Hindawi Publishing Corporation
Discrete Dynamics in Nature and Society
Volume 2010, Article ID 838596, 11 pages
doi:10.1155/2010/838596

Research Article

A Local and Global Search Combine Particle Swarm Optimization Algorithm for Job-Shop Scheduling to Minimize Makespan

Zhigang Lian[1, 2, 3]

[1] *School of Electronic and Information Engineering, Shanghai DianJi University, Shanghai 200240, China*
[2] *School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China*
[3] *Postdoctoral Centre, Jiangnan Shipyard (Group) Co., Ltd. Shanghai, Shanghai 201913, China*

Correspondence should be addressed to Zhigang Lian, lllzg@163.com

Received 5 February 2010; Revised 15 March 2010; Accepted 24 June 2010

Academic Editor: Elena Braverman

The Job-shop scheduling problem (JSSP) is a branch of production scheduling, which is among the hardest combinatorial optimization problems. Many different approaches have been applied to optimize JSSP, but for some JSSP even with moderate size cannot be solved to guarantee optimality. The original particle swarm optimization algorithm (OPSOA), generally, is used to solve continuous problems, and rarely to optimize discrete problems such as JSSP. In OPSOA, through research I find that it has a tendency to get stuck in a near optimal solution especially for middle and large size problems. The local and global search combine particle swarm optimization algorithm (LGSCPSOA) is used to solve JSSP, where particle-updating mechanism benefits from the searching experience of one particle itself, the best of all particles in the swarm, and the best of particles in neighborhood population. The new coding method is used in LGSCPSOA to optimize JSSP, and it gets all sequences are feasible solutions. Three representative instances are made computational experiment, and simulation shows that the LGSCPSOA is efficacious for JSSP to minimize makespan.

1. Introduction

The job-shop scheduling problem (JSSP) is very important in field of production management. The JSSP has been proved to be a difficult task for human planners and schedulers, particularly if optimal solutions are required. With traditional optimization approaches, finding optimal solution to large size combinatorial problems is not a practicable option, because of the vast amount of computer time needed to find such solutions. Predecessors have proposed many different approaches to optimize JSSP and have obtained some harvest, but by current algorithms, even moderately sized problems cannot be solved to guarantee optimality. One of the first attempts to optimize a simple JSSP through application of genetic algorithm can be seen in the research of Davis in 1985. Since then, a significant number of successful applications of GAs to JSSP have appeared in [1–4], and so forth. Gharbi and Haouari have used an approximate decomposition algorithm for scheduling on parallel machines [5]. Xia and Wu presented an effective hybrid optimization approach for multiobjective feasible JSSP in [6]. Many researches show that GAs that are used to solve JSSP are feasible, however, their performance is far from satisfactory. Particle swarm optimization (PSO) is an evolutionary computation technique developed by Eberhart and Kennedy in 1995 [7, 8], inspired by social behavior of bird flocking or fish schooling. Similar to genetic algorithms (GA), PSO is a population-based optimization tool. In recent years, there have been a lot of works focused on the PSO that have been applied widely in the function optimization, artificial neural network training, pattern recognition, fuzzy control, and some other fields where GA can be applied, but PSOA used to solve discrete optimization problem such as JSSP does not have a rich literature. Clerc and Kennedy in [9] have researched the particle swarm stability and convergence in a multidimensional complex space. Goh et al. researched on competitive and cooperative coevolutionary approach to multiobjective particle swarm optimization algorithm design [10]. Eberhart and Shi have researched inertia weights and constriction factors, developments, applications, and resources of particle swarm optimization in [11, 12], at the same time they have presented a modified particle swarm optimizer in [13]. Shi et al. presented an improved GA and a novel PSO-GA-based hybrid algorithm in [14]. Fan has presented a modification to PSOA in [15] and He et al. have put forward a particle swarm optimizer with passive congregation in [16]. Kennedy and Mendes in [17] investigated the impacts of population structures to the search performance of SPSO. Robinson et al. in [18] have investigated particle swarm, genetic algorithm, and their hybrids. Chen and Li [19, 20], Shi and Eberhart [21], and Trelea [22] have researched parameter selection of particle swarm optimization, respectively . Yin and Zhang presented a new heuristic algorithm for solving the JSSP [23]. Lian et al. in [24] presented a similar particle swarm optimization algorithm, and through simulation experiment showed that it was effective to solve optimization problems especially with large size.

