

An Efficient Hybrid Particle Swarm Optimization for the Job Shop Scheduling Problem

Xue-Feng Zhang
Graduate School of Information
Science and Electrical Engineering
Kyushu University
Fukuoka, Japan 819-0382
Email: xuefeng@ar.is.kyushu-u.ac.jp

Miyuki Koshimura
Graduate School of Information
Science and Electrical Engineering
Kyushu University
Fukuoka, Japan 819-0382
Email: koshi@inf.kyushu-u.ac.jp

Hiroshi Fujita
Graduate School of Information
Science and Electrical Engineering
Kyushu University
Fukuoka, Japan 819-0382
Email: fujita@inf.kyushu-u.ac.jp

Ryuzo Hasegawa
Graduate School of Information
Science and Electrical Engineering
Kyushu University
Fukuoka, Japan 819-0382
Email: ryuzo@inf.kyushu-u.ac.jp

Abstract—This paper proposes a hybrid particle swarm optimization algorithm for solving Job Shop Scheduling Problems (JSSP) to minimize the maximum makespan. A new hybrid heuristic, based on Particle Swarm Optimization (PSO), Tabu Search (TS) and Simulated Annealing (SA), is presented. PSO combines local search (by self-experience) with global search (by neighboring experience), achieving a high search efficiency. TS uses a memory function to avoid being trapped at a local minimum, and has emerged as an effective algorithmic approach for the JSSP. This method can also be referred to as calculation of the horizontal direction. SA employs certain probability to avoid becoming trapped in a local optimum and the search process can be controlled by the cooling schedule (also known as calculation of vertical direction). By reasonably combining these three different search algorithms, we develop a robust, fast and simply implemented hybrid optimization algorithm HPTS (Hybrid of Particle swarm optimization, Tabu search and Simulated annealing). This hybrid algorithm is applied to the standard benchmark sets and compared with other approaches. The experimental results show that the proposed algorithm could obtain the high-quality solutions within relatively short computation time. For 6 of 43 instances, new upper bounds among the unsolved problems are found in a short time in HPTS.

Index Terms—Job Shop Scheduling Problem, Particle Swarm Optimization, Tabu Search.

I. INTRODUCTION

The Job Shop Scheduling Problem (JSSP) is one of the existing combinatorial optimization problems and it has been demonstrated to be an NP-hard problem [1]. In JSSP, each one of n jobs ($n \geq 1$) must be processed passing through m machines ($m \geq 1$) in a given sequence. The sequence of m machines is different for each different job and cannot be changed during the processing. When one job was processed on a machine, it can be considered as one operations, each job j needs a combination of m operations to complete the work. One operation is processed on one of m machines, and

just only one operation can be processed at a time. Any job cannot interrupt the machine that is processing one operation of another job. Each machine can process at most one operation at the same time. The main objective of the JSSP is to find a schedule of operations that can minimize the maximum completion (called makespan) that is the completed time of carrying out total operations in the schedule for n jobs and m machines [2].

JSSP can be applied to the manufacture processing and actually effects the production time and the cost of production for a plant. During the past few decades, JSSP has attracted many researchers to develop algorithms. Because JSSP is an NP-hard problem, it is difficult to develop a perfect algorithm to find a solution with a reasonable time especially for higher dimensions. Recently, many researchers made use of evolution algorithm to solve the problem. such as shifting bottleneck approach (SB) [3], tabu search algorithm(TS) [4], genetic algorithm(GA) [5], simulated annealing(SA) [6], particle swarm optimization(PSO) [7], and ant colony(ACO) [8]. Most studies indicate that a single technique can not solve this stubborn problem. Much work has been done on hybrid methods involving GA,SA,TS,and SB techniques as hybrid methods can provide high-quality solution within reasonable computing time, such as Goncalves' HGA [5], Tsung-Lieh Lin's MPSO [2], Wei-Jun Xia's HPSO [9], Fevrier Valdez's PSO+GA [10] [11]. A comprehensive survey of scheduling techniques can be seen from Blazevic et al. and Jain and Meeran. In addition, many research results of the JSSP show that it is very difficult to obtain a good-enough solution only by single search scheme. Motivated by these perspectives, we propose an efficient hybrid intelligent algorithm for the JSSP based on PSO, TS and SA. PSO, TS and SA are evolution computation techniques. They exhibit implicit parallelism and contain certain redundancy and

