A New Particle Swarm Optimization for Multi-objective Open Shop Scheduling

D.Y. Sha

Department of Technology Management, Chung Hua University, Hsinchu, Taiwan R.O.C.

yjsha@chu.edu.tw

Hsing-Hung Lin

hsinhung@gmail.com

Abstract—**Multi-objective open-shop scheduling is definitely significant in practical. However, the research focused on multi-objective open-shop scheduling was relatively scarce. This article proposed a particle swarm optimization to address open-shop scheduling problems with multiple objectives. Originally, particle swarm optimization was invented to treat continuous optimization problems. In this paper, the particle position representation, particle velocity and particle movement are modified due to the discrete essence of the scheduling problem. The modified PSO was tested on solving various benchmark problems to evaluate its performance. The results demonstrated that the algorithm dividing the swarm into three sub-swarms for each object performs better than using only one swarm for all three objectives.**

Keywords-Particle Swarmt; Open Shop; Multi-objective

I. INTRODUCTION

Shop scheduling problems including flow, job and open shop problems, are always attracted attention of researchers. From the practical view, shop scheduling has become a significant factor to maintain competitive position for enterprises facing the rapid changing market. Most previous research into the open shop scheduling problem has concentrated on finding a single optimal solution (e.g., makespan). However, in real world, the multi-objectives requirements of shop scheduling should be achieved at the same time. This makes the academic concentration of objectives in open shop scheduling has been extended from single to multiple.

Since the open shop scheduling problem is NP-hard for m>2 [1], we cannot solve it exactly in a reasonable computation time. Most published research concentrated on developing heuristic algorithm for searching the optimal makespan of open-shop scheduling problem. A neighborhood search algorithm based on the simulated annealing technique is proposed to addresses the problem of scheduling a non-preemptive open shop with the objective of minimizing makespan [2]. An efficient local search algorithm based on the tabu search technique is proposed for solving the open shop scheduling problem with the objective of minimizing makespan [3].

The development and application of a hybrid genetic algorithm (HGA) to the open shop scheduling problem was examined. The hybrid algorithm incorporates a local improvement procedure based on tabu search (TS) into a basic genetic algorithm (GA) [4]. Beam-ACO hybridized the solution construction mechanism of ACO with beam search to tackle open-shop scheduling [5]. Several competitive GAs are presented for detecting global optimal disseminated among many quasi-optimal schedules of open-shop problem [6].A heuristic for the open hob shop scheduling problem using genetic algorithm to minimize makespan was developed [7].A heuristic algorithm, Shortest Processing Time Block (SPTB), is presented to deal with open shop problem to minimize the sum of the completion time. Tang and Bai(2010) [8]

Considering the problem of scheduling preemptive open shops to minimize total tardiness, an efficient constructive heuristic is developed for solving large-sized problems. A branch-and-bound algorithm that incorporates a lower bound scheme based on the solution of an assignment problem as well as various dominance rules are presented for solving medium-sized problems [9].

Blazewicz et al. [10] concerns the application of a non-classical performance measure, a late work criterion, to scheduling problems. It estimates the quality of the obtained solution with regard to the duration of the late parts of tasks not taking into account the quality of this delay.

Particle swarm optimization proposed by Kennedy et al. [11] recently for unconstrained continuous optimization problems is one of the latest evolutionary techniques. The major spirit of PSO is based on observations of the social behaviors of animals such as bird flocking, fish schooling, and swarm theory. PSO has been successfully applied to different field of applications due to the easy implementation and computational efficiency. Nevertheless, the applications of the PSO on the combination optimization problem are still scarce.

The aim of this paper is to explore the development of PSO for elaborate multi-objective open shop scheduling problems. The original PSO was used to solve continuous optimization problems. Due to the discrete solution spaces of scheduling optimization problems, we modified the particle position representations, particle movement, and particle velocity in this work.

The remainder of this paper is organized as follows. Section 2 contains a formulation of the open shop scheduling problem with three objectives. Section 3 describes the modified algorithm of the proposed PSO approach. Section 4 contains the simulated results of benchmark problems. Section 5 provides some conclusions and future directions.