In this paper, a local and global search combine particle swarm optimization algorithm (LGSCPSOA) is used for JSSP to minimize the makespan. In LGSCPSOA the new coding scheme is used, which gets every particle is feasible scheduling. This work differs from the existing ones at least in three aspects. It firstly proposes LGSCPSOA for JSSP; the second is to investigate the effectiveness of various parameters for different size JSSP; the third, computational experiments are performed to show that the LGSCPSOA is valid for JSSP. The rest of the paper is organized as follows. Section 2 gives a definition of the JSSP and formulates its model that is minimum makespan. The LGSCPSOA implementation for JSSP is presented in Section 3. Section 4 discusses experimental results obtained by LGSCPSOA with different parameters on public benchmark JSSP. Finally, Section 5 summarizes the contribution of this paper and conclusions.

2. JSSP Description

In the static JSSP, a finite number of jobs are to be processed by a finite number of machines. Each job consists of a predetermined sequence of task operations, each of which needs to be processed without preemption for a given period of time on a given machine. Tasks of the same job cannot be processed concurrently and each job must visit each machine exactly once. Each operation cannot be commenced until the processing is completed, if the precedent operation is still being processed. A schedule is an assignment of operations to time slots on a machine. The makespan of JSSP is maximum completion time of all jobs, and the objective is to find a schedule that minimizes the makespan. A good schedule is one that minimizes the total amount of time machines are idle.

The JSSP Description

Let n jobs ($j = 1, 2, \ldots, n$) be processed on m machines $M_1, M_2, \ldots, M_m$. Job j consists of operations sequence $O_{1j}, O_{2j}, \ldots, O_{ij}, \ldots$, which must be processed with precedence constraints of the form $O_{ij} \rightarrow O_{i+1,j}$ ($i = 1, 2, \ldots, m_j - 1$). Each operation O_{ij} being processed needs T_{ij} ($i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$) time units on one machine of $M_1, M_2, \ldots, M_m$. All jobs are available at time zero and the operation of each job on a machine requires an uninterrupted period of time. This problem is called the job-shop scheduling problem (JSSP). Its model is set up as follows.

Let: $J = \{1, 2, \ldots, n\}$ for the job set;

$p_{kj} = \text{find}(O_{ij} == k \mid i = 1, 2, \ldots, m)$ denote machine number of working procedure order k of job j process;

$C(i, j)$ for the completion time of job j on machine i as follows:

$$\begin{gathered} C(1,1) = C\left(p_{(O_{11}-1)1}, 1\right) + T(1,1), \\ C(1,j) = \max\left\{C(1,j'), C\left(P_{(O_{1j}-1)}, j\right)\right\} + T(1,j), \\ C(i,1) = C\left(p_{(O_{i1}-1)1}, 1\right) + T(i,1), \\ C(i,j) = \max\left\{C(i,j'), C\left(p_{(O_{ij}-1)j}, j\right)\right\} + T(i,j). \end{gathered} \tag{2.1}$$

Remark 2.1. For $j = 1, \ldots, n$; $i = 1, \ldots, m$, j' is anterior processing job of job j on machine i.

The relation calculation makespan is as follows:

$$C_{\max} = \max_{1 \leq i \leq m} \max_{1 \leq j \leq n} C(i, j). \tag{2.2}$$

The scheduling problem is to find out the best operation sequences on all machines in order to minimize makespan. In this case, the makespan implies the criterion to be optimized.

In follow section, propose LGSCPSOA and use it to find a near optimal schedule whose makespan is the minimum over the set of all possible feasible schedules.

3. LGSCPSO Algorithm

3.1. Original Particle Swarm Optimization

Particle swarm optimization (PSO) is an evolutionary computation technique mimicking the behavior of flying birds and their means of information exchange. The system of PSO is initialized with a population (named swarm in PSO) of random solutions and searches for optima by updating generations. Each individual or potential solution, named particle, flies with a velocity which is adjusted according to the flying experiences of its own and its colleagues. PSOA has been developing rapidly and has been applied widely since it was introduced, as it is easily understood and realized. In PSOA, the potential solutions are "flown" through the problem space by following the current optimum particles, and the detailed information will be given in following.