historical information of past solutions. Moreover, TS and SA also have better local search features. Therefore, the proposed hybrid particle swarm optimization algorithm also effectively exploits the capabilities of distributed and parallel computing of swarm intelligence approaches to achieve the better solution for JSSP.

In this paper, we present an efficient hybrid particle swarm optimization for the Job Shop Scheduling Problem. The remainder of the paper is organized as follows. In Section II, we present the different classes of schedules. In section III, we discuss and analyzes our approach to solve the JSSP. Section IV gives experimental results of the HPTS algorithm and other competitive approaches. Finally, the main conclusions are drawn.

II. PROBLEM FORMULATION

A. Job Shop Scheduling Problem

The JSSP consists of n jobs and m machines. Each job must go through m machines to complete its work. We consider one job consists of m operations. Each operation uses one of m machines to complete one job's work for a fixed time interval. Once one operation is processed on a given machine, it cannot be interrupted before it finishes the job's work. The sequence of operations of one job should be predefined and maybe different for any job. In general, one job being processed on one machine is considered as one operation as noted above, then every job has a sequence of m operations. Each machine can process only one operation during the time interval. The objective of JSSP is to find an appropriate operation permutation for all jobs that can minimize the makespan C_{max} , i.e., the maximum completion time of the final operation in the schedule of $n \times m$ operations.

Let $J = 0, 1, \dots, n, n+1$ denote the set of operations to be scheduled and $M = 0, 1, \dots, m, m+1$ the set of machines. The operations 0 and $n+1$ are dummy, have no duration and represent the initial and final operations. The operations are interrelated by two kinds of constraints. Firstly, the precedence constraints, which force each operation j to be scheduled after all predecessor operations, P_i , are completed. Secondly, operation j can only be scheduled if the machine it requires is idle. Further, let d_j denote the fixed duration (processing time) of operation j .

Let F_j represent the finish time of operation j . A schedule can be represented by a vector of finish times $(F_1, \dots, F_m, F_{m+1})$. Let $A(t)$ be the set of operations being processed at time t , and let $r_{jm} = 1$ if operation j requires machine m to be processed and $r_{jm} = 0$ otherwise.

The conceptual model of the JSSP can be described in the following way [5]:

$$\text{Minimize } F_{n+1}(C_{max}) \quad (1)$$

$$F_k \leq F_j - d_j, j = 1, \dots, n+1, k \in P_j \quad (2)$$

$$\sum_{j \in A(t)} r_{jm} \leq 1, m \in M, t \geq 0 \quad (3)$$

$$F_j \geq 0, j = 1, \dots, n+1 \quad (4)$$

The objective of Eq.1 minimizes the finish time of operation $n+1$ (the last operation), and therefore minimizes the makespan. Constraints in Eq.2 impose the precedence relations between operations and constraints in Eq.3 state that one machine can only process one operation at a time. Finally Eq.4 forces the finish times to be nonnegative.

B. Particle Swarm Optimization

Particle Swarm Optimization (PSO) algorithm is introduced by Kennedy and Eberhart [12] [13] based on the social behavior metaphor. In PSO, a group of particles, without quality and volume, fly through a D-dimensional space, adjusting their positions in search space according to their own experience and their neighbors. Although approximation methods do not guarantee achieving optimal solutions, they are able to attain near-optimal solutions. Therefore, approximation methods are more suitable for larger problems. The particle swarm concept was based on the premise of social behavior. The original intent was to graphically simulate the graceful but unpredictable choreography of a bird flock. A PSO algorithm mimics the behavior of flying birds and their means of information exchange to solve optimization problems. PSO has been introduced as an optimization technique in real-number spaces. But many optimization problems are set in a space featuring discrete components. Typical examples include problems that require ordering and route planning, such as in scheduling and routing problems [14]. They are described as follows [15]:

