Industrial Engineering
& Management Systems
Vol 13, No 1, March 2014, pp.43-51
ISSN 1598-7248 | EISSN 2234-6473 |
http://dx.doi.org/10.7232/iems.2014.13.1.043

Application of Adaptive Particle Swarm Optimization to Bi-level Job-Shop Scheduling Problem

Chompoonoot Kasemset*
Department of Industrial Engineering, Faculty of Engineering, Chiang Mai University, Chiang Mai, Thailand

(Received: May 9, 2013 / Revised: November 1, 2013 / Accepted: January 15, 2014)

ABSTRACT

This study presents an application of adaptive particle swarm optimization (APSO) to solving the bi-level job-shop scheduling problem (JSP). The test problem presented here is 10×10 JSP (ten jobs and ten machines) with tri-bottleneck machines formulated as a bi-level formulation. APSO is used to solve the test problem and the result is compared with the result solved by basic PSO. The results of the test problem show that the results from APSO are significantly different when compared with the result from basic PSO in terms of the upper level objective value and the iteration number in which the best solution is first identified, but there is no significant difference in the lower objective value. These results confirmed that the quality of solutions from APSO is better than the basic PSO. Moreover, APSO can be used directly on a new problem instance without the exercise to select parameters.

Keywords: Adaptive Particle Swarm Optimization, Job-Shop Scheduling, Bi-level Programming

* Corresponding Author, E-mail: chompoonoot.kasemset@cmu.ac.th

1. INTRODUCTION

Bi-level structure is the simplest class of multi-level formulation. Many planning and scheduling problems of supply chain management and production management can be modeled with this structure. Bi-level scheduling problem is a problem when the decision made on how to schedule jobs considers two levels objectives, upper and lower, as a hierarchy structure. The decision of the upper level generally influences the decision on the lower level. The bi-level scheduling problem is found in many cases, for example: the case of multi-level production systems by Lin *et al*. (1997) and Semnani and Zamanifar (2010), the case of cellular manufacturing by Logendran *et al*. (1995), and the case of job-shop scheduling by Kasemset and Kachitvichyanukul (2010).

When the multi-level structure was applied, solution procedures as exact and heuristics algorithm have been developed by many researchers, but the first technique, exact algorithm, is difficult to get the solution in reasonable computational time so a heuristics method has been widely developed in many research works. The examples of well-known heuristics techniques are: genetic algorithm (GA) applied by Kimms (1999) and Pezzella *et al*. (2008); particle swarm optimization (PSO) applied by Zhang *et al*. (2009), Kuo and Huang (2009), and Chander *et al*. (2011); ant colony optimization used by Semnani and Zamanifar (2010); and combined heuristics used by Logendran *et al*. (1995) and Kuo and Han (2011).

Particle swarm optimization or PSO is a population based search method. The effectiveness of PSO was compared with GA and differential evolution in Kachitvichyanukul (2012) and PSO was mention that its mechanism facilitated in solution improvement within short computational time compared with GA. PSO was widely applied in many scheduling problems, for examples, flow-shop scheduling by Rahimi-Vahed and Mirghorbani

(2007) and Pan *et al*. (2008); job-shop scheduling (JSP) by Sha and Hsu (2006), Lei (2008), Lian *et al*. (2006), Xia and Wu (2005), Pongchairerks and Kachitvichyanukul (2009), Zhang *et al*. (2009), Pratchayaborirak and Kachitvichyanukul (2011), Wisittipanich and Kachitvichyanukul (2013). However, most researchers in this field used PSO in a single and multiple objective optimizations, in contrast, there was only a few researchers working on multi-level programming problems. In addition, the initial step of basic PSO is to setup PSO-parameter, i.e., acceleration constants, number of iteration, number of particle, etc., as optimal values to accelerate the solution improving during PSO process. The parameter optimization for PSO is a tedious job. Many research works applied the concept of design of experiment (DOE) to find optimal value of PSO-parameter.

One method to avoid the step of parameter optimization is to design multi-level PSO proposed by Pongchairerks and Kachitvichyanukul (2009) that separated the top level PSO to fine-tune and assigned the parameter values for PSO and the lower level PSO developed for finding JSP solution. However, this approach has the disadvantage of long computational time.

The initiation of adaptive PSO (APSO) is developed to circumvent the step of parameter optimization. An idea of a self-adaptive PSO is developed to allow an automatic parameter setting within the PSO process. Many research works dealt with APSO for adaptive inertia weight as proposed by Shi and Eberhart (1998), Ueno *et al*. (2005), Gao and Ren (2007), Arumugam and Rao (2008) and Ai and Kachitvichyanukul (2008a); and for acceleration constants proposed by Ai and Kachitvichyanukul (2008b).