Suppose that the searching space is N-dimensional and m particles form the colony. The ith particle represents an N-dimensional vector X_i $(i = 1, 2, \ldots, m)$, and it means that the ith particle locates at $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ in the searching space. The particle's fitness could be calculated through putting its position into a designated objective function. The ith particle's "flying" velocity is also an N-dimensional vector as $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$. Denote the best position of the ith particle as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$, and the best position of all colonies as $P_g = (p_{g1}, p_{g2}, \ldots, p_{gD})$, respectively. The original particle swarm optimization algorithm (OPSOA) could be performed by the following equations [7, 8]:

$$\begin{gathered} v_{id}(k+1) = v_{id}(k) + c_1 r_1 \big(p_{id}(k) - x_{id}(k)\big) + c_2 r_2 \big(p_{gd}(k) - x_{id}(k)\big), \\ x_{id}(k+1) = x_{id}(k) + v_{id}(k+1), \end{gathered} \tag{3.1}$$

where k represents the iterative number, c_1, c_2 are learning factors, usually $c_1 = c_2 = 2$. r_1, r_2 are random numbers between $(0, 1)$. The termination criterion for the iterations is determined according to whether the max generation or a designated value of the fitness of P_g is reached.

3.2. LGSCPSOA for JSSP

3.2.1. Coding Scheme in LGSCPSOA for JSSP

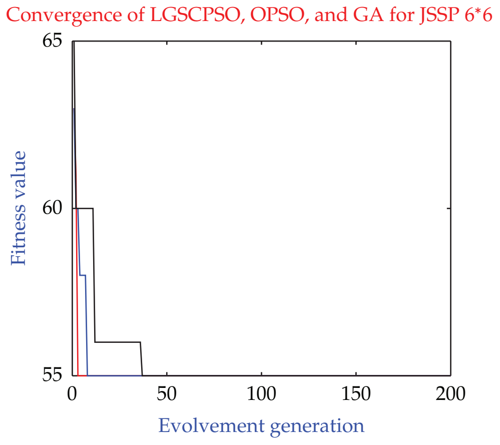
OPSOA has been introduced as an optimization technique in real-number space, but many optimization problems in discrete space such as JSSP. OPSOA has a disadvantage tendency to get stuck in a near optimal solution and may find it difficult to improve solution accuracy by fine tuning. This section presents LGSCPSOA to optimize JSSP in discrete space, and develop an effective "problem mapping" and "solution generation" mechanism.

In many algorithms for JSSP using the machine coding, and it gets some sequences are not feasible solutions. In LGSCPSOA the working procedure coding is used, and it gives birth to all children (sequences) are feasible solutions.

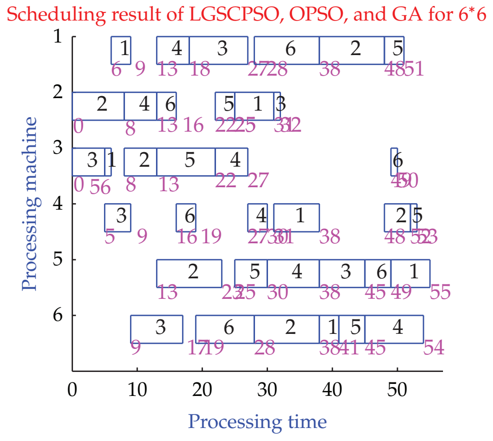
Suppose $X = [x_1, x_2, \ldots, x_n]$ $(x_i \in R, i = 1, 2, \ldots n)$ be one random particle, following steps make particles able to denote JSSP scheduling and will be used in LGSCPSOA to solve JSSP.

(1) Sorting the weight $X = [x_1, x_2, \ldots, x_n]$ from small to large gets $X' = [x'_1, x'_2, \ldots, x'_n]$, and using y_i denotes the position of x'_i in $X = [x_1, x_2, \ldots, x_n]$. Obtain the new sequence $Y = [y_1, y_2, \ldots, y_n]$. For example, 3×2 JSSP, $X = [0.8637, 0.1872, 0.8725, 0.6442, 0.2132, 0.6429]$, sorting X gets $X' = [0.1872, 0.2132, 0.6429, 0.6442, 0.8637, 0.8725]$, then get the new sequence $Y = [2, 5, 6, 4, 1, 3]$;

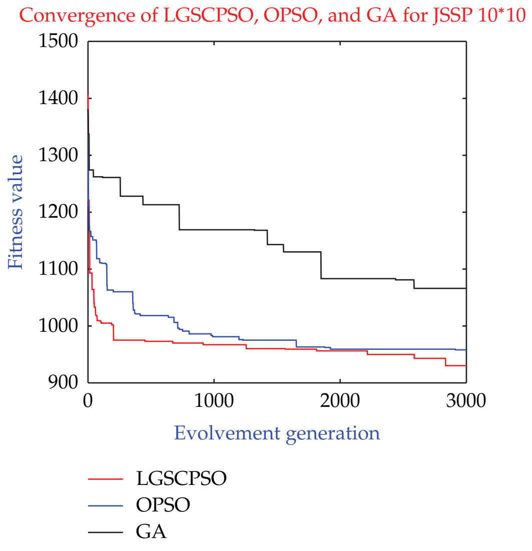
(2) Every weight of sequence Y is divided by machine number M, and then rounded towards plus infinity to get a final one feasible solution sequence. For the above