$$\begin{aligned} V_{id} &= \omega \times V_{id} + C_1 \times \text{Rand}() \times (P_{id}^{\text{best}} - P_{id}) \\ &+ C_2 \times \text{Rand}() \times (P_{gid}^{\text{best}} - P_{id}) \end{aligned} \quad (5)$$

$$P_{id} = P_{id} + V_{id} \quad (6)$$

where V_{id} represents the velocity of particle i . It also can be regarded as the distance to be traveled by particle i from its current position. P_{id} represents the particle position, P_{id}^{best} called "pbest", the local best solution, represents particle i 's best previous position, and P_{gid}^{best} called "gbest", the global best solution, represents the best position among all particles in the swarm. ω is an inertia weight. It regulates the trade-off between the global exploration and local exploitation abilities of the swarm. The acceleration constants C_1 and C_2 represent the weights of the stochastic acceleration terms that pull each particle toward "pbest" and "gbest" positions. $\text{Rand}()$ is a random function with range $[0, 1]$.

For Eq.5, the first part represents the inertia of previous velocity; the second part is the "cognition" part, which represents individuals thinking independently; and the third part is the "social" part, which represents cooperation among the particles [16].

III. THE PROPOSED ALGORITHM FOR JSSP

A. Implementation of Hybrid Particle Swarm Optimization

Some issues are in applying PSO algorithm to solve scheduling problems. The original PSO design is developed

to solve continuous function. The PSO algorithm is problem-independent, which means little specific knowledge relevant to a given problem is required. All we have to know is the fitness evaluation of each solution. This advantage makes PSO more robust than many other search algorithms. However, since PSO is a stochastic search algorithm, it is prone to inadequate global search at the end of a run. PSO may fail to find the required optimum in cases when the problem to be solved is too complicated and complex. SA employs certain probability to avoid becoming trapped in a local optimum, and the search process can be controlled by the cooling schedule. By designing the neighborhood structure and cooling schedule of TS and SA, we can control the search process and avoid individuals being trapped in local optimum more efficiently. Therefore, a hybrid algorithm of PSO, TS and SA, named HPTS, is presented in Figure 1.

1) Encoding scheme and initial solution: Suppose that the searching space is D-dimensional and m particles form the swarm. The i th particle represents a D-dimensional vector X_i ($i = 1, 2, \dots, m$). It means that the i th particle locates at $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ ($i = 1, 2, \dots, m$). 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. The i th particle "flying" velocity is also a D-dimensional vector, denoted as $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$ ($i = 1, 2, \dots, m$). Denote the best position of the i th particle as $P_i = (p_{i1}, p_{i2}, \dots, p_{iD})$, and the best position of the swarm as $G_i = (g_{i1}, g_{i2}, \dots, g_{iD})$, respectively.

2) The neighborhood search: The neighborhood search strategy is directly effective on the efficiency of the local search for the JSSP, and unnecessary and infeasible moves must be eliminated if it is possible. Currently, the most well-known neighborhood structures are all based on the concept of blocks. In the HPTS algorithm, taking into account a balance of the effectiveness and efficiency, we ultimately adopt N_6^1 neighborhood structure, which is introduced by Balas and Vazacopoulos [17].

3) Tabu list: The main task is to avoid the search process turning back to the solutions visited in the previous steps. The elements stored in the tabu list are the attributes of moves, rather than the attributes of solutions for the considered problem. The main purpose of using this attributive representation is to reduce the computational cost. In HPTS, the size of the list is set the same as the swarm size. During the algorithm running, the tabu list will be updated dynamically.

4) Fitness function: Fitness function is usually used as the performance evaluation of particles in the swarm, which is represented to map an original objective function value to a fitness function that represents relative superiority of particles. In HPTS, each particle's fitness function is expressed by the makespan of the corresponding schedule.

