A Hybrid Metaheuristic Algorithm for Flexible Job Shop Scheduling "PSO&TS"

Asen Tochev[1] and Vassil Guliashki[2]

Abstract – **In this paper is presented a new developed hybrid metaheuristic algorithm, designed to solve flexible job shop scheduling problems. The algorithm combines a Particle Swarm Optimization procedure with Tabu search technique, using neighborhoods with one operation movement, as well as a final local search using neighborhoods with one and two operations movement. An illustrative example is presented. The novel algorithm is called "PSO&TS" and is suitable for large sized flexible job shop scheduling problems.**

Keywords – **Flexible Job Shop Scheduling Problem, Particle Swarm Optimization, Tabu Search.**

I. INTRODUCTION

In the job shop scheduling problem (JSSP), there are n jobs that must be processed on a group of m machines. The set of machines is denoted by M. Each job i consists of a sequence of h operations (O_{i1_i}, O_{i2_i}, . . . , O_{ih_i}), where O_{ik_i} (the k_i-th operation of job i) must be processed without interruption on a predefined machine j for p_{ik_ij} time units. The operations O_{i1}, O_{i2}, . . . , O_{ih_i} must be processed one after another in the given order and each machine can process at most one operation at a time.

In this paper we study the flexible job shop scheduling problem (FJSSP). The FJSSPs are an extension of classical JSSP taking into account the production flexibility. Unlike the classical JSSP, where each operation is processed on a predefined machine, each operation O_ik_{ij} in the FJSSP can be processed on one machine M_l out of several machines $M_l \in M_{ij}$, $M_{ij} \subseteq M$. If $M_{ij} \subset M$ for at least one operation, then it is a partial flexibility of JSSP (P-FJSSP); while if $M_{ij} \equiv M$ for each operation, then it is a total flexibility of JSSP (T-FJSSP).

In a flexible job shop, each job i consists of a sequence of h_i operations (O_{i1}, O_{i2}, . . . , O_ih_i). The processing time of operation Oik_i on machine j, $j = 1,...,m$; is $p_{ik_ij} > 0$, where $k = 1,...,h_i$. The FJSSP problem is to choose for each operation Oik_i a machine $M(Oik_i) = Mik_i$ and a starting time sik_i at which the operation must start its processing.

Detailed definition of FJSSP is given (see [1-4]) as follows:

- A set of J independent jobs.
- Each job J_i , $i = 1, ..., n$; can be processed on a given set of machines M_i .
- The O_{i,k_i} represents the k_i-th operation of J_i . The machines set waiting for processing the $O_{i,k}$ noted by $M_j \subset M$.
- We use $p_{i,k_i,j}$ to represent the processing time of $O_{i,k}$ operated on the j-th machine.
- There have two assumptions: a started operation cannot be interrupted; each machine can process only one operation at the same time.

The objective in our paper is to find the minimum time to finish all the operations.

In FJSSP one job can be processed on different machines and the distinct jobs can have different number of operations. In flexible job shop, job J_i has hi operations to be performed in a strong sequence, corresponding to the index consequence 1,..., h; each job/operation is attributed to more than one machine which can perform operations on each of them. In this article we assume that each operation can be performed on at least two machines as alternatives.

This model represents the expansion of the upper JSSP - model. Unlike the previous model, in which each operation can be performed only on a single machine, in this model an operation Oik_i can be performed on a machine M_l from a number of possible machines (machines subgroup). This subgroup is naturally different for each operation. In other words, it is not known in advance, any transaction of which one machine must be fulfilled.

Flexible job shop scheduling problem (FJSSP) can be decomposed naturally into two tasks: 1) object routing (appointment) at which each transaction is assigned to a corresponding machine chosen among the set of machines, on which its processing is possible; 2) subtask scheduling (sequence) where you have to calculate the start times of the appointed operations of all machines to obtain acceptable schedule (which does not violate the technological limitations) and to optimize predefined criteria (objective functions).