The main contribution of this research is to present the application of APSO to solving the bi-level job-shop scheduling problem. This APSO is applied to solve the same problem that was solved by using the basic PSO. The comparisons are made on three parameters: 1) the upper level objective value, 2) the lower level objective value, and 3) the iteration number in which the best solution is first identified. Three parameters are used to display the performance of PSO and APSO in order to compare the quality of the solution and the speed in finding the near-optimal solution.

The organization of this paper is as follows. Preliminaries, including problem formulation, PSO and adaptive concept, are explained in Section 2. Numerical illustration and conclusion are presented in Sections 3 and 4.

2. PRELIMINARIES

2.1 The Bi-level Multi-objective Job-Shop Scheduling Formulation

The bi-level multi-objective job-shop scheduling was first proposed by Kasemset and Kachitvichyanukul (2010). The mathematical model was formulated as bi-level form with the upper level of this model that aims to minimize the total idle time on bottleneck machines (B_n) as Eq. (1) based on the concept of Theory of Constraints (TOC) that aims to maximize the utilization of bottlenecks. The lower level objective is to improve the schedule performance measures formulated as multiple objectives by minimizing the aggregated maximum value of completion time (C_{max}), tardiness (T_{max}), and earliness (E_{max}) using weighting technique as Eq. (1) to (5). When, W_1, W_2, and W_3 are the weights for C_{max}, T_{max}, and E_{max}, respectively.

Objective:
Minimize

$$\sum_{b=1}^{B} [\max_{i,b} \{X_{ib}+s_{ib}+U_i p_{ib}\} - \sum_{i=1}^{n} (s_{ib}+U_i p_{ib})] \quad (1)$$

where X_{ib} solves

Minimize

$$W_1 C_{max}+W_2 T_{max}+W_3 E_{max} \quad (2)$$

and;

$$C_{max} = \max_{i,j} \{X_{ij}+s_{ij}+U_i p_{ij}\} \quad (3)$$
$$T_{max} = \max_{i,j} \{0, (X_{ij}+s_{ij}+U_i p_{ij}) - D_i\} \quad (4)$$
$$E_{max} = \max_{i,j} \{0, D_i - (X_{ij}+s_{ij}+U_i p_{ij})\} \quad (5)$$

When i, j, and b represent jobs, machines, and bottleneck machines, respectively. X_{ij} are decision variables representing starting time of job i on machine j. U_i and D_i are demand and due date of job i. s_{ij} and p_{ij} are set up time and processing time of job i on machine j. Those objectives are subjected to the set of constraint of precedence constraints, machine conflict constraints, job ready time constraint, earliest starting time constraints calculated from transfer lot size, non-negativity constraints and binary constraints. The completed mathematical model and numerical examples are presented in detail as in Kasemset and Kachitvichyanukul (2010).

When solving small size JSP, optimal solution can be derived by using any optimizer program. However, when the size of problem is increased, the solution cannot be obtained by optimizers due to long computational time. Thus, PSO based method was proposed by Kasemset and Kachitvichyanukul (2012) to facilitate in solving JSP when the size of problem was increased. The PSO based method is considered to facilitate in searching for a near optimal solution within reasonable computational time.

2.2 PSO Based Method for Solving Bi-level Multi-objective JSP

PSO is a population based search method introduced by Kennedy and Eberhart (1995). A particle in-

side the swarm is similar to a chromosome in a population of the GA, but the difference point between PSO and GA is that there are no genetic operations, i.e., crossover and mutation, during PSO process.

PSO process starts with randomly initializing the swarm. The dimension of each particle is designed to match with the solution of any problem. During the process, each particle moves through the solution space with its own assigned velocity accelerated toward its previous best position called *pbest* (personal best) and the best position of the swarm called *gbest* (global best). The exchanged experience allows the particles to move to better solutions.

In the basic PSO algorithm, each iteration step mainly consists of only two set of updating equations: velocity as in Eq. (6) and position as in Eq. (7)

$$v_{id} = w\, v_{id} + c_p u(p_{id} - x_{id}) + c_g\, u(p_{gd} - x_{id}) \quad (6)$$

$$x_{id} = x_{id} + v_{id} \quad (7)$$

The velocity equation as Eq. (6) consists of three elements its velocity, cognitive behavior, and social behavior (detail can be found in Ai and Kachitvichyanukul, 2007).

