Proceedings of the International Conference on
Information Technologies (InfoTech-2017)
20-21 September 2017, Bulgaria

ENHANCED PSO ALGORITHM FOR SINGLE-OBJECTIVE FLEXIBLE JOB SHOP SCHEDULING PROBLEMS

Asen Tochev, Vassil Guliashki

Institute of Information and Communication Technologies - BAS
E-mails: tochevassen@yahoo.com ,vggul@yahoo.com
BULGARIA

Abstract: The paper represents an enhanced new algorithm called FPSO-FJSSP, based on Particle Swarm Optimization (PSO) technique for the combinatorial flexible job shop scheduling problem (FJSSP). Five test examples from literature sources are tested by the new algorithm. In all cases the optimal solution has been found. The performance of the FPSO-FJSSP algorithm is illustrated on one of the test examples. The obtained results are encouraging. A variant of the new algorithm can be applied to solve multiple objective FJSSP.
Key words: FJSSP, Particle swarm optimization, Heuristics

1. INTRODUCTION

The flexible job shop scheduling problem (FJSSP) belongs to the class of NP-hard optimization problems. The exact methods need great computational time to find out the optimal solution. That's why many heuristics and metaheuristics are developed to solve the FJSSP. One well known heuristic, which is used by many researchers for solving FJSSP is the Particle Swarm Optimization (PSO). This is a population based stochastic optimization technique, inspired by the food searching process by flocks of birds, shoals of fish and swarms of insects.

PSO has many similarities with evolutionary optimization techniques such as genetic algorithms (GA). The search is initialized with a population of uniformly distributed solutions, which are updated during the generations/iterations. But, unlike GA, PSO has no evolutionary operators such as crossover and mutation. In PSO, the potential solutions, called particles fly in the feasible region of the task after the current optimal particle. Each particle remembers its coordinates in the feasible region that are associated with the best solution (fitness) that it has achieved to date.

(The value of fitness is also remembered.) This value is denoted by *pbest*. Another "best" value, which is used in optimization by swarm particles is the best value obtained so far by any particle in the vicinity of a specific particle considered. This location is indicated by *lbest*. If the particle considered the entire population as its destination, the best value is a global best and it means a *gbest*.

The concept of PSO consists in changing the speed (acceleration) of each particle to its *pbest* and *lbest* locations (local version of PSO) at each iteration. Acceleration is weighted by random member, with separate random numbers.

In recent years, PSO has been successfully applied in many areas of research and applications. It has been demonstrated that PSO obtained better results (mainly in terms of convergence of the search process) with a faster and cheaper way compared to other evolutionary methods.

In [22] is offered a hybrid algorithm combining optimization by swarm particles and simulated annealing procedure to solve the problem in a flexible production schedule.

In [23] is offered a hybrid algorithm of PSO and TS to solve the flexible job shop scheduling problem.

In [13] is suggested a novel hybrid algorithm, where PSO was used to produce a swarm of high quality candidate solutions, while TS was used to obtain a near optimal solution around the given good solution. The computational results have proved that the proposed hybrid algorithm is efficient and effective for solving FJSP, especially for the problems with large scale.

In [18] is proposed a novel initialization method based on the improved Kacem assignments scheme. Experimental results indicate that this method is efficient and competitive compared to some existing methods.

In [24] is proposed a new effective multi-objective approach based on the hybridization of the particle swarm optimization (PSO) and local search algorithm of variable neighborhood search (VNS) to solve the FJSP for minimizing the makespan, the maximal machine workload, and the total workload of machines.

Discrete PSO (DPSO) is a modified version of PSO which applies discrete or qualitative distinction between variables. Some authors consider the DPSO as a good tool solving combinatorial optimization problems due to its easy implementation, simple structure and robustness (see [15, 19]). In [14] is stated that DPSO algorithms could be classified in five categories. In [10] is proposed a DPSO algorithm with particles including binary variables.