5) Cooling schedule: The SA process can be controlled by the cooling schedule. In general, the cooling schedule is specified by several parameters and/or methods, namely the initial temperature T_0 , the rule designated how to lower the temperature, and the termination condition. In proposed algo-

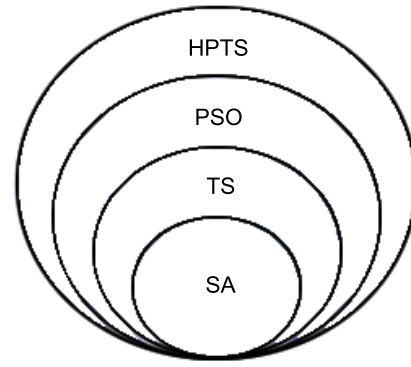


Fig. 1. The structure of HPTS.

rithm, the initial temperature T_0 is set by $T_0 = \Delta f_{max}$, where Δf_{max} is the maximal difference in fitness values between any two neighboring solutions. We specify the temperature with $T_k = B \times T_{k-1}$ during the k th epoch ($k = 1, 2, 3, \dots$), where B is a parameter called decreasing rate and has a value less than 1.

6) Termination criterion: If one of the following conditions is satisfied: whether the algorithm stops when the algorithm has performed a given total number of iterations, or the elite solution stack has been exhausted, or the solution is proved to be optimal.

B. The Process of Proposed Algorithm

In the HPTS, three different operations are defined: PSO operation, TS operation and SA operation. PSO combines local search (by self-experience) with global search (by neighboring experience), achieving a high search efficiency. TS uses a memory function to avoid being trapped at a local minimum. It has emerged as an effective algorithmic approach for the JSSP. This method can also be referred to as calculation of the horizontal direction. SA employs certain probability to avoid being trapped in a local optimum and the search process can be controlled by the cooling schedule (also known as calculation of vertical direction). The groups use a random initialization, which generates initial particle's initial position and velocity in the search space. Each particle's best position is set to the current location. The current position of the corresponding fitness value is calculated for each particle, according to the evolutionary structure. The hybrid HPTS framework can be converted to the traditional PSO that provides initial solution for TS and SA during the hybrid search process. Such hybrid HPTS strategy retains the advantages of TS and SA, which provides a promising methodology to solve JSSP. In addition, HPTS can be applied to many combinatorial optimization problems by simple modification. The description of HPTS is shown in algorithm 1 and algorithm 2. The process of implementing the framework is as follows:

Step 1: Initialize a swarm of particles with random position and velocities in the D-dimensional problem space.

Step 2: For each particle, evaluate the desired optimization fitness function.

Step 3: Individual particle's fitness value are with D-dimensional problem space. If the current value is better than , then set the position equal to the current value, and the position equal to the current position in the D-dimensional problem space.

Step 4: Compare the fitness evaluation value with the swarm's best fitness value obtained so far. If current value is better than , then reset to the current particle's fitness value.

Step 5: Change the velocity and position of the particle according to Eq.5 and Eq.6, respectively.

Step 6: Loop to step (2) until a termination criterion is met, usually a sufficiently good fitness or a specified number of generations.

Algorithm 1 PSO

- 1: **while** the maximum of generation is not met **do**
 - 2: Generation++;
 - 3: Generate next swarm;
 - 4: Find a new local optimum (*gbest*) and a global optimum (*pbest*);
 - 5: Update *gbest* and *pbest*;
 - 6: **end while**
-

Algorithm 2 TS (SA)