The PSO based method for bi-level multi-objective JSP proposed by Kasemset and Kachitvichyanukul (2012) were developed based on basic PSO, but there are two unique points designed for handling the bi-level model and transfer lot concept that are: 1) The movement of particles occurs by considering two fitness values representing both level objective values from bi-level formulation. 2) The schedule representation step is differently designed comparing with any PSO method for JSPs because the starting time, X_{ij}, should be derived by considering one additional parameter called "Earliest Starting Time" and this parameter can be calculated based on transfer lot concept proposed in Kasemset and Kachitvichyanukul (2010).

The PSO based method for bi-level multi-objective JSP from Kasemset and Kachitvichyanukul (2012) is briefly described as three main steps.

1) Encoding and decoding scheme: This PSO applied operation-based representation and random keys representation proposed addresses in Cheng *et al*. (1996) for decoding process. The advantages of these methods are that every schedule decoded always yield a feasible schedule.
   For an n job and m machine JSP, a chromosome contains $n \times m$ genes. Then, a solution is encoded using random keys representation. Each gene contains number of position and random number. Then, all genes are sorted in ascending order before starting operation-based representation by repeating genes based on number of operations of each job. For n job and m machine, each job appears on the chromosome exactly m times. After that, all genes are sorted back to its position number to present a sequence of operations and ready for schedule representation in the following step.
2) Schedule representation: The decoded particle is transformed to a schedule. In this step, the starting time of each job on each machine (X_{ij}) is calculated. When X_{ij} are known, the finish time of each job can be calculated and the performance measurements, C_{max}, T_{max} and $E_{max,}$ can be also determined in the next step.
3) Fitness value evaluation and updating: As previously mention, when X_{ij} are calculated, the finish time of each job can be derived, so B_n, C_{max}, T_{max}, and E_{max} can be calculated following Eqs. (1) and (3)–(5). In this step, two fitness values, B_n and F_t (following Eq. (2)), are used for particle's movement. To select *gbest*, B_n is firstly considered due to its higher priority as the upper level objective. When there are many particles with minimum B_n, F_t will then be considered. This is the unique point of PSO based method for bi-level formulation.

Addressed by Ai and Kachitvichyanukul (2008b), the problem when basic PSO is adopted is how to set up initial parameters of PSO, i.e., inertia weight, acceleration constants (c_p and c_g), number of particles and other parameters as optimal values. Those parameters surely have effects on the quality of the solution obtained from PSO, so in many research works, the concept of DOE is needed to find the optimal value of those parameters.

One method to avoid the step of parameter optimization is the concept of parameter-adaptive or self-adaptive parameter. PSO with self-adaptive parameter, called APSO, is developed to circumvent the step of parameter optimization. The detail of APSO used in this research is briefly explained in the next section.

2.3 Concept of Self-adaptive Parameter in PSO

The concept of APSO is developed to avoid the process of PSO-parameter selection because the steps for finding the optimal set of parameters are not easy to be carried out. When the problem is changed, the set of these parameters may be different.

Key parameters in PSO can be listed as;

1) Inertia weight (w): Particles move with the new velocity from the combination vectors. Inertia weight is a weight to control the magnitude of the current velocity on updating the new velocity. This parameter is one factor playing important role in the velocity update as in Eq. (6) and this parameter controls the search behavior of the swarm, as well.
2) Acceleration constants: For basic PSO, acceleration constants are c_p and c_g giving relative importance of *pbest* and *gbest* position when the velocity is updated. Each constant controls the distance that a particle is allowed to move from its current position to any best position. Other acceleration constants (i.e., c_l respects to the local best, c_n respects to near-neighbor

best, etc.) can be also used depended on structure of modified PSO.

3) Number of particles: Number of particles or population size represents the number of particles in the system. This parameter has effects on fitness value and computational time. Generally, a small population size leads to poor convergence while a large population size yields good convergence but time consuming.

Beside those parameters previously mention, other parameters, i.e., number of iterations, re-initialization related factors, and so on, are also affected the performance of PSO. In this research, APSO proposed by Ai and Kachitvichyanukul (2008a, b) are used to solve the bi-level JSP. This APSO applied the concept of self-adaptive parameter for inertia weight and acceleration constants (c_p and c_g). The short explanations are address for these two parameters as follows.