II. Open Shop Scheduling

A. Problem Statement

The common characters of shop scheduling problems are as follows. There is a set of ***n*** jobs that have to be processed on a set of ***m*** machines. Each job consists of ***m*** operations, each of which must be processed on a different machine on a given process time. At any time, at most one operation can be processed on each machine and at most one operation of each job can be processed. Unlike flow-shop and job-shop scheduling problems, the exceptional condition of open-shop scheduling problem is that the operations of each job can be processed in any order.

B. Problem Objectives

The task of this study is to assign jobs to machines so that the completion time, also called the makespan, total flow time and machine idle time is minimized simultaneously. Makespan is to minimize the maximum total processing time on any machine. Total flow time means the sum of the completion times of all jobs. The idle time of each machine during the work cycle are totally computed to machine idle time.

III. Particle Swarm Optimization

PSO is based on observations of the social behavior of animals, such as birds in flocks or fish in schools, as well as on swarm theory. The population consisting of individuals or particles is initialized randomly. Each particle is assigned with a randomized velocity according to its own movement experience and that of the rest of the population. The relationship between the swarm and particles in PSO is similar to the relationship between the population and chromosomes in a GA.

In PSO, the problem solution space is formulated as a search space. Each particle position in the search space is a correlated solution to the problem. Particles cooperate to determine the best position (solution) in the search space (solution space).

Suppose that the search space is D-dimensional and there are ρ particles in the swarm. Particle i is located at position $X^i=\{x_1^i, x_2^i, \ldots, x_D^i\}$ and has velocity $V^i=\{v_1^i, v_2^i, \ldots, v_D^i\}$, where i=1, 2, …,ρ. Based on the PSO algorithm, each particle move towards its own best position (*pbest*), denoted as $Pbest^i=\{pbest_1^i, pbest_2^i,\ldots, pbest_n^i\}$, and the best position of the whole swarm (*gbest*) is

denoted as $Gbest=\{gbest_1, gbest_2, \ldots, gbest_n\}$ with each iteration. Each particle changes their position according to its velocity, which is randomly generated toward the *pbest* and *gbest* positions. For each particle r and dimension s, the new velocity v_s^r and position x_s^r of particles can be calculated by the following equations:

$$v_{rs}^{\tau} = w \times v_{rs}^{\tau-1} + c_1 \times rand_1 \times [pbest_{rs}^{\tau-1} - x_{rs}^{\tau-1}] + c_2 \times rand_2 \times [gbest_s^{\tau-1} - x_{rs}^{\tau-1}] \quad (1)$$

$$x_{rs}^{\tau} = x_{rs}^{\tau-1} + v_{rs}^{\tau-1} \quad (2)$$

In Eqs. (1) and (2), τ is the iteration number. The inertial weight w is used to control exploration and exploitation. A large w value keeps the particles moving at high velocity and prevents them from becoming trapped in local optima. A small w value ensures a low particle velocity and encourages particles to exploit the same search area. The constants c_1 and c_2 are acceleration coefficients to determine whether particles prefer to move closer to the *pbest* or *gbest* positions. The $rand_1$ and $rand_2$ are two independent random numbers uniformly distributed between 0 and 1. The termination criterion of the PSO algorithm includes a maximum number of generations, a designated value of *pbest*, and lack of further improvement in *pbest*. The standard PSO process is outlined as follows:

Step 1: Initialize a population of particles with random positions and velocities in a D-dimensional search space.

Step 2: Update the velocity of each particle using Eq. (1).

Step 3: Update the position of each particle using Eq. (2).

Step 4: Map the position of each particle into the solution space and evaluate its fitness value according to the desired optimization fitness function. Simultaneously update the *pbest* and *gbest* positions if necessary.

Step 5: Loop to Step 2 until the termination criterion is met, usually after a sufficient good fitness or a maximum number of iterations.

The original PSO was designed for a continuous solution space. We must modify the PSO position representation, particle velocity, and particle movement so they work better with combinational optimization problems. These changes are described in next section.

IV. METHODS

There are four types of feasible schedules in OSSPs, including inadmissible, semi-active, active, and non-delay. The optimal schedule is guaranteed to be an active schedule. We can decode a particle position into an active schedule employing Giffler and Thompson's [12] heuristic. There are two different representations of particle position associated with a schedule. The results of

Zhang *et al.*[13] demonstrated that permutation-based position representation outperforms priority-based representation. While choosing to implement permutation-based position presentation, we must also adjust the particle velocity and particle movement.