Many studies pointed out that the use of PSO to solve problems with continuous variables in a convex feasible domain is characterized by its high convergence. Unfortunately in the case of combinatorial optimization problems as JSSP the results concerning the convergence are not so good. This is due to the fact, that the moving a particle is performed by reassigning the operations on machines using stochastic operators or logistic functions (see [5, 8]). Assigning the operations on randomly (with an uniform distribution) chosen machines, the probability p to assign an operation on the optimal machine in case that there are $m \geq 3$ machines and only one

machine process this operation in optimal time is $p \leq ½$. The probability P, all the operations to be reassigned on the optimal machines (with the shortest processing times for the corresponding operations) is small. This probability decreases exponential according to the number of machines and the operations number. In the worst case it could be expressed as: $P \sim O(\left(\frac{1}{m-1}\right)^l)$, where m is the number of machines and l is the total sum of all operations in all jobs.

There are many studies on solving the flexible job shop scheduling problem (FJSSP) based on heuristic approaches. Comprehensive surveys in this area are given in [11, 12, 20]. Important methods for this class of problems have been proposed in [2, 3, 6]. In [17] a mathematical model for FJSSP has been developed. The objective function is maximizing the total profit while meeting some constraints.

In this paper is proposed a new fast PSO algorithm, called FPSO-FJSSP, which is based on strategic reassignment of operations on the optimal machines. The neighborhood of a particle is explored by strategic rearrangement of operations. After each rearrangement the rearranged operations are reassigned on the machines with earliest completion time (ECT), which if it is possible coincide with the optimal machines.

2. PROBLEM FORMULATION AND A SET OF TEST EXAMPLES

The formulation of the problem is the following: There is given a set of n jobs: J_1, ..., J_n, which have to be performed on m machines M_1, ... , M_m. For each job there is given operations consequence: $J_i = (O_{i,1}, ..., O_{i,n_i})$, where n_i is the number of operations for the corresponding job, $i = 1,..., n$; $\sum_{i=1}^{n} n_i = l$, where l is the number of all operations O_{i,n_i}. This problem is called FJSSP (flexible job shop scheduling problem). The FJSPs (flexible job shop problems) are an extension of classical JSPs (job shop problems) taking into account the production flexibility. Unlike the classical JSP where each operation is processed on a predefined machine, each operation O_{ij} in the FJSP can be processed on more than one machine. The possible machines are denoted by M_k, where $M_k \in M_{ij}$, $M_{ij} \subseteq M$. If $M_{ij} \subset M$ for at least one operation, then there is a partial flexibility of JSSP (PFJSSP); while if $M_{ij} \equiv M$ for each operation, there is a total flexibility of JSSP (TFJSSP).

A set of benchmark test problems is given in [1]. The following set of test examples is used in this study: 1) M2J2O4 (two machines, two jobs and four operations) [4], M3J2O5 [16], M4J3O8 [18], M3J3O9 [21], M5J4O12 [9].

3. SIMPLE DESCRIPTION OF THE NEW FPSO-FJSSP ALGORITHM

3.1. Concepts

Definition: By *Cmax* is denoted the *makespan*, i. e. the maximum completion time for all operations in the schedule under consideration.

3.1.1 Calculate for each job J_i the sum S_i, $i = \overline{1, n_i}$ of the optimal times for all operations in this job.

$$S_i = \sum_{j=1}^{n_i} \min(t_{j_1}, \dots, t_{j_m}), \quad \text{for the job } J_i,\ i = \overline{1, n};$$

where n_i is the number of operations in J_i. Put a weight $w_i = S_i$ to each job J_i, $i = \overline{1, n}$;. The jobs with higher weight should be included in the schedule as far as possible to the left (to the beginning of the schedule) in order the jobs with lower weight to be included in the schedule using the idle times of machines and the makespan to be minimal.

3.1.2 If two jobs J_i and J_j have the same weight, we first assign on machines the operations of the job having a lower index.

3.1.3 The operations in the schedule should be arranged in subsets according the corresponding jobs. The final schedule is called schedule in a canonical form.

3.1.4 To each machine M_k, $k = \overline{1, m}$; assign a number R_k corresponding to the number of times this machine proceeds an operation from all the jobs in an optimal time (using the processing time table). Between two machines, higher priority has that one, which has smaller R_k. A machine M, which proceeds never in optimal time, does not take part in such comparisons.

3.1.5 Assign the operations from the first job in the schedule on the optimal machines, taking into account the machine priorities. If there are more than one machine M with the same weight, assign the correspondent operation on the machine with earliest completion time, and if the completion times are equal, assign it on the machine with a smaller index.