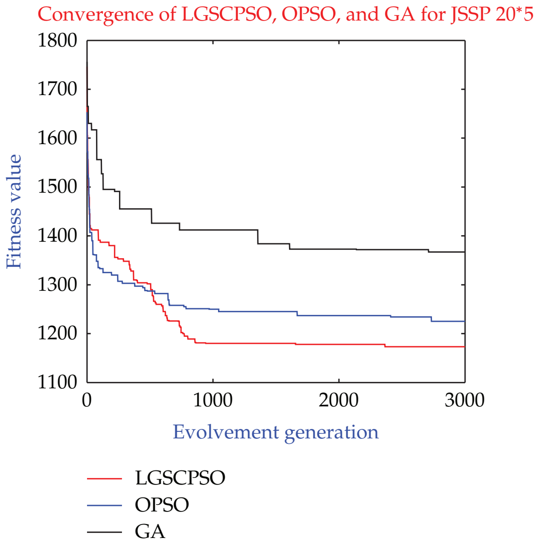
(a)



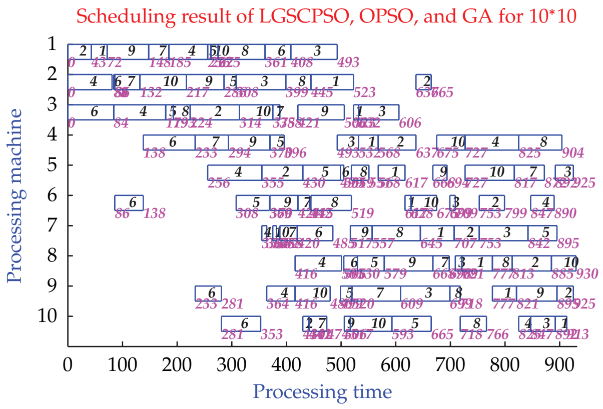
(b)



(c)

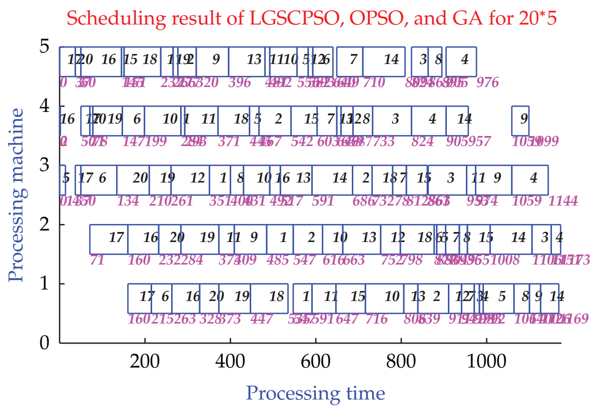


(d)



(e)

Figure 1: Continued.



(f)

Figure 1: Convergence figure and GANTT of OPSO, GA, and LGSCPSOA for JSSP.

above example $Y/2 = [2/2, 5/2, 6/2, 4/2, 1/2, 3/2]$ then rounded towards plus infinity to obtain a final solution sequence se = $[1, 3, 3, 2, 1, 2]$;

(3) *Working procedure coding*: Suppose $n \times m$ JSSP, randomly give birth to following sequence:

$$\text{se}(\text{sequence}) = [1, 3, \dots, n, 5, 3, \dots, n, \dots, 1, 2, \dots, n], \tag{3.2}$$

where the number indicates job and the same number denotes different working procedure of the job according to their order of appearance in se. For example, the 3×2 JSSP, solution sequence se $= [1, 3, 3, 2, 1, 2]$, which denote in turn process working procedure 1 of job 1, working procedure 1 of job 3, working procedure 2 of job 3, working procedure 1 of job 2, working procedure 2 of job 1, and lastly process working procedure 2 of job 2. Using this working procedure coding scheme makes every sequences is feasible solutions.

3.2.2. Iterative Model of the LGSCPSOA

In OPSOA, the information of individual best and global best are shared only by next generation particles. Based on OPSOA, in this paper, I firstly present LGSCPSOA, in which it also adequately utilizes information of the best particle of every generation population, at the same time; in every generation, introducing into some best particles displaces random selecting particles in population. The detailed information will be given in the following.