A. Particle Position Representation

In this study, we randomly generated a group of particles (solutions) represented by a permutation sequence that is an ordered list of operations. For an *n*-job *m*-machine problem, the position of particle *k* can be represented by an *m*×*n* matrix, i.e.,

$X^k = \begin{bmatrix} x_{11}^k & x_{12}^k & \cdots & x_{1n}^k \\ x_{21}^k & x_{22}^k & \cdots & x_{2n}^k \\ \vdots & \vdots & & \vdots \\ x_{m1}^k & x_{m2}^k & \cdots & x_{mn}^k \end{bmatrix}$, where x_{ij}^k denotes the priority of operation o_{ij}, which means the operation of job *j* that must be processed on machine *i*.

The Giffler and Thompson (G&T) algorithm is briefly described below.

Notation:

(*i*,*j*) is the operation of job *j* that must be processed on machine *i*

S is the partial schedule that contains scheduled operations

Ω is the set of operations that can be scheduled

$s_{(i,j)}$ is the earliest time at which operation *(i,j)* belonging to Ω can be started.

$p_{(i,j)}$ is the processing time of operation *(i,j)*.

$f_{(i,j)}$ is the earliest time at which operation *(i,j)* belonging to Ω can be finished, $f_{(i,j)} = s_{(i,j)} + p_{(i,j)}$.

G&T algorithm:

Step 1: Initialize $S = \phi$; Ω to contain all operations without predecessors.

Step 2: Determine $f^* = \min_{(i,j)\in\Omega}\{f(i,j)\}$ and the machine m^* on which f^* can be realized.

Step 3:

(1)Identify the operation set $(i', j') \in \Omega$ such that (i', j') requires machine m^*, and $S(i', j') < f^*$.

(2) Choose (i, j) from the operation set identified in Step 3(1) with the largest priority.

(3) Add (i, j) to S.

(4) Assign $s_{(i,j)}$ as the starting time of (i, j).

Step 4: If a complete schedule has been generated, stop. Otherwise, delete (i, j) from Ω, include its immediate successor in Ω, and then go to Step 2.

The movement of particles is modified in accordance with the representation of particle position based on the insertion operator.

B. Particle Velocity

The original PSO velocity concept is that each particle moves according to the velocity determined by the distance between the previous position of the particle and the *gbest* (*pbest*) solution. The two major purposes of the particle velocity are to move the particle toward the *gbest* and *pbest* solutions, and to maintain the inertia to prevent particles from becoming trapped in local optima.

In the proposed PSO, we concentrated on preventing particles from becoming trapped in local optima rather than moving them toward the *gbest* (*pbest*) solution. If the priority value increases or decreases with the present velocity in this iteration, we maintain the priority value increasing or decreasing at the beginning of the next iteration with probability w, which is the PSO inertial weight. The larger the value of w is, the greater the number of iterations over which the priority value keeps increasing or decreasing, and the greater the difficulty the particle has returning to the current position. For an n-job problem, the velocity of particle k can be represented as

$$V^k = \begin{bmatrix} v_{11}^k & v_{12}^k & \cdots & v_{1n}^k \\ v_{21}^k & v_{22}^k & \cdots & v_{2n}^k \\ \vdots & \vdots & \cdots & \vdots \\ v_{m1}^k & v_{m2}^k & \cdots & v_{mn}^k \end{bmatrix}, \text{ where } v_{ij}^k \text{ is the velocity of the operation } o_{ij} \text{ of particle } k,\ v_{ij}^k \in \{-1,0,1\}.$$

The initial particle velocities are generated randomly. Instead of considering the distance from x_{ij}^k to $pbest_{ij}^k(gbest_{ij})$, our PSO considers whether the value of x_{ij}^k is larger or smaller than $pbest_{ij}^k(gbest_{ij})$ If x_{ij}^k has decreased in the present iteration, this means that $pbest_{ij}^k(gbest_{ij})$ is smaller than x_{ij}^k, and x_{ij}^k is set moving toward $pbest_{ij}^k(gbest_{ij})$ by letting $v_{ij}^k \leftarrow -1$. Therefore, in the next iteration, x_{ij}^k is kept decreasing by one (i.e., $x_{ij}^k \leftarrow x_{ij}^k - 1$) with probability w. Conversely, if x_{ij}^k has increased in this iteration, this means that $pbest_{ij}^k(gbest_{ij})$ is larger than x_{ij}^k, and x_{ij}^k is set moving toward $pbest_{ij}^k(gbest_{ij})$ by letting $v_{ij}^k \leftarrow 1$. Therefore, in the next iteration, x_{ij}^k is kept increasing by one (i.e. $x_{ij}^k \leftarrow x_{ij}^k + 1$) with probability w.