3.1.6 Continue to assign operations from the other jobs according the jobs' weights. Each job should be assigned to an optimal machine if possible. If there is more than one optimal machine, a higher priority machine is selected. Calculate the completion times of all machines by checking the idle times (pauses) of the machines, if they can be used to process the current operation. Finally, operations are assigned to the machine, which has the least completion time.

3.2. Stopping criteria

– Global number of iterations;

– There is no improvement of the makespan *Cmax* at the current iteration;

– The best makespan in the population is equal to the maximal sum among the sums of optimal times for each job;

– The best makespan in the population is equal to the sum of optimal times for all jobs, divided to the number of machines;

3.3. Generating the initial population

The initial population is calculated in the following manner: The current schedule is composed on the base of l generated random numbers $r \in (0, 1)$. The interval (0, 1) is divided in n parts, corresponding to each job. Each random number determines the current job, where one operation is taken from to be put in the

schedule. Taking the current operation from the chosen job, the constraint for the strong operations order/consequence is hold strictly. The operations are assigned on the machines according the rule on the first free machine, and if there are several free machines at one at the same time, the machine with the smallest index is chosen. In case the current job is completed (i.e. all its operations are included in the schedule), continue with the non-completed job, having smallest index.

3.4. Algorithm FPSO-FJSSP

The new algorithm FPSO-FJSSP is organized in the following steps:

Step 1. Insert a number *npart* of all particles, included in the population.

Step 2. Generate the initial population of *npart* particles by the procedure described in **3.3**.

Step 3. Set iteration counter *icount* = 1;

Step 4. Iteration: Set the counter *ipart* = 1.

Step 5. Bring the current particle (schedule) from the population in canonical form.

Step 6. Move all operations from the job, where the bottleneck is occurred, as a subset to the left before the preceding job in the schedule. In case this is a step backwards, then don't move these operations, but the operations from the last moved job. (Here is applied a Tabu strategy: the steps backwards are forbidden.) Reassign the moved operations and all operations after them in the schedule on the machines with earliest completion times, and if the completion times are equal, assign the corresponding operation on the machine with a smaller index.

Step 7. Repeat **Step 6** until no more movements are possible.

Step 8. Save the schedule with the best obtained makespan *Cmax*.

Step 9. If *ipart* = *npart*, set *icount* = *icount* +1.

Step 10. check if the Stopping criteria, described in **3.2** are met. In this case go to **Step 11**. Otherwise set *ipart* = *ipart* +1. If *ipart* > *npart* go to **Step 11**, otherwise go to **Step 4**.

Step 11. In the best obtained schedule, the last operation in the bottleneck machine is attempted to be assigned to another machine with the aim to improve its completion time and the makespan.

Step 12. STOP.

4. ILLUSTRATIVE EXAMPLE

The following illustrative example is considered (see [9]):

Job	Oper.	M_1	M_2	M_3	M_4	M_5	Job	Oper.	M_1	M_2	M_3	M_4	M_5
J_1	O_{11}	2	5	4	1	2	J_3	O_{31}	9	2	6	7	9
	O_{12}	5	4	5	7	5		O_{32}	6	1	2	5	4
	O_{13}	4	5	5	4	5		O_{33}	2	5	4	2	4
J_2	O_{21}	2	5	4	7	8		O_{34}	4	5	2	1	5
	O_{22}	5	6	9	8	5	J_4	O_{41}	1	5	2	4	12
	O_{23}	4	5	4	54	5		O_{42}	5	1	2	1	2

The sums of optimal processing times for the correspondent jobs are:

$\Sigma(J_1) = 9$, $\Sigma(J_2) = 11$, $\Sigma(J_3) = 10$, $\Sigma(J_4) = 2$. According this values the priorities for performing the jobs are ordered as follows: J_2, J_3, J_1, J_4.

Machine M_1 performs optimally 6 of the operations. Machine M_2 performs optimally 3 of the operations. Machine M_3 performs optimally 2 of the operations. Machine M_4 performs optimally 5 of the operations. Machine M_5 performs optimally 1 operation. Hence the priorities of machines are ordered as follows: M_5, M_3, M_2, M_4, M_1.