2.3.1 Adaptive inertia weight

Proposed by Ai and Kacitvichyanukul (2008a), the inertia weight is set in the range of minimum (w^{min}) and maximum value (w^{max}). Whenever the swarm velocity index ($\bar{\omega}$) as in Eq. (8) is lower than the desired velocity index (ω^*) as in Eq. (9), the inertia weight is increased, and reversely when the swarm velocity index is greater than the desired velocity index, the inertia weight is decreased. The amount of increases and decreases of inertia weight depends on the difference between the velocity index of the swarm and the desired velocity index.

$$\bar{\omega} = \frac{\sum_{l=1}^{L}\sum_{h=1}^{H}|\omega_{lh}|}{L \times H} \tag{8}$$

$$\omega^* = \begin{cases} (1-(1.8\tau/T))\omega^{max}. & 0 \le \tau \le T/2 \\ (0.2-(0.2\tau/T))\omega^{max}. & T/2 \le \tau \le T \end{cases} \tag{9}$$

where, l is particle index (l = 1, 2, ⋯, L), h is dimension index (h = 1, 2, ⋯, H), τ is iteration index (τ = 1, 2, ⋯, T), and ω^{max} is maximum velocity index.

The updating of inertia weight is as follow Eqs. (10)–(13).

$$\Delta w = \frac{\omega^* - \bar{\omega}}{\omega^{max}} (w^{max}\text{-}w^{min}) \tag{10}$$

$$w = w + \Delta w \tag{11}$$

$$w = w^{max} \text{ if } w > w^{max} \tag{12}$$

$$w = w^{min} \text{ if } w < w^{min} \tag{13}$$

2.3.2 Adaptive acceleration constants

The concept of adaptive acceleration used in this research is modified from Ai and Kachitvichyanukul (2008b). For this research, adaptive concept is used only for c_p and c_g while the original work proposed to use adaptive concept for four acceleration constants. The concept used here starts with determining the difference between the corresponding objective function of particle's position and the objective function of respective term. In minimizing problem, the large difference means high priority of that term. Then, particles intend to move toward to that term.

The acceleration constants can be determined as the proportion of the respective degree of importance to the constant c^*, which is defined as the sum of the acceleration constants. The degree of importance of the whole swarm consisting L particles can be express as Eqs. (14)–(15).

$$\Delta Z_p = \sum_{l=1}^{L} max\ \{Z_l - Z_p,\ 0\} \tag{14}$$

$$\Delta Z_g = \sum_{l=1}^{L} max\ \{Z_l - Z_g,\ 0\} \tag{15}$$

For any iteration, Z_l is the fitness value of particle l and Z_p and Z_g are the fitness value of the *pbest* and *gbest* at current iteration. The acceleration constants can be derived as the proportion of degree of importance and updated using exponential weighted moving average technique for avoiding rapid changing of parameters following Eqs. (16)–(18).

$$\Delta Z = \Delta Z_p + \Delta Z_g \tag{16}$$

$$c_p = \alpha c_p + (1-\alpha)\frac{\Delta z_p}{\Delta z} c^* \tag{17}$$

$$c_g = \alpha c_g + (1-\alpha)\frac{\Delta z_g}{\Delta z} c^* \tag{18}$$

In this research paper, c_p and c_g are adaptive parameters. The process of adaptive c_p and c_g can be presented as pseudo-code as in Figure 1.

From Figure 1, starting from particles initialization or encoding process, particles are improved since step 04 based on Eqs. (6) and (7) as a concept of basic PSO. After that, they are decoded and evaluated using two fitness functions following Eq. (1) for B_n value and Eq. (2) for F_t value. Then, B_n is used for the first priority to set *pbest* and *gbest* and F_t is used as the second priority as step 07–14. Then, inertia weight, c_p and c_g are updated as step 15–18. This process is continued until the number of iteration is completed.

3. NUMERICAL ILLUSTRATION

The test problem from Kasemset (2009) is solved by applied both PSO and APSO. This test problem is the modified JSP with ten jobs and ten machines (10×10) considering monthly demand, job due-date, transfer lot size and set-up time of each operation. The detail of job sequences is provided in Appendix.

Bottleneck machines are known as machine number 2, 7, and 9. (The detail of bottleneck identification can be found in detail as addressed in Kasemset (2009) and Kasemset and Kachitvichyanukul (2007). Based on the bi-level formulation, the upper level objective is to

```
01: Start
02: Initialize particle swarm (encoding)
03: While (no. of iteration is not completed)
04: Particles move using Eqs. (6)–(7)
05: Decoding each particle
06: Evaluate fitness values for each particle in the swarm
    using Eqs. (1)–(5)
07: Sort B_n as the first fitness value
08: Obtain minimum B_n
09: While (more than one particle has minimum B_n)
10: Sort F_t as the second fitness value
11: Obtain minimum F_t
12: Obtain pbest
13: Sort pbest for all particle (by 1st fitness value follow by
    2nd fitness value)
14: Obtain gbest
15: Adaptive w using Eqs. (8)–(13)
16: Obtain w
17: Adaptive c_p and c_g using Eqs. (14)–(15)
18: Obtain c_p and c_g
19: While (no. of iteration is not completed), repeat the
    cycle from step 04-18
20: end
```

Figure 1. Pseudo-code of proposed adaptive particle swarm optimization.

minimize the total idle time of three bottleneck machines and the lower level objective is to minimize the aggregated value of C_{max}, T_{max}, and E_{max} with W_1, W_2, and $W_3 = 1$.