JSSP is NP hard, hence FJSSP is also NP hard (see [5]). Its optimal solution is difficult to be obtained even when the objective is to minimize the makespan (the time for processing all the jobs). For this reason, many studies are focused on the development of heuristic procedures for this problem. Well known among them are the particle swarm optimization (PSO) algorithm (see [9, 10]) and the metaheuristic Tabu search (TS) [14-17]. According [18] the job shop instances larger than 15 x 15 have to be solved by heuristics and the Tabu search heuristic is the best known for JSP. In [9] is demonstrated that PSO obtains better results (mainly considering the convergence of the search process) in a faster and cheaper way in comparison to other evolutionary methods. Based on these statements the idea arises, to combine PSO and TS techniques using the advantages of these very successful optimization procedures and to create a hybrid metaheuristic algorithm, designed to solve flexible job shop scheduling problems.

The paper is organized as follows: in section II the

[1]Asen Tochev is with the Institute of Information and Communication Technologies – BAS, "Acad. G. Bonchev" Str. Bl. 2, 1113 Sofia, Bulgaria, E-mail: tochevassen@yahoo.com.

[2]Vassil Guliashki is with the Institute of Information and Communication Technologies – BAS, "Acad. G. Bonchev" Str. Bl. 2, 1113 Sofia, Bulgaria, E-mail: vggul@yahoo.com.

subprocedure PSO is presented; in s. III the subprocedure TS is presented. The phases of the new algorithm are considered in s. IV. The PSO&TS algorithm is presented in s. V. An illustrative example is given in s. VI. In s. VII some conclusions are drawn.

II. Subprocedure PSO

One powerful heuristic is the Particle Swarm Optimization (PSO) . It has been first developed by Eberhart and Kennedy [6-8]. The basic idea in PSO is to imitate the intelligent swarming behavior, observed in flocks of birds, schools of fish, swarms of bees, etc. Each particle in PSO represents one solution and it makes steps from its current position to a new position. This movement is determined as a sum of three vectors: *inertia*, *competition* and *cooperation*. The inertia vector is determined by the current velocity of the particle $v(t)$ weighted by a constant w. In this way the tendency of the particle to maintain its current velocity is reflected. The competition vector links the current position of the particle $x(t)$ to its personal best position found during the search process. This vector is weighted using a uniformly distributed random diagonal matrix $Dr1$. The cooperation vector links the current position of the particle $x(t)$ to the global best position, found by the particles. This vector is weighted using a second uniformly distributed diagonal matrix $Dr2$. It is clear that the cooperation among particles is important for finding the global optimum solution. The inertia and the competition are necessary for the particle to avoid trapping in the local minima.

Suppose that the search space is d-dimensional, and then the i-th particle of the swarm can be represented by a d-dimensional vector, $x_i = (x_{i1}, x_{i2}, ..., x_{ij}, ..., x_{id})$, $d = m * n$. The velocity of the particle can be represented by another vector, $v_i = (v_{i1}, v_{i2}, ..., v_{ij}, ..., v_{id})$. The best solution achieved by the particle is denoted as $p_i = (p_{i1}, p_{i2}, ..., p_{ij}, ..., p_{id})$, and the global best solution/particle is denoted by $g = (g_1, g_2, ..., g_j, ..., g_d)$. The particle updates its velocity and position by means of equation (1) and (2) as follows.

$$v_{i=1} = w*v_i+b*Dr1*(p_i-x_i)+c*Dr2*(g_i-x_i) \quad (1)$$

$$x_i^{(k+1)} = x_i^{(k)} + v_i \quad (2)$$

The appropriate choice of inertial weight w in (1) can provide a balance between global and local research's operating capabilities of the swarm, and resulted in an average less reps to find enough optimum solution. Accelerating constants b and c in equation (1) represent the weight of stochastic acceleration terms that draw every particle i in p_i and g_i directions. In this study b = 1.8 and c = 2.2. Lower values lead to small convergence speed, while higher values result in jerky movements through the search region.