Suppose that the searching space is n-dimensional and m particles form the colony. The ith particle represents an n-dimensional vector X_i $(i = 1, 2, \dots, m)$, and it means that the ith particle locates at $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$ in the searching space. The position of each particle is a potential result. We could calculate the particle's fitness by putting its position into a designated objective function. When the fitness is lower, the corresponding X_i is "better." The ith particle's "flying" velocity is also an n-dimensional vector as $V_i = (v_{i1}, v_{i2}, \dots, v_{in})$. Denote the best position of the ith particles as $P_i = (p_{i1}, p_{i2}, \dots, p_{in})$, the best position of the kth generation population (local best particle) as $P_l = (p_{l1}, p_{l2}, \dots, p_{ln})$, and the best position of all colonies (global best particle) as $P_g = (p_1, p_2, \ldots, p_n)$, respectively. Using the best particle of every generation population merges into the local best particle population and displaces random selecting particles in local best generation. After finding three best values, the particle updates its velocity and positions with the following formulas:

$$\begin{aligned} v_{id}(k+1) = {} & w v_{id}(k) + c_1 r_1 \left(\alpha\left(p_{id}(k) - x_{id}(k)\right) + (1-\alpha)\left(p_{ld}(k) - x_{id}(k)\right)\right) \\ & + c_2 r_2 \left(p_{gd}(k) - x_{id}(k)\right), \end{aligned} \tag{3.3}$$

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \quad (i = 1, 2, \ldots, m;\ d = 1, 2, \ldots, n) \tag{3.4}$$

$$P_l(k+1) = P_l(k-1, k-2, \ldots, 1) \cup P_l(k), \tag{3.5}$$

where w is an inertia weight, which is initialized typically in the range of $[0,1]$. The local best particle of k generation is $u\%$ of the k generation neighborhood particles whole number. $\alpha \in [0,1]$ is weight index that is chosen according to different optimization problems, which reflects relatively important degree of best position of the ith particle and the kth generation population particle. The variables c_1 and c_2 are acceleration constants, which control how far a particle will move in a single iteration, and other parameters are the same as in (2.1) and (2.2). The termination criterion for the iterations is determined according to whether the end generation or a designated value of the fitness of P_g is reached. In LGSCPSOA, each particle of the swarm shares mutual information and benefits from previous experiences of all other colleagues during the search process.

3.2.3. Pseudo Code of LGSCPSOA

(See Algorithm 1).

4. Experimental Results and Discussion

To analyze the effectiveness and performance of LGSCPSOA for JSSP to minimize makespan, three representative instances taken from the internet based on practical data have been selected to experiment, and through simulation, the best effective parameters of LGSCPSOA for JSSP to minimize makespan are found out.

To get the average performance of the LGSCPSOA ten runs on each instance are performed and the solution quality is averaged, and let the best of them be the solution of this algorithm. We use the arithmetic mean, lower, and upper bound of makespan, and obtaining the best solution time four indexes to estimate LGSCPSOA. The computation comparing results of GA, OPSOA, and LGSCPSOA with different operators or parameters are as in Table 1.

Remark 4.1. In Table 1, P (problem) denotes the problem size, and BS (best solution) is this problem's optimum value, and PS (population size) and GN (generation number) indicate the population size and algorithm terminate generation. In this computing experiment, we let the displaced particles number $r_N = \text{PS}/3$;

From the Table 1 we can obtain that the LGSCPSOA is clearly better than the OPSOA and GA for JSSP. LGSCPSOA obtains smaller upper bound, arithmetic mean, and lower bound of makespan than OPSOA and GA in run ten times. The best solution found in each

Table 1: The comparisons of GA, OPSOA and LGSCPSOA with different parameters for JSSP.