The inertial weight *w* influences the velocity of particles in PSO. We randomly update velocities at the beginning of each iteration. For each particle *k* and operation o_{ij}, if v_{ij}^k is not equal to 0, v_{ij}^k is set to 0 with probability (1–*w*). This ensures that x_{ij}^k stops increasing or decreasing continuously in this iteration with probability (1–*w*).

C. Particle Movement

In our PSO, the particle movement is based on the insert operator proposed by Sha and Hsu [14]. We set $x_{ij}^k \leftarrow p + rand_2 - 0.5$ if we want to insert o_{ij} into the *p*th location in the permutation list. In addition, the location of operation o_{ij} in the operation sequence of *k*th *pbest* and *gbest* solution are $pbest_{ij}^k$ and $gbest_{ij}$. When particle *k* moves, for all o_{ij}, if v_{ij}^k equals 0, the x_{ij}^k will be set to $pbest_{ij}^k + rand_2 - 0.5$ with probability c_1 and set to be $gbest_{ij} + rand_2 - 0.5$ with probability c_2, where $rand_2$ is a random variable between 0 and 1, and c_1 and c_2 are constants between 0 and 1, and c_1+$c_2 \leqq 1$. For example, assume that V^k, X^k, $pbest^k$, *gbest*, c_1, and c_2 are as follows:

$$V^k = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix},\ X^k = \begin{bmatrix} 2.5 & 3.3 \\ 1.3 & 4.2 \end{bmatrix},\ pbest^k = \begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix},$$
$$gbest = \begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix},\ c_1 = 0.7,\ c_2 = 0.1$$

For o_{11}:

Because $v_{11}^k \neq 0, x_{11}^k \leftarrow x_{11}^k + v_{11}^k,$ that is, $x_{11}^k = 1.5.$

For o_{12}:

Because $v_{12}^k = 0,$ randomly generate $rand_1 = 0.6.$

Because $rand_1 \leq c_1,$ randomly generate $rand_2 = 0.3.$

Because $pbest_{12}^k \geq x_{12}^k,$ set $v_{12}^k \leftarrow 1,$ and then

$x_{12}^k \leftarrow pbest_{12}^k + rand_2 - 0.5,$ that is, $x_{12}^k = 3.8.$

For o_{21}:

Because $v_{21}^k = 0,$ randomly generate $rand_1 = 0.9.$

Because $rand_1 > c_1 + c_2,$ x_{21}^k does not be changed.

For o_{22}:

Because $v_{22}^k = 0,$ randomly generate $rand_1 = 0.75.$

Because $c_1 < rand_1 \le c_1 + c_2,$ generate $rand_2 = 0.8.$

Because $gbest_{21}^k < x_{22}^k,$ set $v_{22}^k \leftarrow -1,$ and then

$x_{22}^k \leftarrow gbest_{22}^k + rand_2 - 0.5,$ that is, $x_{22}^k = 2.3.$

Finally, after the particle moved, the V^k and X^k are:

$$V^k = \begin{bmatrix} -1 & 1 \\ 0 & -1 \end{bmatrix} \text{ and } X^k = \begin{bmatrix} 1.5 & 3.8 \\ 1.3 & 2.3 \end{bmatrix}.$$

D. Pareto Set

In real world, empirical scheduling decisions should not only involve the deliberation of more than one objective at a time, but also need to prevent the conflict of two or more objectives. The solution set of multi-objective optimization problem with conflicting objective function consisted with the solutions that no other solution is better than all other objective functions is called Pareto optimal. A multi-objective minimization problem with m decision variables and n objectives is given below to describe the concept of Pareto optimality.