In the last years PSO algorithms have been successfully applied in many research areas for different applications. It is demonstrated – see [9], that PSO obtains better results (mainly considering the convergence of the search process) in a faster and cheaper way in comparison to other evolutionary methods. In [10] another hybrid algorithm, combining particle swarm optimization and a tabu search technique to solve FJSSP is proposed.

III. Subprocedure TS

This is a widely used robust metaheuristic created by Fred Glover (see [11-13]). The main idea of TS is to perform a strategic search procedure, starting from one solution and avoiding cycles that lead to visiting the same solution by means of a memory (more precisely a set of memories), storing some characteristics of solutions or movements (steps in given directions) which are forbidden (tabu) for a certain number of iterations. In this manner is avoided the trap of local optimality. The list of forbidden characteristics or movements stored in the memory is called tabu list. Typically the stopping criteria correspond to the iterations limit and to the number of consecutive iterations without improving the best obtained solution. Considering FJSSP the Tabu search technique requires a starting solution and a neighborhood structure [19]. An initial solution is used to start TS. It is stored as a current seed and the best solution. Exploring the neighborhood $V(x_i)$ of x_i are produced new solutions, called candidate solutions. The best according its objective function value candidate solution, which is not a tabu or which satisfies preliminary defined aspiration criteria, is selected as a new seed solution. This selection is called move. The new seed solution is compared with the current seed solution. If it is better, it is stored as new best solution. To avoid cycling the last L solutions and/or found are stored in the so called tabu list. Iterations are repeated until the stop criteria are satisfied.

Single objective TS algorithm for FJSSP is presented in [14]. Recently a Pareto-based tabu search algorithm for multi-objective FJSSP is proposed in [16]. Another effective hybrid Tabu search algorithm for the same multi-objective problem is developed in [15]. In [17] the multi-objective FJSSP is solved using a novel path-relinking algorithm based on the state-of-the-art Tabu search algorithm.

IV. Proposed phases of the new algorithm

1) Phase generation of one initial solution like the procedure described in [20].

2) Phase Initialization of initial swarm in the PSO.

3) Phase PSO

4) Phase TS - as an initial parameter of the algorithm is taken the global_best solution obtained by means of PSO. The path through the critical machine is assumed to be "tabu".

V. Hybrid algorithm PSO & TS

The main idea of the proposed algorithm is the following: First a good solution to be found out by means of PSO heuristic. Then the "critical path" (the sequence of operations on the machine finishing last its processing) is assumed to become "tabu". After that a TS procedure is used, attempting to improve global_best generated by PSO . The phases 3)

"PSO" and 4) "TS" are repeated iteratively until the stop condition is met as shown on Fig. 1.