- 1: Set iteration *iter* = *iter* + 1, generate neighbors of the current solution *s** by a neighborhood structure. If the *s** is optimal, then stop;
 - 2: Select the best neighbor which is not tabu or satisfies the aspiration criterion, and store it as the new current solution *s**, update the tabu list;
 - 3: **for** *gbest* particle *s* of swarm **do**
 - 4: Temperature $T_k = T_0$;
 - 5: **while** $T_k \geq T_{end}$ **do**
 - 6: Generate a neighborhood solution *s** from *s* by pair-exchange method;
 - 7: Compute the fitness of *s**;
 - 8: Evaluate *s**, $\Delta = f(s^*) - f(s)$;
 - 9: **if** $\min[1, \exp(-\Delta/T_k)] < \text{random}[0,1]$ **then**
 - 10: Accept *s**;
 - 11: Update the best solution found so far if possible;
 - 12: **end if**
 - 13: **end while**
 - 14: $T_k = B * T_{k-1}$ (weight value parameters are included in B);
 - 15: **end for**
 - 16: If *iter* ≤ *improveiter* then go to loop;
 - 17: If a termination criterion is satisfied then stop. Otherwise 'pop' a solution on top of the solution stack *L*, shift the solution to active schedule and install the active solution as the current solution *s**, set *iter* = 0, and empty tabu list. Go to step 2;
-

TABLE I
THE SETTINGS OF THE CONTROL PARAMETERS.

Iterations	The number of particles	Inertia weight
$K = 300$	$N = 400$	$w_{max} = 0.4, w_{min} = 0.2$
Acceleration	Initial temperature	Cooling parameters
$c_1 = c_2 = 0.2$	$T_0 = 10$	$\alpha = 0.97$

TABLE II
THE EXPERIMENTAL RESULTS.

Problem	Size	BKS	HGA	MPSO	HPSO	HPTS
FT06	6 × 6	55	55	55	55	55
FT10	10 × 10	930	930	930	930	930
FT20	20 × 5	1165	1165	1165	1178	1165
LA01	10 × 5	666	666	666	666	666
LA02	10 × 5	655	655	655	655	655
LA03	10 × 5	597	597	597	597	597
LA04	10 × 5	590	590	590	590	590
LA05	10 × 5	593	593	593	593	593
LA06	15 × 5	926	926	926	926	926
LA07	15 × 5	890	890	890	890	890
LA08	15 × 5	863	863	863	863	863
LA09	15 × 5	951	951	951	951	951
LA10	15 × 5	958	958	958	958	958
LA11	20 × 5	1222	1222	1222	1222	1222
LA12	20 × 5	1039	1039	1039	1039	1039
LA13	20 × 5	1150	1150	1150	1150	1150
LA14	20 × 5	1292	1292	1292	1292	1292
LA15	20 × 5	1207	1207	1207	1207	1207
LA16	10 × 10	945	945	945	945	945
LA17	10 × 10	784	784	784	784	784
LA18	10 × 10	848	848	848	848	848
LA19	10 × 10	842	842	842	842	842
LA20	10 × 10	902	902	902	907	902
LA21	15 × 10	1046	1046	1046	1046	1046
LA22	15 × 10	927	927	927	927	927
LA23	15 × 10	1032	1032	1032	1032	1032
LA24	15 × 10	935	953	941	938	935
LA25	15 × 10	977	986	977	977	977
LA26	20 × 10	1218	1218	1218	1218	1218
LA27	20 × 5	1235	1256	1239	1236	1236
LA28	20 × 5	1216	1232	1216	1216	1216
LA29	20 × 5	1152	1196	1173	1164	1166
LA30	20 × 5	1355	1355	1355	1355	1355
LA31	30 × 10	1784	1784	1784	1784	1784
LA32	30 × 10	1850	1850	1850	1850	1850
LA33	30 × 10	1719	1719	1719	1719	1719
LA34	30 × 10	1721	1721	1721	1721	1721
LA35	30 × 10	1888	1888	1888	1888	1888
LA36	15 × 15	1268	1279	1278	1269	1268
LA37	15 × 15	1397	1408	1411	1401	1399
LA38	15 × 15	1196	1219	1208	1208	1201
LA39	15 × 15	1233	1246	1233	1240	1233
LA40	15 × 15	1222	1241	1225	1226	1222

IV. EXPERIMENTAL RESULTS

To illustrate the effectiveness of the HPTS algorithm for JSSP to minimize the makespan, 43 instances taken from OR-Library [18] as test benchmarks to test our new proposed algorithm. In the 43 instances, FT06, FT10, and FT20 were designed by Fisher and Thompson [19]. Instances LA01-LA40 that were designed by Lawrence [20].