By convention the number used to denote the time means the end of the current time unit (e.g. of a second, hour, day, month or year). The obligatory constraints about the order/consequence of operations in the jobs are hold. In the following tables the first line is used for the starting times, the second for the machines numbers where the operations are assigned on, the third line is used for the jobs numbers, the fourth line is used for the operations numbers and the last (fifth) line is used for the final time.

The following solution from the initial population is used to be improved:

Schedule 0 (S_0):

0	1	0	4	0	11	2	9	7	13	11	15
1	2	3	4	5	1	2	3	2	5	3	4
4	4	1	1	3	1	2	3	2	2	3	3
1	2	1	2	1	3	1	2	2	3	3	4
1	2	4	11	9	15	7	11	13	**18**	15	16

The Schedule 0 has *Makespan*(S_0) = 18.

This schedule is transferred to its canonical form by moving the operations O_{22} and O_{23} to the left next to the operation O_{21} (with re-assigning them on other, if possible, optimal machines), the operation O_{32} is moved to the right (with re-assigning on an optimal machine) next to the operation O_{33} and operation O_{31} is moved to the right to its successor operations without re-assigning because it starts from time 0. Operations O_{33} and O_{34} are also re-assigned on optimal machines following the machine priorities. In this way the new schedule with arrangement (J_4, J_1, J_2, J_3) is obtained:

Schedule 1 (S_1):

0	1	0	4	11	2	7	12	0	9	11	13
1	2	3	4	1	2	5	3	5	2	4	4
4	4	1	1	1	2	2	2	3	3	3	3
1	2	1	2	3	1	2	3	1	2	3	4
1	2	4	11	15	7	12	**16**	9	10	13	14

The Schedule 1 has *Makespan*(S_1) = 16. An improvement of the makespan is obtained. The bottleneck is in J_2. It is moved on block to the left before J_1 (Step 6) with re-assignment of all operations in J_2, J_1 and J_3 on, if possible, optimal machines. In this way, the new schedule with arrangement (J_4, J_2, J_1, J_3) is obtained:

Schedule 2 (S_2):

0	1	1	3	8	0	2	6	0	6	7	9
1	2	1	5	3	4	2	4	3	2	4	4
4	4	2	2	2	1	1	1	3	3	3	3
1	2	1	2	3	1	2	3	1	2	3	4
1	2	3	8	**12**	1	6	10	6	7	9	10

The Schedule 2 has *Makespan*(S_2) = 12. An improvement of the makespan is obtained. The bottleneck is again in J_2. It is moved on block to the left before J_4 (Step 6) with re-assignment of all operations in J_2, J_4, J_1 and J_3 on, if possible, optimal machines. In this way the new schedule with arrangement (J_2, J_4, J_1, J_3) is obtained:

Schedule 3 (S_3):

0	2	7	2	3	0	4	8	0	8	9	12/11
1	5	3	1	2	4	2	4	3	2	1	4/3
2	2	2	4	4	1	1	1	3	3	3	3
1	2	3	1	2	1	2	3	1	2	3	4
2	7	11	3	4	1	8	12	6	9	11	**13**

The Schedule 3 has *Makespan*(S_3) = 13. An improvement of the makespan is not obtained. The bottleneck is in J_3. It is moved on block to the left before J_1 (Step 6) with re-assignment of all operations in J_3 and J_1 on, if possible, optimal machines. In this way the new schedule with arrangement (J_2, J_4, J_3, J_1) is obtained:

Schedule 4 (S_4):

0	2	7	2	3	0	6	7	9	0	3	8
1	5	3	1	2	3	2	4	4	4	1	1
2	2	2	4	4	3	3	3	3	1	1	1
1	2	3	1	2	1	2	3	4	1	2	3
2	7	11	3	4	6	7	9	10	1	8	**12**

The Schedule 4 has *Makespan*(S_4) = 12. An improvement of the makespan is not obtained. The bottleneck is in J_1. The movement of J_1 on block to the left before J_3 is forbidden (Tabu strategy: the steps back are forbidden). Hence J_3 is moved on block to the left before J_4 (Step 6) with re-assignment of all operations in J_3, J_4 and J_1 on, if possible, optimal machines. In this way the new schedule with arrangement (J_2, J_3, J_4, J_1) is obtained:

Schedule 5 (S_5):