3.1 Solving by PSO

Previously, this 10×10 JSP is solved by PSO based method proposed by Kasemset and Kacitvichyanukul (2012) with parameter setting as shown in Table 1.

Table 1. Parameters used in basic particle swarm optimization

Parameter	Value
c_p, c_g	1.6
w_{max}, w_{min}	0.9, 0.6
No. of iteration	1000
No. of particle	100
R_{start}^{th}	150
R_{iter}	100
R_{ratio}	0.80

Adapted from Kasemset and Kacitvichyanukul (2012).

Table 2. Results from particle swarm optimization (30 replications)

Parameter	B_n	F_t	Iteration no. in which the best solution is first identified
Min	22470	49821	675
Max	31816	57934	998
Mean	26971.0	53479.2	886.9
SD	2188.03	2241.35	85.60

The result conclusion of PSO proposed by Kasemset and Kacitvichyanukul (2012) shown in Table 2. Three parameters are collected as 1) 1st level objective value (B_n), 2) 2nd level objective value (F_t), and 3) iteration number in which the best solution is first identified.

3.2 Solving by APSO

APSO is used to solve the test problem and the results are also collected 30 replications. Parameters, excepted c_p and c_g, are used as previous PSO test. Initially, c_p and c_g are equally set at 1, c^* is 4 and α is 0.8. The test results are shown in Table 3.

Table 3. Results from adaptive particle swarm optimization

Replication no.	B_n	F_t	c_p	c_g	Iteration
1	26043	54141	0.82	0.98	787
2	29938	50922	0.76	0.94	763
3	24477	54680	0.83	1.08	825
4	26072	57327	0.78	1.03	958
5	26811	52820	0.73	1.00	676
6	21810	51841	0.81	1.19	882
7	25388	52721	0.87	1.09	453
8	27280	58923	0.84	1.10	674
9	25781	49327	0.88	1.06	696
10	25662	51228	0.67	1.02	950
11	27434	54453	0.79	1.02	297
12	26138	51025	0.77	0.99	326
13	21233	54745	0.86	1.10	711
14	23284	54936	0.77	1.08	998
15	21901	52597	0.73	1.06	720
16	26088	53941	0.74	1.04	720
17	26108	52363	0.77	0.98	907
18	23168	55647	0.82	1.16	642
19	25521	52902	0.77	1.06	903
20	22611	52728	0.69	1.07	876
21	24216	54422	0.79	1.08	616
22	26943	58147	0.80	1.07	753
23	28050	61838	0.93	1.13	931
24	26871	53686	0.79	1.01	634
25	23380	50629	0.81	1.12	666
26	25117	53911	0.75	1.01	933
27	26725	54728	0.80	0.98	753
28	26058	59130	0.80	1.06	349
29	25144	51628	0.83	1.13	665
30	26663	53452	0.83	1.14	399
Min	21233	49327	-	-	297
Max	29938	61838	-	-	998
Mean	25397.17	54047.79	0.794	1.059	715.4
SD	1986.438	2839.98	-	-	194.4

Table 4. B_n mean comparison

Parameter	PSO	APSO
Mean	26971.0	25397.2
SD	2188.03	1986.44
Max	31816	29938
Min	22470	21233
p-value of mean comparison	0.0018	

PSO: particle swarm optimization, APSO: adaptive PSO.

Table 5. F_t mean comparison

Parameter	PSO	APSO
Mean	53479.2	54047.8
SD	2241.35	2839.98
Max	57934	61838
Min	49821	49327
p-value of mean comparison	0.2031	

PSO: particle swarm optimization, APSO: adaptive PSO.

3.3 Results Comparison

3.3.1 B_n comparison

The B_n results of PSO and APSO are presented as the difference in mean comparison shown in Table 4.