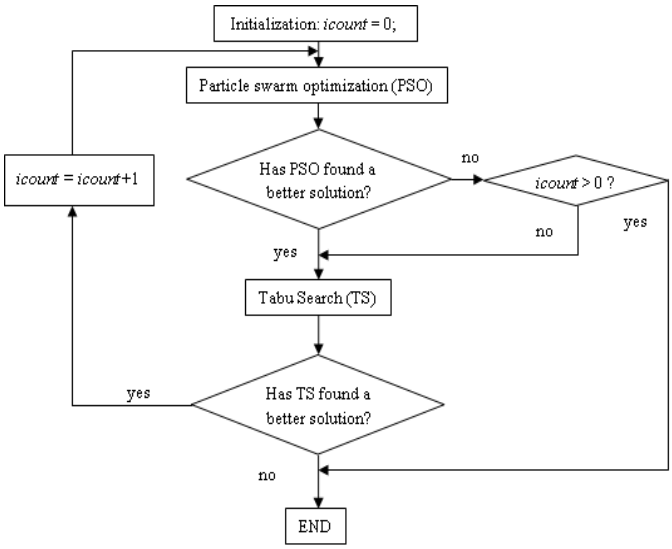

Fig. 1. Flow chart of PSO&TS algorithm

In PSO, each solution is a point in the search space and may be considered as a particle. Initially, a set of particles is randomly created and are moved through the search space. In their movement, the particles have memory, and each particle regulates its position on the base of its own experience as well as the experience of a next particle by using the best position faced by itself and its neighbors. The previous best value is known as the *pbest* (personally best) of the particle, and the best value of all the particles' *pbest* in the swarm is known as the *gbest* (globally best). In each iteration, the velocity and the position of each particle will be updated on the basis of its *pbest* and *gbest*. Let the search space is *d*-dimensional, and the position of the *i*-th particle at the *t*-th iteration is shown by $X_i^t = (x_{i1}^t, x_{i2}^t, ..., x_{iD}^t)$. The velocity of the *i*-th particle at the *t*-th iteration is $V_i^t = (v_{i1}^t, v_{i2}^t, ..., v_{iD}^t)$. The best position of the *i*-th particle found out till the *t*-th iteration is .

$P_i^t = (p_{i1}^t, p_{i2}^t, ..., p_{iD}^t)$. The best position detected for all swarm is denoted as $G^t = (g_1^t, g_2^t, ..., g_D^t)$ at the *t*-th iteration. The velocity and position of particles are updated by equations (3) and (4), corresponding to (1) and (2) in the standard PSO:

$$v_i^{t+1} = w \times v_i^t + c_1 \times rand() \times (p_i^t - x_i^t) + c_2 \times rand() \times (g_i^t - x_{id}^t) \quad (3)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (4)$$

where x_i^t and v_i^t indicate the position and velocity of the *i*-th component of one particle at the *t*-th iteration, respectively. w is the inertia weight represented to balance between the global and local search capabilities, and *rand*() is a random number creator with a uniform distribution over [0, 1]. c_1 and c_2 are the acceleration constants, which suggest the weighting of stochastic acceleration terms that force each particle toward *pbest* and *gbest*, respectively. The updating formula forces particles moving toward compound vector of local and global optimal solutions. Therefore, opportunity for particles to get the optimal solution will increase.

The pseudcode of the algorithm is presented as follows:

Step 1. Initialization.
1) PSO:
- Initialize a swarm of particles;
- Give initial values of *w, b, c, iter_lim*;
- Put the counter *i* = 0.
2) TS:
- Initialize to the algorithm to generate the initial solution S0; // global_best
- Sbest = S0; // initialize global_best
- TabuList = null;

Step 2. Iterative process:

L1: while (the stop criterion is not met)
{- generate a next swarm according to equations for particles;
- find new particle_best of each particle and global_best of swarm;
- update particle_best of each particle and global_best of the swarms;
If global_best has not been improved
Then counter *i* = *i*+1;
End If }
If counter *i* == *iter_lim*
{// for particle global_best to perform TS
until (it is not satisfied the stop condition)
CandidateList ← []
BestCandidate ← null
for (Scandidate in Senvironment)
if ((not TabuList contains Scandidate)
and (fitness(Scandidate)> fitness(BestCandidate)))
then BestCandidate ← S_candidate
end if
end for
S ← BestCandidate
If (fitness(BestCandidate)>fitness(S_best)))
S_best ← BestCandidate
end if
push into TabuList (BestCandidate);
if (TabuList.size > max TabuSize)
then TabuList.deleteFirst ()
end if }
end if
If global_best improved
Then GOTO L1
return global_best = S_best

Step 3. End. Show the optimization solution.

Variables *w, b, c, iter_lim* should be chosen by the decision-maker / the user. In the next example the following values are used: *w* = 1; *b* = 1.8; *c* = 2.2; *max_gen* = 100.

VI. Illustrative example

Below is solved an example of FJSSP of Fattahi et al. called M2J2O4, where M means machines, J – jobs and O – operations.