0	2	7	0	6	7	9	2	3	0	3	8
1	5	3	3	2	4	4	1	2	4	1	1
2	2	2	3	3	3	3	4	4	1	1	1
1	2	3	1	2	3	4	1	2	1	2	3
2	7	11	6	7	9	10	3	4	1	8	**12**

The Schedule 5 has *Makespan*(S_5) = 12. An improvement of the makespan is not obtained. The bottleneck is in J_1. It is moved on block to the left before J_4 (Step 6) with re-assignment of all operations in J_1 and J_4 on, if possible, optimal machines. In this way the new schedule with arrangement (J_2, J_3, J_1, J_4) is obtained:

Schedule 6 (S_6):

0	2	7	0	6	7	9	0	1	5	2	3
1	5	3	3	2	4	4	4	2	1	1	4
2	2	2	3	3	3	3	1	1	1	4	4
1	2	3	1	2	3	4	1	2	3	1	2
2	7	**11**	6	7	9	10	1	5	9	3	4

The Schedule 6 has *Makespan*(S_6) = 11. An improvement of the makespan is obtained. The bottleneck is in J_2. It corresponds to the sum of optimal processing times for this job. No more improvements are possible. This is the optimal solution.

In the similar way are obtained the optimal schedules for the other examples as follows:

M2J2O4: The makespan *Cmax* = 66.
Optimal schedule:

0	45	0	37
1	1	2	2
2	2	1	1
1	2	1	2
45	**66**	37	61

M3J2O5: The makespan *Cmax* = 53.
Optimal schedule:

0	20	38	20	30
1	2	2	1	3
2	2	2	1	1
1	2	3	1	2
20	38	**53**	30	48

M4J3O8: The makespan *Cmax* = 12.
Optimal schedule:

0	4	8	0	2	6	0	6/8
3	1	3	1	4	4	2	2/4
1	1	1	2	2	2	3	3
1	2	3	1	2	3	1	2
4	8	**12**	2	6	8	6	11

M3J3O9: The makespan *Cmax* = 46.
Optimal schedule:

0	8	19	8	19	29	18	28	37
1	2	3	1	2	3	1	2	3
3	3	3	1	1	1	2	2	2
1	2	3	1	2	3	1	2	3
8	19	29	18	28	37	26	35	**46**

5. CONCLUSION

Many heuristics and metaheuristics are developed to solve the FJSSP. It could be seen that the convergence for the combinatorial problems is smaller in comparison to that one for problems with continuous variables. The particle swarm optimization (PSO) is one of the most popular evolutionary techniques applied for the FJSSP, but its convergence is not very high.

In this paper is presented a new enhanced FPSO-FJSSP algorithm, based on strategic rearrangement of the operations and of reassigning them on the optimal machines having earliest completion times. Five test examples are used. In all the cases the optimal schedules are obtained on the first iteration.

The obtained results are very encouraging and show that the convergence of the new algorithm is very high, and the quality of the obtained solutions is very good. The created algorithm will be tested on a greater set of test examples including examples with larger size – up to 20 machines, 20 jobs and 100 operations. A comparison with other evolutionary and exact algorithms and methods will be done.

REFERENCES

[1] Behnke D., Geiger M. J., "Test Instances for the Flexible Jop Shop Scheduling Problem with Work Centers", Arbeitspapier/Research report, RR-12-01-01, January 2012, ISSN 2192-0826, http://edoc.sub.uni-hamburg.de/hsu/volltexte/2012/2982/pdf/RR-12-01-01.pdf.

[2] Brandimarte, P. (1993) Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41(3):157–183, 1993.

[3] Chan, F.T.S., T.C. Wong, and L.Y. Chan. (2006) Flexible job-shop scheduling problem under resource constraints. *International Journal of Production Research*, 44(11), 2006: 2071–2089.

[4] Fattahi P., M. S. "Mehrabad, and F. Jolai. Mathematical Modeling and Heuristic Approaches to Flexible Job Shop Scheduling Problems". *Journal of Intelligent Manufacturing*, 18(3):331–342, 2007.

[5] Ge H., W. Du, F. Qian, A Hybrid Algorithm Based on Particle Swarm Optimization and Simulated Annealing for Job Shop Scheduling, http://www.paper.edu.cn.