The proposed approach was coded in MATLAB programming language and run on a 2.27 GHz Intel(R) Core(TM)2 Duo CPU, RAM 4GB personal computer. Each instance is executed for 10 runs. The algorithm ran with the settings of the control parameters which can be shown in Tabel I. The proper

values of these parameters were experimentally determined in a pre-processing phase. Specifically, the performance of the proposed HPTS was compared against that of Goncalves' HGA [5], Tsung-Lieh Lin's MPSO [2], Wei-Jun Xia's HPSO [9].

Due to the stochastic behavior of these algorithms, and the fact that none of them has a natural termination point, it was decided to run the algorithms for the same fix time duration and report the best solution obtained after this time has elapsed. The benchmarks range from small size instances with 6 jobs and 6 machines to large size instances with 30 jobs and 10 machines.

In Tabel II, *Problem* denotes the JSSP problems, *Size* represents the Problem size n jobs on m machines. *BKS* represents the well-known optimal solutions in the benchmark. Among 43 instaces, HPTS can find the well-known optimal solutions with 40 instances that is much better than HGA (33), MPSO (36), and HPSO (35). Moreover, in about 93 % of the instances, and the deviation of the minimum found makespan from the well-known optimal solutions is only on average 0.4 %. The proposed algorithm yields a significant improvement in solution quality with respect to almost all the other algorithms, except for the HGA, MPSO and HPSO approaches that has a better performance in the LA29, LA36, LA37, LA38, LA40 instances mainly. For instances LA24, LA25, LA27, LA28, LA29, LA36, LA37, LA38, LA39, LA40, the deviation between the best minimum founded solution and the best known solution are all less than the deviation results of HGA. HPTS can obtain better solution for instances LA24, LA27, LA29, LA36, LA37, LA38, LA40 than MPSO. Except for LA29, HPTS also can obtain better solution (LA24, LA36, LA37, LA38, LA39, LA46) than HPSO.

Obviously, the experimental results show that HPTS is more efficient than other existing discrete particle swarm optimization and genetic algorithms, respectively. The superior results also indicate the successful incorporation of the improved PSO, TS and SA, which facilitates the escape from local minimum points and increases the possibility of finding a better solution. Therefore, the computational results show that the proposed algorithm could obtain the high-quality solutions within relatively short computation time. So, it is a robust, fast and simply hybrid optimization algorithm.

V. CONCLUSION

Although there are many literatures on classical PSO, the P-SO for JSSP does not have rich literatures. In this paper, a new hybrid heuristic HPTS, based on Particle Swarm Optimization (PSO), Tabu Search (TS) and Simulated Annealing (SA), is presented. We have applied the HPTS algorithm to JSSP in minimizing makespan. The experimental results show the effectiveness of the proposed approach. Moreover, our HPTS algorithm outperforms HGA, MPSO, and HPSO algorithms in some solution quality thoroughly. For 6 of 43 instances, new upper bounds among the unsolved problems are found in a short time in HPTS. It can achieve the optimal value in the search space quickly with smaller population size, making use

of its better local searching ability to get the final optimal solution at the end of evolution.

In addition, we have proposed a new algorithm structure: three-tier structure of hybrid algorithm, which brings the new research direction and inspiration for the hybrid algorithm. To improve the algorithm performance, we plan to apply parallel and distributed computing to the HPTS algorithm.

ACKNOWLEDGMENT

This paper is supported by the research fund JSPS KAKENHI (20240003, 21300054). Xue-Feng Zhang is sponsored by the China Scholarship Council (CSC).