From Table 4, one-side upper B_n-mean comparison is tested between PSO and APSO. The test hypothesizes are set as follow:

H_0: Mean B_n solved by PSO is equal to mean B_n solved by APSO.
H_1: Mean B_n solved by PSO is greater than mean B_n solved by APSO

The p-value from the test is 0.0018 and less than the significant level (α) 0.05, so the null hypothesis is rejected. Thus, it can be concluded that mean B_n solved by PSO differs from mean B_n solved by APSO. In fact, there is strong evidence that mean B_n solved by PSO exceeds mean B_n solved by APSO. For this test problem, APSO can help in B_n improvement.

3.3.2 F_t comparison

The F_t results are presented as the difference in mean comparison as shown in Table 5.

From Table 5, one-side upper F_t-mean comparison is tested between PSO and APSO. The test hypothesizes are set as follow:

H_0: Mean F_t solved by PSO is equal to mean F_t solved by APSO.
H_1: Mean F_t solved by PSO is greater than mean F_t solved by APSO.

The p-value is 0.2031 and greater than the significant level (α) 0.05, so the null hypothesis is accepted. It can be concluded that there is no difference in F_t-mean from both methods. It implies that APSO cannot help in F_t improvement for this test problem.

3.3.3 Iteration number in which the best solution is first identified

The number of iteration found the best solution of each replication for 30 replications of PSO and APSO are compared and presented as the difference in mean comparison shown in Table 6.

Table 6. No. of iteration mean comparison

Parameter	PSO	APSO
Mean	886.9	715.4
SD	85.60	194.40
Max	998	998
Min	675	297
p-value of mean comparison	<0.0001	

PSO: particle swarm optimization, APSO: adaptive PSO.

To confirm this, Table 6 shows one-side upper of this value mean comparison as the test hypothesizes are:

H_0: Mean iteration number from PSO is equal to mean iteration number from APSO.
H_1: Mean iteration number from PSO is greater than mean iteration number from APSO.

The test p-value is less than 0.0001. When p-value is very small, the null hypothesis is rejected. Thus, there is strong evidence that mean iteration number from PSO exceeds mean iteration number from APSO. For this test problem, it can be concluded that the convergence speed of APSO is faster than PSO.

Three comparisons addressed here show that APSO outperforms PSO in term of better B_n value and the iteration number in which the best solution is first identified in this research work.

3.4 Solving by PSO with c_p and c_g set from APSO

To confirm the performance of APSO, the additional test is set to test that APSO can help in finding the optimal PSO-parameter. From Table 3 in Section 4.2, results from APSO, average c_p and c_g are 0.794 and 1.059, respectively. Table 7 shows the results of 30 replication solved by PSO with adjusted c_p and c_g.

The parameter mean comparisons between basic PSO and PSO with c_p and c_g values set from APSO are shown in Table 8. The test hypothesizes are set as follow:

H_0: 1st variable (V_1) is equal to 2nd variable (V_2).
H_1: 1st variable (V_1) is greater to 2nd variable (V_2).

Table 7. Results from PSO with c_p and c_g from APSO

Replication no.	B_n	F_t	Iteration
1	22077	51146	746
2	25403	57431	268
3	26210	49400	655
4	25336	53626	923
5	26692	49126	493
6	25292	53031	534
7	23658	53743	576
8	24582	52017	939
9	27064	54530	640
10	23002	50629	572
11	25648	52313	644
12	24708	52216	572
13	24997	53238	989
14	26037	51611	542
15	24941	53913	383
16	24690	53822	928
17	25006	53524	922
18	27899	52310	572
19	26128	51872	317
20	22956	50906	636
21	25362	51094	858
22	25897	60043	922
23	22740	62959	434
24	27402	51520	924
25	21410	52436	550
26	23505	55520	625
27	23972	52547	529
28	23594	52021	918
29	20498	50629	821
30	27406	55734	174
Min	20498	49126	174
Max	27899	62959	989
Mean	24803.73	53163.57	653.53
SD	1799.087	2916.287	221.82

PSO: particle swarm optimization, APSO: adaptive PSO.

These results show that B_n and the iteration number in which the best solution is first identified are significantly different (using $\alpha = 0.05$). For F_t value, it cannot be concluded that there is any difference in mean of F_t from both methods.

These test results led us to conclude that APSO is useful in finding the optimal c_p and c_g for improving the B_n value and convergence speed of PSO.

4. CONCLUSION

This study presented the use of PSO and APSO in bi-level problem solving. PSO and APSO are used to solve the same problem of 10×10 bi-level JSP. The results are represented as the comparisons of three factors: 1) 1st level objective value (B_n), 1) 2nd level objective value (F_t), and 3) iteration number in which the best solution is first identified.