The non-dominated solution is defined as solutions which dominate the others but do not dominate themselves. Solution η is said a Pareto-optimal solution if there exist no other solution ξ in the feasible space which could dominate η. The set including all Pareto-optimal solutions is termed the Pareto-optimal Set, or the efficient set. The graph plotted using collected Pareto-optimal solutions in feasible space is designated as Pareto front.

The external Pareto optimal set is employed to deposit a limited size of non-dominated solutions. Maximum size of archive set is specified in advance. This method is applied to forbid missing fragment of non-dominated front during the searching process. The Pareto-optimal front is getting formed as archive updated iteratively. While the archive set is empty enough and a new non-dominated solution is detected, the new solution will enter the archive set. As the new solution enters the archive set, any solution in the archive set dominated by this solution will be withdrawn from the archive set. In case the maximum archive size reaches its preset value, the archive set have to decide which solution could be replaced. In this study, we propose a novel Pareto archive set updating process in order to preclude from losing non-dominated solutions when the Pareto archive set is full. When a new non-dominated solution is discovered, the archive set would be updated when one of the following situation occurs: (i) number of solutions in the archive set is less than the maximum value; (ii) number of the solutions in the archive set is equal to (or greater than) the maximum value, then one of the solutions in the archive set that is most dissimilar to the new solution will be replaced

by the new solution. We measure the dissimilarity by Euclidean distance. A longer distance implies a higher dissimilarity. The non-dominated solution in the Pareto archive set with the longest distance to the new found solution will be replaced.

V. COMPUTATIONAL EXPERIMENTS

In order to compare the performance of our PSO with traditional meta-heuristic algorithm, we code the GA algorithm to program with C++ language in addition. At first, we apply PSO to solve the hardest benchmark problem generated by Guéret and Prins (1999). The program runs 20 times on each problem on a Pentium 4 3.0-GHz computer with 1 GB of RAM running Windows XP. During the pilot experiment, we used four swarm sizes N (50, 100, 150, and 200) to test the algorithm. The outcome of N=150 was best, so that value was used in all further tests. Parameters c_1 and c_2 were tested at various portfolios in the range 0.1–0.7 in increments of 0.2. The inertial weight w was reduced from 0.9 to 0.1 during iterations. The combination of c_1=0.1, c_2=0.8, w=0.1 gave the best results. The maximum iteration limit was set to 60 and the maximum archive size was set to 150.

The GA program also runs 20 times on each problem on a Pentium 4 3.0-GHz computer with 1 GB of RAM running Windows XP. The parameter setting of GA algorithm is described as follows. During the pilot experiment, we used four population sizes N (50, 100, 150, and 200) to test the algorithm. The outcome of N=150 was best, so that value was used in all further tests. The crossover and mutation rate is test in the range of 0.1-0.9. The combination of cross rate equals 0.5, mutation rate equals 0.1 gave the best results. The maximum iteration limit was set to 60 and the maximum archive size was set to 150. The results of MOGA for notorious open shop scheduling problems are demonstrated in Table1.

Table1 The results of MOGA for benchmark problems gp03-gp10

Problem	Makespan		MIT		TFT		CPU
	Best	Average	Best	Average	Best	Average	Time
gp03-01	1168	1168.15	0	0	3174	3384.9	3.59
gp03-02	1170	1170	0	32.6	3177	3397.05	3.75
gp03-03	1168	1168	0	0	3336	3426.2	3.91
gp03-04	1166	1166	0	0	3170	3413	3.75
gp03-05	1170	1170	0	15.9	3181	3403.6	3.75
gp03-06	1169	1169	0	16.1	3177	3394.35	3.90
gp03-07	1165	1165.05	0	16.4	3166	3379.25	3.60
gp03-08	1167	1167	0	0	3172	3366.4	3.90