REFERENCES

- [1] M. Garey, D. Johnson, R. Sethi, "The complexity of flow shop and job shop scheduling," *Mathematics of Operations Research*, 1, 1976, pp. 117-129.
- [2] L. Tsung-Lieh, H. Shi-Jinn, "An efficient job-shop scheduling algorithm based on particle swarm optimization," *Expert Systems with Applications*, 37(2010), pp.2629-2636.
- [3] J. Adams, E. Balas, D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Management Science*, 34(1988), pp.391-401.
- [4] E. Nowicki, C. Smutnicki, "An advanced tabu search algorithm for the job shop problem," *Journal of Scheduling*, 8(2005), pp.145-159.
- [5] J.F. Goncalves, J.J.D.M. Mendes, M.G.C. Resende, "A hybrid genetic algorithm for job shop scheduling," *European Journal of Operational Research*, 167(2005), pp.77-95.
- [6] R.K. Sursh, A. Albrecht, K.C. Wong, "Pareto archived simulated annealing for job shop scheduling with multiple objectives," *The International Journal of Advanced Manufacture Technology*, 29(2005), pp.184-196.
- [7] H.W. Ge, L. Sun, Y.C. Liang, "An effective PSO and AIS-based hybrid intelligent algorithm for job-shop scheduling," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 38(2008), pp.358-368.
- [8] A. Udomsakdigool, V. Kachitvichyanukul, "Multiple colony ant algorithm for job-shop scheduling," *International Journal of Production Research*, 46(2008), pp.4155-4175.
- [9] X. Weijun, W. Zhiming, "A Hybrid particle swarm optimization approach for the job-shop scheduling problem," *Int J Manuf Technol*, 29(2006), pp. 360-366.
- [10] V. Fevrier, M. Patricia, C. Oscar, M. Oscar, "A New Evolutionary Method with a Hybrid Approach Combining Particle Swarm Optimization and Genetic Algorithms using Fuzzy Logic for Decision Making," *IEEE Congress on Evolutionary Computation*, 2008, pp. 1333-1339.
- [11] V. Fevrier, M. Patricia, C. Oscar, "An improved evolutionary method with fuzzy logic for combining Particle Swarm Optimization and Genetic Algorithms," *Appl. Soft Comput*, 2011, 11(2): 2625-2632.
- [12] J. Kennedy, R. Eberhart, "Particle swarm optimization," In *Proceedings of IEEE international conference on neural network*, 1995, pp.1942-1948.
- [13] R. Eberhart, J. Kennedy, "A new optimizer using particle swarm theory," In: *Proceedings of Sixth International Symposium on Micro Machine and Human Science, Japan*, 1995, pp. 39-43.
- [14] D.Y. Sha, L. Hsinghung, "A particle swarm optimization for multi-objective flowshop scheduling," *Original Article Int J Adv Manuf Technol*, (45) 2009, pp. 749-758.
- [15] Y. Shi, R. Eberhart, "Empirical study of particle swarm optimization," In: *Proceedings of Congress on Evolutionary Computation*, 1999, pp.1945-1950.
- [16] J. Kennedy, "The particle swarm: social adaptation of knowledge," In: *Proceedings of IEEE International Conference on Evolutionary Computation*, 1997, pp.303-308.
- [17] E. Balas, A. Vazacopoulos, "Guided local search with shifting bottleneck for job shop scheduling," *Management Science*, (44)1988, pp. 62-75.
- [18] J.E. Beasley, "Or-library: Distributing test problems by electronic mail," *Journal of the Operational Research Society*, (14)1990, pp. 1069-1072.
- [19] H. Fisher, G.L. Thompson, "Probabilistic learning combinations of local job shop scheduling rules," *Prentice-Hall, NJ: Englewood Cliffs*, 1963, pp. 225-251.
- [20] S. Lawrence, "An experimental investigation of heuristic scheduling techniques," In *supplement to resource constrained project scheduling*, GSIA, Pittsburgh, PA: Carnegie Mellon University, 1984.