The results show that the solutions solved by APSO are better than PSO in terms of B_n and the iteration number in which the best solution is first identified when the mean comparison of the two parameters are tested at $\alpha = 0.05$. In contrast, F_t values from both methods are not significantly different. For B_n and the iteration number in which the best solution is first identified, the comparisons confirm that the solution quality of PSO and APSO is significantly different, which indicates that APSO can improve the solution of the bi-level problem in this study.

In an additional test, when c_p and c_g are set for PSO from average values of those parameters from APSO, the performance of PSO was improved in terms of B_n and the convergence speed, as well. The results demonstrate that the advantage of APSO allows researchers to remove the step of parameter optimization and parameter setting required when the basic PSO is used with the equivalent solution quality and the number of iteration needed.

Table 8. 3-Parameter mean comparisons

Parameter	B_n		F_t		Iteration no. in which the best solution is first identified	
	PSO (V_1)	PSO-adjust c_p & c_g (V_2)	PSO (V_1)	PSO-adjust c_p & c_g (V_2)	PSO (V_1)	PSO-adjust c_p & c_g (V_2)
Mean	26971.0	24803.7	53479.2	53163.6	886.9	653.5
SD	2188.03	1799.09	2241.35	2916.29	85.60	221.82
Max	31816	27899	57934	62959	998	989
Min	22470	20498	49821	49126	675	174
p-value of mean comparison (1-side upper, $V_1 > V_2$)	<0.0001		0.3192		<0.0001	

PSO: particle swarm optimization.

ACKNOWLEDGMENTS

The author would like to acknowledge Professor Voratas Kachitvichyanukul for the valuable comments and suggestions on the preparation of this paper.

REFERENCES

Ai, T. J. and Kachitvichyanukul, V. (2007), Dispersion and velocity indices for observing dynamic behavior of particle swarm optimization, *Proceedings of the IEEE Congress on Evolutionary Computation*, Singapore, 3264-3271.

Ai, T. J. and Kachitvichyanukul, V. (2008a), A study on adaptive particle swarm optimization for solving vehicle routing problems, *Proceedings of the 9th Asia Pacific Industrial Engineering and Management Systems Conference*, Bali, Indonesia.

Ai, T. J. and Kachitvichyanukul, V. (2008b), Adaptive particle swarm optimization algorithms, *Proceedings of the 4th International Conference on Intelligent Logistics Systems*, Shanghai, China, 460-469.

Arumugam, M. S. and Rao, M. V. C. (2008), On the improved performances of the particle swarm optimization algorithms with adaptive parameters, cross-over operators and root mean square (RMS) variants for computing optimal control of a class of hybrid systems, *Applied Soft Computing*, **8**(1), 324-336.

Chander, A., Chatterjee, A., and Siarry, P. (2011), A new social and momentum component adaptive PSO algorithm for image segmentation, *Expert Systems with Applications*, **38**(5), 4998-5004.

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.

Gao, Y. and Ren, Z. (2007), Adaptive particle swarm optimization algorithm with genetic mutation operation, *Proceedings of the 3rd International Conference on Natural Computation*, Haikou, China, 211-215.

Kachitvichyanukul, V. (2012), Comparison of three evolutionary algorithms: GA, PSO, and DE, *Industrial Engineering and Management Systems*, **11**(3), 215-223

Kasemset, C. (2009), *TOC based job-shop scheduling*, dissertation, Asian Institute of Technology, Pathumthani, Thailand.

Kasemset, C. and Kachitvichyanukul, V. (2007), Simulation-based procedure for bottleneck identification. In: *AsiaSim 2007*, Springer, Heidelberg, Germany, 47-55.

Kasemset, C. and Kachitvichyanukul, V. (2010), Bi-level multi-objective mathematical model for job-shop scheduling: the application of Theory of Constraints, *International Journal of Production Research*, **48**(20), 6137-6154.

Kasemset, C. and Kachitvichyanukul, V. (2012), A PSO-based procedure for a bi-level multi-objective TOC-based job-shop scheduling problem, *International Journal of Operational Research*, **14**(1), 50-69.

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

Kimms A. (1999), A genetic algorithm for multi-level, multi-machine lot sizing and scheduling, *Computers and Operations Research*, **26**(8), 829-848.

Kuo, R. J. and Huang, C. C. (2009), Application of particle swarm optimization algorithm for solving bi-level linear programming problem, *Computers and Mathematics with Applications*, **58**(4), 678-685.

Kuo, R. J. and Han, Y. S. (2011), A hybrid of genetic algorithm and particle swarm optimization for solving bi-level linear programming problem: a case study on supply chain model, *Applied Mathematical Modelling*, **35**(8), 6905-3917.