gp03-09	1162	1162	0	7.85	3167	3402.5	3.60
gp03-10	1165	1165	0	7.9	3172	3385.3	3.90
Average	1167	1167.02	0	9.675	3189.2	3395.255	3.76
gp04-01	1281	1283.7	0	204.1	4325	4705.8	4.22
gp04-02	1270	1271.75	0	147.5	4309	4798.6	4.69
gp04-03	1288	1290.7	0	321.6	4574	4889.9	4.37
gp04-04	1261	1261	233	252	4527	4651.85	4.69
gp04-05	1289	1290.25	0	362.2	4309	4881.25	4.53
gp04-06	1269	1270.85	179	339.7	4539	4848.3	5.00
gp04-07	1271	1277.8	0	194	4582	4845.2	4.53
gp04-08	1259	1259	191	496	4524	4549.3	4.69
gp04-09	1280	1284.5	0	320.4	4316	4642.45	4.69
gp04-10	1263	1263.45	188	219.8	4785	5005.45	4.69
Average	1273.1	1275.3	79.1	285.7	4479	4781.81	4.61
gp05-01	1245	1253.8	454	716.1	5824	5960	6.25
gp05-02	1247	1267.2	270	672	5587	6004.5	6.25
gp05-03	1265	1265	260	339.6	5588	6043.9	6.25
gp05-04	1263	1275.95	231	516.8	5633	5890.1	6.41
gp05-05	1281	1285.5	274	406.1	5574	6019.5	6.25
gp05-06	1270	1282.15	228	484.7	5589	5884.25	6.09
gp05-07	1269	1269.65	0	493.9	5552	5786.8	6.25
gp05-08	1288	1294.45	295	383.7	5836	6048.45	6.40
gp05-09	1262	1274.15	262	589.4	5573	5973.75	6.25
gp05-10	1257	1274.5	237	610.7	5675	6013.65	6.25
Average	1264.7	1274.235	251.1	521.3	5643.1	5962.49	6.26
gp06-01	1266	1284.95	271	671.1	7076	7256.45	8.75
gp06-02	1289	1289.85	0	670.5	6914	7254.25	8.12
gp06-03	1257	1261.8	0	670.9	6842	7233.65	8.60

gp06-04	1275	1283.35	240	902.5	6903	7066.35	9.21
gp06-05	1301	1302.6	105	787.7	6910	7283.45	8.44
gp06-06	1285	1294.95	0	914.4	6918	7251.8	8.44
gp06-07	1292	1295.7	294	640	6826	7454.2	8.44
gp06-08	1268	1271.75	426	864.7	6652	7015.5	8.28
gp06-09	1246	1254.9	467	714.7	6985	7024	8.75
gp06-10	1258	1267.55	504	892.4	6585	7007	8.75
Average	1273.7	1280.74	230.7	772.9	6861.1	7184.665	8.57
gp07-01	1189	1190.55	594	641.3	7974	8025.8	23.44
gp07-02	1186	1190.7	618	668.6	7811	7854.7	23.12
gp07-03	1239	1264.95	253	866.65	8001	8482.15	23.59
gp07-04	1173	1188.15	407	758	7882	8022.65	23.75
gp07-05	1188	1202.85	416	593.5	7991	8188.85	23.60
gp07-06	1200	1236.75	394	903.6	7740	8147.6	23.75
gp07-07	1186	1211.6	362	614	7668	8069.9	23.29
gp07-08	1191	1191	626	735.05	7902	7966.5	23.28
gp07-09	1222	1222	510	618.3	7829	7859.3	23.28
gp07-10	1271	1273.65	556	1145.65	8091	8466.7	23.91
Average	1204.5	1217.22	473.6	754.465	7888.9	8108.415	23.50
gp08-01	1182	1203.1	457	869.4	9072	9283.6	33.12
gp08-02	1166	1184.05	453	867.2	9101	9184.9	33.28
gp08-03	1148	1180.85	416	644	8987	9073.8	32.97
gp08-04	1181	1189	656	768	8955	9041.75	32.81
gp08-05	1224	1227.65	482	899.5	8824	9185.45	32.97
gp08-06	1170	1183.3	700	908.7	8983	9092.35	33.60
gp08-07	1169	1199.15	560	746.2	9028	9271.2	32.97
gp08-08	1182	1191.5	455	921.7	9210	9348.45	33.43
gp08-09	1152	1190.5	471	803.8	8810	9216.15	32.97