P	BS	PSGN	A	aC	Min/Max/Mean/Times 0.1 (M2)	0.2 (M4)	0.3 (M6)	0.4 (M8)	0.5 (M9)
				0.1	55/58/55.8/7	55/57/55. 2/9	55/59/56.1/7	55/59/55.8/8	55/59/56.3/5
				0.2	55/58/55.8/7	55/59/55.7/8	55/59/8/55.8	55/59/56.2/7	55/59/56.4/5
				0.3	55/58/56.2/5	55/58/55.8/7	55/59/55.7/8	55/59/55.5/8	55/58/55. 6/8
				0.4	55/59/55.8/7	55/59/55.9/7	55/59/55.8/7	55/59/56.1/7	55/58/55.8/7
			LGCPSO	0.5	55/58/55.9/6	55/58/55.65/8	55/60/56.15/7	55/59/56.7/5	55/59/55.5/9
6 × 6	55	100		0.6	55/59/55.9/7	55/59/55. 9/8	55/59/56.2/6	55/59/56.5/5	55/59/56.2/7
		200		0.7	55/59/55.8/7	55/59/56.1/7	55/59/56.3/6	55/59/55.8/7	55/59/56.2/7
				0.8	55/58/55.3/8	55/59/56.1/7	55/59/56.3/6	55/59/55.8/7	55/59/55.4/8
				0.9	55/58/55.5/8	55/59/55.2/9	55/59/56.3/7	55/59/55.6/9	22/59/55.7/7
			PSO	1	55/59/55.6/9	55/59/55.7/8	55/59/55.7/7	55/59/55.9/7	55/59/56.1/7
			GA	C1	55/59/57.45/3	55/59/57.4/0	55/59/57.7/1	55/59/57.3/2	55/59/57.3/2
				C2	55/59/57.9/1	56/59/58/0	55/59/57.6/1	57/60/58.1/0	57/59/58.3/0
				0.1	946/1078/998.7/0	955/1040/1000.4/0	946/1072/1004.1/0	956/1060/1005.7/0	943/1083/1007.3/0
				0.2	951/1038/989.6/0	**930**/1049/990.1/1	945/1008/990.2/0	962/1032/999.6/0	967/1067/1013.9/0
				0.3	951/1052/993.5/0	951/1061/997.5/0	968/1067/1023.2/0	951/1037/1005.2/0	956/1040/1001.2/0
				0.4	952/1048/1000.6/0	961/1048/1004.9/0	955/1064/1004.2/0	937/1063/1004.8/0	965/1063/1009.3/0
			LGCPSO	0.5	951/1063/999.8/0	946/1045/999.8/0	954/1085/1001.7/0	948/1032/996.8/0	955/1049/1010.3/0
10 × 10	930	200		0.6	951/1044/1001.2/0	951/1042/994.2/0	957/1051/999/0	951/1052/997.5/0	937/1039/994/0
		3000		0.7	937/1028/987.5/0	946/1072/988.9/0	949/1093/1025.4/0	951/1032/996.4/0	963/1088/1.0076/0
				0.8	953/1030/991.8/0	937/1076/996.9/0	952/1045/997.1/0	937/1049/1001.9/0	951/1056/1003.5/0
				0.9	937/1057/986.8/0	951/1030/988.6/0	937/1047/995.2/0	937/1053/996.7/0	960/1054/998.8/0
			PSO	1	1003/1115/1051.4/0	998/1040/1018/0	971/1035/1011/0	958/1057/1014/0	979/1081/1032.2/0
			GA	C1	1066/1159/1121.2/0	1066/1169/1125.2/0	1084/1165/1110.4/0	1066/1181/1121.2/0	1074/1170/1115.1/0
				C2	1096/1184/1153.9/0	1108/1200/1160.3/0	1102/1206/1165/0	1109/1218/1172.9/0	1107/1192/1163.5/0
				0.1	1289/1473/1410.4/0	1214/1461/1334.4/0	1207/1405/1274.6/0	1209/1314/1245.6/0	1351/1454/1406.6/0
				0.2	1200/1422/1307/0	**1173**/1348/1.2515/0	1184/1638/1252.4/0	1201/1289/1244.8/0	1334/1442/1395.4/0
				0.3	1285/1469/1411/0	1211/1430/1313.4/0	1195/1275/1234.6/0	1186/1331/1244.4/0	1282/1427/1375.4/0
				0.4	1316/1451/1398/0	1233/1429/1292.8/0	1178/1290/1232.8/0	1185/1271/1231.2/0	1366/1429/1398.4/0
				0.5	1308/1417/1370.2/0	1193/1405/1306.4/0	1199/1311/1255.4/0	1197/1297/1247/0	1271/1407/1366.8/0
20 × 5	1165	200		0.6	1279/1463/1380.6/0	1185/1267/1219.6/0	1200/1230/1218.2	1214/1287/1254.8/0	1219/1341/1286.4/0
		3000		0.7	1302/1457/1375.8/0	1188/1343/1262/0	1216/1265/1244/0	1185/1273/1245.2/0	1180/1332/1262.4/0
				0.8	1263/1436/1361/0	1226/1301/1257.8/0	1197/1268/1241/0	1199/1336/1256.6/0	1215/1262/1241.6/0
				0.9	1258/1443/1319.8/0	1195/1306/1258.9/0	1192/1285/1232.2/0	1206/1322/1254.4/0	1198/1264/1233.2/0
			PSO	1	1264/1296/1281.2/0	1226/1304/1262.8/0	1225/1315/1263.6/0	1232/1287/1253.6/0	1207/1236/1216.4/0
			GA	C1	1399/1487/1426.5/0	1435/1519/1488.3/0	1367/1494/1444.8/0	1414/1498/1453.1/0	1396/1516/1442.6/0
				C2	1445/1535/0/1485.6	1435/1555/1499.9/0	1444/1522/1491.6/0	1400/1490/1442.4/0	1418/1529/1500.3/0

```
Step 1: Initialize parameters; including swarm size PS, end of generation $G$, and other
        parameters will be used in LGSCPSOA;
Step 2: Segment and scheduling;
    (i)   Generate stochastically initialization population and velocity using Section 3.2.1 coding
          scheme
    (ii)  Evaluate each particle's fitness
    (iii) Initialize $gbest$ position with the lowest fitness particle in the swarm
    (iv)  Initialize $ibest$ and $lbest$ position with a copy of particle itself
    (v)   $k := 0$
    While (the end $G$ of generation is not met)
    {
    (i)   $k := k + 1$
    (ii)  Generate next generation particle population by recurrence equations (3.3), (3.4)
          and (3.5)
    (iii) Evaluate swarm
              {(1) Compute each particle's fitness in the swarm
               (2) Find new $P_g$ of the swarm, $P_l$ of generation population and $P_i$ of each
                   particle by comparison, and update $P_i$ and $P_g$
               (3) Update $P_l$ with (3.5)
                   Using the best particle of $k$ generation replace the $k - 1$ generation $u$%
                   particles which are random selected}
    }
Step 3: Output optimization result $P_g$;
End
```

