2), 176-194.

Ponnambalam, S. G., Ramkumar, V., and Jawahar, N. (2001), A multiobjective genetic algorithm for job shop scheduling, *Production Planning and Control*, **12**(8), 764-774.

Pratchayaborirak, T. and Kachitvichyanukul, V. (2011), A two-stage PSO algorithm for job shop scheduling problem, *International Journal of Management Science and Engineering Management*, **6**(2), 84-93.

Ripon, K. S. N., Tsang, C. H., and Kwong, S. (2006), Multi-objective evolutionary job-Shop scheduling using jumping genes genetic algorithm, *Proceeding of the International Joint Conference on Neural Networks*, Vancouver, Canada, 3100-3107.

Rookkapibal, L. and Kachitvichyanukul, V. (2006), Particle swarm optimization for job shop scheduling problems, *Proceedings of the 36th CIE Conference on Computers and Industrial Engineering*, Taipei.

Shi, Y. and Eberhart, R. (1998), A modified particle swarm optimizer, *Proceedings of the IEEE World Congress on Computational Intelligence Evolutionary Computation*, Anchorage, AK, 69-73.

Udomsakdigool, A. and Kachitvichyanukul, V. (2006), Two-way scheduling approach in ant algorithm for solving job shop problems, *International Journal of*

Industrial Engineering and Management Systems, **5**(2), 68-75.

Van Laarhoven, P. J. M., Aarts, E. H. L., and Lenstra, J. K. (1992), Job shop scheduling by simulated annealing, *Operations Research*, **40**(1), 113-125.

Veeramachaneni, K. , Peram, T., Mohan, C., and Osadciw, L. A. (2003), Optimization using particle swarms with near neighbor interactions, *Proceedings of the International Conference on Genetic and Evolutionary Computation*, Chicago, IL, 110-121.

Wisittipanich, W. and Kachitvichyanukul, V. (2012), Two enhanced differential evolution algorithms for job shop scheduling problems, *International Journal of Production Research*, **50**(10), 2757-2773.

Yamada, T. and Nakano, R. (1995), A genetic algorithm with multi-step crossover for job-shop scheduling problems, *Proceedings of the 1st International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, Sheffield, UK, 146-151.

Zitzler, E. and Thiele, L. (1999), Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, *IEEE Transaction on Evolutionary Computation*, **3**(4), 257-271.

Zitzler, E., Laumanns, M., and Thiele, L. (2001), SPEA2: improving the strength Pareto evolutionary algorithm, *TIK Report 103*, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Zurich, Switzerland.