gp08-10	1187	1202.8	368	811.7	8910	9250.45	32.50
Average	1176.1	1195.19	501.8	824	8988	9194.81	33.06
gp09-01	1166	1190.05	664	983.6	10109	10275.05	46.56
gp09-02	1158	1173.85	255	733.5	9907	10194.45	45.63
gp09-03	1157	1209.5	689	1177	10113	10537.95	45.78
gp09-04	1164	1181.55	478	786.6	10071	10323.45	46.72
gp09-05	1199	1206.75	346	978.8	10209	10355.3	46.40
gp09-06	1139	1159.6	591	967.7	10071	10221.55	46.10
gp09-07	1153	1159.05	507	615	9870	10105	46.41
gp09-08	1151	1179	561	762.6	10044	10186.85	45.78
gp09-09	1176	1180.2	614	753.4	9877	10129.1	46.25
gp09-10	1142	1165.95	526	750.5	9880	10051.6	46.72
Average	1160.5	1180.55	523.1	850.9	10015.1	10238.03	46.23
gp10-01	1157	1163.9	617	849.55	11165	11292.55	63.75
gp10-02	1155	1170	691	1041.85	11151	11397.55	63.44
gp10-03	1141	1157.8	476	775.95	11064	11212.95	63.75
gp10-04	1113	1136.4	484	692.15	10871	11082.85	63.60
gp10-05	1145	1160.05	522	706.75	11155	11271.35	64.38
gp10-06	1148	1190.2	460	815.15	11181	11488.2	63.44
gp10-07	1139	1160.3	519	855.2	11084	11264.6	63.90
gp10-08	1146	1182.8	474	888.95	11142	11529.45	64.07
gp10-09	1147	1164.2	480	779.75	11070	11326.95	63.90
gp10-10	1163	1180.3	491	741.55	11123	11354	63.60
Average	1145.4	1166.595	521.4	814.685	11100.6	11322.04	63.78

The comparison of MOPSO and MOGA for objectives makespan, machine idle time and total flow time are showed in Table 2, 3, 4 respectively.

Table 2 The comparison of MOPSO and MOGA for makespan

	PSO		GA		Error Ratio	
	Makespan		Makespan		Makespan	
	Best	Avg.	Best	Avg.	Best	Avg.
gp03	1167.0	1167.0	1167.0	1167.0	**0**	**0**
gp04	1272.7	1272.7	1273.1	1275.3	**0**	**0**
gp05	1263.6	1263.7	1264.7	1274.2	**0**	**0**
gp06	1271.6	1272.2	1273.7	1280.7	**0**	**0**
gp07	1195.5	1196.2	1204.5	1217.2	**0**	**0**
gp08	1144.1	1150.9	1176.1	1195.1	**0**	**0**
gp09	1124.5	1132.5	1160.5	1180.5	**0**	**0**
gp10	1094.3	1110.3	1145.4	1166.5	**0**	**0**

Table 3 The comparison of MOPSO and MOGA for machine idle time

	PSO		GA		Error Ratio	
	Machine Idle Time		Machine Idle Time		Machine Idle Time	
	Best	Avg.	Best	Avg.	Best	Avg.
gp03	0.0	3.1	0	9.6	**0**	**0**
gp04	162.7	328.1	79.1	285.7	1.05	0.14
gp05	285.6	500.4	251.1	521.2	0.13	**0**
gp06	308.1	620.2	230.7	772.8	0.33	**0**
gp07	248.8	558.0	473.6	754.4	**0**	**0**
gp08	139.8	392.3	501.8	823.9	**0**	**0**
gp09	143.6	363.8	523.1	850.8	**0**	**0**
gp10	50.5	243.3	521.4	814.6	**0**	**0**

Table 4 The comparison of MOPSO and MOGA for total flow time

	PSO	GA	Error Ratio
	Total Flow Time	Total Flow Time	Total Flow Time

	Best	Avg.	Best	Avg.	Best	Avg.
gp03	3237.5	3405.0	3189.2	3395.2	0.015	0.0028
gp04	4456.5	4775.2	4479.0	4781.8	**0**	**0**
gp05	5575.2	5847.8	5643.1	5962.4	**0**	**0**
gp06	6788.7	7086.5	6861.1	7184.6	**0**	**0**
gp07	7837.6	8025.2	7888.9	8108.4	**0**	**0**
gp08	8890.8	9013.1	8988.0	9194.8	**0**	**0**
gp09	9899.2	10007.3	10015.1	10238.0	**0**	**0**
gp10	10778.2	10950.7	11100.6	11322.0	**0**	**0**