[6] Gutierrez, C. and I. Garcia-Magarino (2011) Modular design of a hybrid genetic algorithm for a flexible job-shop scheduling problem. *Knowledge-Based Systems*, 24(1):102—112, 2011.

[7] Huang S., N. Tian, Y. Wang, Z. Ji, Multi-objective flexible job-shop scheduling problem using modified particle swarm optimization, Huang et al. Springer Plus (2016) 5:1432, DOI: 10.1186/s.40064-016-3054-z.

[8] Jarboui B., S. Ibrahim, P. Siarry, A. Rebai, A combinatorial particle swarm optimization for solving permutation flowshop problems, Computers and Industrial Engineering 54 (2008) 526-538.

[9] Kacem I., S. Hammadi, and P. Borne. Pareto-Optimality Approach for Flexible Job-Shop Scheduling Problems: Hybridization of Evolutionary Algorithms and Fuzzy Logic. *Mathematics and Computers in Simulation*, 60(3-5):245–276, 2002.

[10] Kennedy J. and R. C. Eberhart, A discrete binary version of the particle swarm algorithm, pp. 4104-4108,
https://wenku.baidu.com/view/d52bd6c108a1284ac9504307.html,
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.454.6101&rep=rep1&type=pdf

[11] Kirilov L., V. Guliashki, K. Genova, V. Angelova, (2016) "An Overview of Multiple Objective Job Shop Scheduling Techniques", *JÖKULL Journal*, Impact factor: 1.604, ISSN: 0449-0576, Vol. 66, No 2; Feb. 2016, pp. 172-206

[12] Kirilov L., V. Guliashki, K. Genova, (2016) "*Multicriteria Decision Making in Manufacturing Scheduling*", (in Bulgarian), Editor: Prof. Dr. Ivan Mustakerov, ISBN: 978-954-552-074-7, Publisher „Education Ltd“, 2016.

[13] Li J., Q. Pan, S. Xie, B. Jia and Y. Wang, A hybrid particle swarm optimization and tabu search algorithm for flexible job-shop scheduling problem, International Journal of Computer Theory and Engineering, Vol. 2, No. 2 April, 2010, 1793-8201.

[14] Muthuswamy S., S. Lam, DISCRETE PARTICLE SWARM OPTIMIZATION FOR THE ORIENTEERING PROBLEM, International Journal of Industrial Engineering, 18(2), 92-102, 2011.

[15] Pan, Q-K., Tasgetiren, M. F., & Liang, Y-C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Computers & Operations Research*, 35:2807–2839.

[16] Saidi-Mehrabad M., P. Fattahi, Flexible job shop scheduling with tabu search algorithms, *Int. J. Adv. Manuf. Technol.* (2007) 32: 563-570, DOI 10.1007/s00170-005-0375-4.

[17] Sayedmohammadreza Vaghefinezhad, Kuan Yew Wong, A Genetic Algorithm Approach for Solving FJSSP (2012).

[18] Tang J., G. Zhang, B. Lin, B. Zhang, A Hybrid Algorithm for Flexible Jop-shop Scheduling Problem, SciVerse ScienceDirect, Procedia Engineering 15 (2011) 3678-3683, www.sciencedirect.com, www.elsevier.com/locate/procedia.

[19] Tasgetiren, M. F., Liang, Y-C., Sevkli, M., & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European Journal of Operational Research*, 177:930–947.

[20] Tochev A., Heuristics and Metaheuristics for single- and multi-objective flexible job shop scheduling problems, *Proceedings of the International Conference on Information Technologies, (InfoTech-2016), 20-21 September 2016, Bulgaria.*

[21] Udaiyakumar K.C., M. Chandrasekaran, Application of Firefly Algorithm in Job Shop Scheduling Problem for Minimization of Makespan, 12th Global Congress on Manufacturing and Management, GCMM 2014, ScienceDirect, Procedia Engineering 97 (2014) 1798-1807, www.sciencedirect.com, www.elsevier.com/locate/procedia.

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

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

[24] Zhang G., Lingjie Zhang, Yongcheng Wang and Lihui Wu, An Effective Hybrid Particle Swarm Optimization for Flexible Job Shop Scheduling Problem, *The Open Automation and Control Systems Journal*, 2014, *6*, 1604-1611.