Algorithm 1: Pseudo Code of LGSCPSOA.

category of LGSCPSOA with different parameter is illustrated with bold letters. From Table 1 one can observe that LGSCPSOA with $w \approx 0.2$ and $\alpha \approx 0.2$ has the highest performance since using these parameter has smaller upper bound, arithmetic mean and lower bound of makespan in relation to the solutions obtained by the other parameters schemes. The convergence figure and GANTT of the best solution of different size problems are shown in Figure 1.

From simulation figure, we can see clearly that the convergence rate of LGSCPSOA is faster than the OPSOA and GA. And from above experimenting data, we can testify that the new LGSCPSOA is more efficacious than OPSOA and GA for JSSP to minimize makespan.

5. Conclusion and Perspectives

Although there is a huge amount of literature on OPSOA, there are not too many papers focused on solving scheduling of JSSP. In this paper, we have discussed a new approach LGSCPSOA for JSSP to minimize makespan, in which p_i, p_l, p_g, and V_i give out the information to others, and it is a one-way information sharing mechanism. There are not many parameters needed to be tuned in LGSCPSOA. LGSCPSOA requires only primitive and simple mathematical operators, and is computationally inexpensive in terms of both memory requirements and time. Although it does not guarantee the optimality for 20×5 JSSP, such an approach provides solution with good quality in a reasonable time limit and compared with OPSOA and GA. The performance of the new approach is evaluated in comparison with GA and OPSOA for three representative instances and obtained results show that the effectiveness of the proposed approach is the best. The proposed LGSCPSOA approach in this paper can be considered as effective mechanisms for optimization technique from this point of view.

There are a number of research directions that can be considered as useful extensions of this research. Although the LGSCPSOA is tested with three representative instances, a more comprehensive computational study should be made to test the efficiency of proposed solution technique. The LGSCPSOA proposed in this paper for small size JSSP is very efficacious, but for large size problem is not approving. We just simulate the effectiveness when $r_N = p\text{size}/3$ for JSSP, and in the future can research the efficiency of different r_N. Applying LGSCPSOA to other discrete combinatorial optimization problems is also possible in further research. The development of OPSOA is still ongoing, and furthermore, there are still many unknown areas in LGSCPSOA research such as the mathematical validation of particle swarm theory.

Acknowledgments

This work is supported by the Research Activation Fee of Shanghai Dian Ji University (Grant no. 09C403), Humanities and Social Sciences Youth Fund Project of Ministry of Education (Grant no. 09yjc630151), and Shanghai Universities Select and Train Outstanding Young Teachers in special funds for scientific research (Grant no. sdju200903).

References

[1] S. Kobayashi, I. Ono, and M. Yamamura, “An efficient genetic algorithm for job shop scheduling problems,” in *Proceedings of the 6th International Conference on Genetic Algorithms (ICGA ’95)*, pp. 506–511, 1995.