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

Lian, Z., Jiao, B., and Gu, X. (2006), A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan, *Applied Mathematics and Computation*, **183**(2), 1008-1017.

Lin, F. R., Shaw, M. J., and Locascio, A. (1997), Scheduling printed circuit board production systems using the two-level scheduling approach, *Journal of Manufacturing Systems*, **16**(2), 129-149.

Logendran, R., Mai, L., and Talkington, D. (1995), Combined heuristics for bi-level group scheduling problems, *International Journal of Production Economics*, **38**(2/3), 133-145.

Pan, Q. K., Fatih Tasgetiren, M., and Liang, Y. C. (2008), A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem, *Computers and Operations Research*, **35**(9), 2807-2839.

Pezzella, F., Morganti, G., and Ciaschetti, G. (2008), A genetic algorithm for the flexible job-shop scheduling problem, *Computers and Operations Research*, **35**(10), 3202-3212.

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

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), 83-92.

Rahimi-Vahed, A. R. and Mirghorbani, S. M. (2007), A multi-objective particle swarm for a flow shop scheduling problem, *Journal of Combinatorial Optimization*, **13**(1), 79-102.

Semnani, S. H. and Zamanifar, K. (2010), New approach to multi-level processor scheduling, *International Journal on Artificial Intelligence Tools*, **19**(3), 335-346.

Sha, D. Y. and Hsu, C. Y. (2006), A hybrid particle swarm optimization for job shop scheduling problem, *Computers and Industrial Engineering*, **51**(4), 791-808.

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

Ueno, G., Yasuda, K., and Iwasaki, N. (2005), Robust adaptive particle swarm optimization, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Waikoloa, HI, 3915-3920.

Wisittipanich, W. and Kachitvichyanukul, V. (2013), An efficient PSO algorithm for finding Pareto-frontier in multi-objective job shop scheduling problems, *Industrial Engineering and Management Systems*, **12**(2), 151-160.

Xia, W. and Wu, Z. (2005), An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems, *Computers and Industrial Engineering*, **48**(2), 409-425.

Zhang, G., Shao, X., Li, P., and Gao, L. (2009), An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem, *Computers and Industrial Engineering*, **56**(4), 1309-1318.

APPENDIX

There are ten products (jobs) and ten machines. The detail of jobs is provided in Table A1. The monthly demand of all jobs in this case is 400 units equally. The due date of all jobs is 43200 and transfer lot size is 100 units.

Table A1. Test problem 10×10 job-shop scheduling problem

Process step	Product no. with machine, setup time (min), process time (min)									
	1	2	3	4	5	6	7	8	9	10
1	4, 8, 8	5, 7, 2	9, 8, 3	7, 9, 4	3, 6, 9	1, 9, 9	7, 0, 5	4, 9, 8	0, 9, 4	3, 0, 5
2	8, 6, 8	3, 0, 5	8, 6, 1	2, 6, 8	4, 8, 8	4, 8, 1	1, 8, 6	6, 7, 3	6, 7, 1	0, 5, 9
3	6, 9, 4	6.6.9	0, 8, 3	1, 6, 1	9, 8, 2	5, 6, 4	4, 9, 7	3, 8, 2	3, 8, 1	1, 8, 2
4	5, 9, 9	4, 7, 5	1, 6, 5	4, 9, 9	8, 9, 5	6, 6, 6	3, 9, 6	2, 5, 1	7, 8, 5	8, 6, 7
5	1, 6, 7	2, 9, 4	6, 6, 4	3, 5, 4	0, 9, 9	8, 0, 8	0, 9, 5	1, 7, 1	1, 6, 6	7, 5, 6
6	2, 8, 9	8, 6, 6	5, 8, 5	6, 7, 5	2, 6, 7	2, 0, 8	8, 9, 7	5, 9, 4	2, 0, 9	9, 9, 6
7	9, 7, 7	0, 9, 2	7, 7, 8	5, 6, 6	6, 9, 5	7, 6, 9	2, 6, 6	7, 8, 5	4, 7, 6	6, 5, 8
8	7, 9, 9	1, 8, 2	4, 8, 5	0, 7, 6	5, 6, 8	9, 6, 2	5, 9, 9	0, 6, 2	5, 5, 8	4, 8, 1
9	0, 8, 6	7, 9, 4	2, 5, 5	9, 6, 3	7, 6, 7	3, 7, 9	6, 5, 2	8, 9, 5	8, 9, 3	5, 5, 9
10	3, 9, 2	9, 6, 3	3, 7, 7	8, 6, 7	1, 8, 6	0, 8, 8	9, 7, 1	9, 7, 9	9, 9, 7	2, 9, 6