VI. CONCLUSION

Although a large amount of research has addressed the open-shop scheduling problem, most of this has focused on minimizing the maximum completion time (*i.e.*, makespan). Other objectives exist in the real world, such as minimizing the machine idle time, that might help improve efficiency and reduce production costs. PSO, inspired by the behavior of flocks of birds and schools of fish, has the advantages of a simple structure, easy implementation, immediate accessibility, short search time, and robustness. However, few applications of PSO to multi-objective open-shop scheduling problems can be found in the literature. Therefore, we proposed a MOPSO algorithm to solve the open-shop scheduling problem with multiple objectives, including minimization of makespan, total flow time, and machine idle time.

The original PSO was developed for continuous optimization problems. To make it suitable for open-shop scheduling (*i.e.*, a combinational problem), we modified the representation of particle position, particle movement, and particle velocity. We also introduced a Pareto set and used a diversification strategy.

The results demonstrated that the proposed MOPSO could obtain more optimal solutions than the MOGA. The relative error ratios of each problem scenario in our MOPSO algorithm were less than in the MOGA. The performance measure results also revealed that the proposed MOPSO algorithm outperformed MOGA in simultaneously minimizing makespan, total tardiness, and total machine idle time.

We will attempt to apply MOPSO to other shop scheduling problems with multiple objectives in future research. Other possible topics for further study include modification of the particle position, particle movement, and particle velocity representation. Issues related to Pareto optimization, such as solution maintenance strategy and performance measurement, also merit future investigation.

REFERENCES

[1] Gonzalez T., Sahni S., "Open shop scheduling to minimize finish time," *Journal of the ACM*, Vol. 23, 1976, pp. 665–79.

[2] Liaw C.-F., "Applying simulated annealing to the open shop scheduling problem," *IIE Transactions*, Vol. 31, 1999, pp. 457–65.

[3] Liaw C.-F., "A tabu search algorithm for the open shop scheduling problem," *Computers & Operations Research*, Vol.26, 1999, pp. 109–26.

[4] Liaw C.-F., "A hybrid genetic algorithm for the open shop scheduling problem," *European Journal of Operational Research*, Vol.124, 2000, pp. 28–42.

[5] Blum C., "Beam-ACO—Hybridizing ant colony optimization with beam search: an application to open shop scheduling," *Computers & Operations Research*, Vol. 32, 2005, pp. 1565–91.

[6] Prins C., "Competitive genetic algorithms for the open-shop scheduling problem," *Mathematical Methods of Operations Research*, Vol. 52, 2000, pp. 389–411.

[7] Senthilkumar P., Shahbudeen P., "GA based heuristic for the open job shop scheduling problem," *International Journal of Advanced Manufacturing Technology*, Vol. 30, 2006, pp. 297–301.

[8] Tang L., Bai D., "A new heuristic for open shop total completion time problem," *Applied Mathematical Modeling*, vol. 34 2010, pp. 735–743.

[9] Liaw C.-F., "Scheduling preemptive open shops to minimize total tardiness," *European Journal of Operational Research*, Vol.162, 2005, pp. 173–183.

[10] Bllzewicz J., Pesch E., Sterna M., Werner F., "Open shop scheduling problems with late work criteria," *Discrete Applied Mathematics*, Vol. 134, 2004, pp. 1–24.

[11] Kennedy J, Eberhart R.C., "Particle swarm optimization," *Proceedings of the 1995 IEEE International Conference on Neural Networks.* New Jersey: IEEE Press; 1995, pp. 1942–8.

[12] Giffler J, Thompson G.L., "Algorithms for solving production scheduling problems," *Operations Research*, Vol. 8, 1960, pp. 487–503.

[13] Zhang H., Li H., Tam C.M., "Particle swarm optimization for resource-constrained project scheduling," *International Journal of Project Management*, Vol. 24, 2006, pp. 83–92.

[14] Sha D.Y., Hsu C.-Y., "A new particle swarm optimization for the open shop scheduling problem," *Computers & Operations Research*, Vol. 35, 2008, pp. 3243–3261.

[15] Guéret C., Prins C., "A new lower bound for the open-shop problem," *Annals of Operations Research*, vol. 92, 1999, pp. 165–183.