[2] R. Nakano and T. Yamada, “Conventional genetic algorithm for job shop problem,” in *Proceedings of the 4th International Conference on Genetic Algorithm*, pp. 474–479, 1991.

[3] G. Shi, “A genetic algorithm applied to a classic job-shop scheduling problem,” *International Journal of Systems Science*, vol. 28, no. 1, pp. 25–32, 1997.

[4] H. Zhou, W. Cheung, and L. C. Leung, “Minimizing weighted tardiness of job-shop scheduling using a hybrid genetic algorithm,” *European Journal of Operational Research*, vol. 194, no. 3, pp. 637–649, 2009.

[5] A. Gharbi and M. Haouari, “An approximate decomposition algorithm for scheduling on parallel machines with heads and tails,” *Computers and Operations Research*, vol. 34, no. 3, pp. 868–883, 2007.

[6] W. Xia and Z. Wu, “An effetcive hybrid optimization approach for multi-objetcive flexible job-shop scheduling problems,” *Computers & Industrial Engineering*, vol. 48, no. 2, pp. 409–425, 2005.

[7] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, pp. 39–43, Nagoya, Japan, 1995.

[8] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of the 1st IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Australia, 1995.

[9] M. Clerc and J. Kennedy, “The particle swarm: explosion stability and convergence in a multi-dimensional complex space,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[10] C. K. Goh, K. C. Tan, D. S. Liu, and S. C. Chiam, “A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design,” *European Journal of Operational Research*, vol. 202, no. 1, pp. 42–54, 2010.

[11] R. C. Eberhart and Y. Shi, “Particle swarm optimization: developments, applications and resources,” in *Proceedings of the IEEE Conference on Evolutionary Computation (ICEC ’01)*, vol. 1, pp. 81–86, 2001.

[12] R. C. Eberhart and Y. Shi, “Comparing inertia weights and constriction factors in particle swarm optimization,” in *Proceedings of the IEEE Conference on Evolutionary Computation (ICEC ’00)*, pp. 84–88, Washington, DC, USA, 2000.

[13] Y. Shi and R. Eberhart, “A modified particle swarm optimizer,” in *Proceedings of the IEEE World Congress on Computational Intelligence*, pp. 69–73, May 1998.

[14] X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu, and L. M. Wang, “An improved GA and a novel PSO-GA-based hybrid algorithm,” *Information Processing Letters*, vol. 93, no. 5, pp. 255–261, 2005.

[15] H. Y. Fan, “A modification to particle swarm optimization algorithm,” *Engineering Computations*, vol. 19, no. 7-8, pp. 970–989, 2002.

[16] S. He, Q. H. Wu, J. Y. Wen, J. R. Saunders, and R. C. Paton, “A particle swarm optimizer with passive congregation,” *BioSystems*, vol. 78, no. 1–3, pp. 135–147, 2004.

[17] J. Kennedy and R. Mendes, “Population structure and particle swarm performance,” in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, pp. 1671–1676, IEEE Pres, 2002.

[18] J. Robinson, S. Sinton, and Y. Rahmat-Samii, “Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna,” in *Proceedings of the IEEE Antennas and Propagation Society International Symposium and URSI National Radio Science Meeting*, vol. 1, pp. 168–175, IEEE, San Antonio, Tex, USA, 2002.

[19] X. Chen and Y. Li, “On convergence and parameters selection of an improved particle swarm optimization,” *International Journal of Control, Automation, and Systems*, vol. 6, no. 4, pp. 559–570, 2008.

[20] X. Chen and Y. Li, “A modified PSO structure resulting in high exploration ability with convergence guaranteed,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 37, no. 5, pp. 1271–1289, 2007.

[21] Y. Shi and R. C. Eberhart, “Parameter seletcion in particle swarm optimization, evolutionary programming VII,” in *Proceedings of the 7th Annual Conference on Evolutionary Programming*, pp. 591–600, New York, NY, USA, 1998.

[22] I. C. Trelea, “The particle swarm optimization algorithm: convergence analysis and parameter selection,” *Information Processing Letters*, vol. 85, no. 6, pp. 317–325, 2003.

[23] A. Yin and S. Zhang, “A new heuristic algorithm for solving the job shop scheduling problem,” in *Proceedings of the International Conference on Computational Methods in Science and Engineering (ICCMSE '07)*, vol. 963 of *AIP Conference Proceedings*, pp. 1412–1416, December 2007.

[24] Z. Lian, B. Jiao, and X. S. A. Gu, “A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan,” *Applied Mathematics and Computation*, vol. 183, no. 2, pp. 1008–1017, 2006.