TABLE I
PROCESSING TIMES

p_{ijh}		*M*1	*M*2
J1	O_{11}	25	37
	O_{12}	32	24
J2	O_{21}	45	65
	O_{22}	21	65

1) $S_{km} = 0$ for every $O_{km} \in A$
Then
$S_{11} = 0$ for $O_{11} \in A$
$S_{21} = 0$ for $O_{21} \in A$
$t(A) = \min(s_{km} + p_{km}) = \min(25,65) = 25$
$M = \{0\}$
$(k,m) \in A$
$O_{11} \in A$. $SO_{11} = 25 = t(A)$. It follows that SO_{11} is not smaller of t(A).
$O_{21} \in A$. $SO_{21} = 25 = t(A)$. It follows that SO_{21} is not smaller of t(A).
Then $G = \{0\}$.
$m^* = M2$
O_{21} runs on machine 2.
Then $G = \{O_{21}\}$
$(k,m^*) = (k,2) = O_{21}$
Earlest time of $O_{21} = 65$.
$k^* = J_2$
$A = \{O_{11}\}$
$A = \{O_{11}, O_{22}\}$
$R = \{O_{21}\}$
With this ends the first iteration.
On second iteration:
$R = \{O_{21}, O_{22}\}$
On third iteration:
$R = \{O_{21}, O_{22}, O_{11}\}$
On forth iteration:
$R = \{O_{21}, O_{22}, O_{11}, O_{12}\}$ is the initializing schedule for PSO.
2) Initial schedule $R = \{O_{21}, O_{22}, O_{11}, O_{12}\}$.
3) PSO:
The schedule $R = \{O_{21}, O_{22}, O_{11}, O_{12}\}$ is not improved.
4) TS:
1) O_{11},O_{12} are run on M_1. O_{21}, O_{22} – on M_2. Makespan = 130.
O_{21},O_{11},O_{12} are run on M_1. O_{22} – on M_2. Makespan = 102. This is better than 130.
O_{11},O_{12}, O_{22} are run on M_1. O_{21} – on M_2. Makespan = 86.
O_{12} is run on M_1. O_{11}, O_{21}, O_{22} – on M_2. Makespan = 167.
O_{11} is run on M_1. O_{21}, O_{12}, O_{22} – on M_2. Makespan = 154.
2) O_{11},O_{12}, O_{22} are run on M_1. O_{21} – on M_2. Makespan = 86. Minimum.
$O_{11},O_{12},O_{21},O_{22}$ are run on M_1. M_2 - deadtime. Makespan = 187.
O_{12},O_{22} are run on M_1. O_{11}, O_{21} – on M_2. Makespan = 167.
3)O_{11},O_{12} are run on M_1. O_{21}, O_{22} – on M_2. Makespan = 130. Minimum.
O_{22} is run on M_1. O_{11}, O_{21}, O_{12} – on M_2. Makespan = 126.
O_{11},O_{21},O_{22} are run on M_1. O_{21} – on M_2. Makespan = 91.
O_{11},O_{12},O_{22} are run on M_1. O_{21} – on M_2. Makespan = 130.
O_{21},O_{22} are run on M_1. O_{11}, O_{12} – on M_2. Makespan = 66.
Minimum. Solution. (The optimal solution is obtained.)

VII. CONCLUSION

The received results are encouraging. The PSO&TS algorithm has good performance and could be applied to solve large size problems. As a future step of this study PSO&TS will be tested on a set of benchmark examples.

REFERENCES

[1] Jun-qing Li, Quan-ke Pan, Sheng-xian Xie, Yu-ting Wang, "A Fast TS Algorithm for Solving the Flexible Job-shop Scheduling Problem". In Proceeding of International Conference on Information Technology (ICIT'2009), 2009, pp:5-4.

[2] Bruker P, Schlie R, (1990), "Job-shop scheduling with multi-purpose machines", *Computing*, 1990, 45(4), pp:369-375.

[3] Brandimarte P, (1993), "Routing and scheduling in a flexible job shop by tabu search", *Annals of Operations Research*, 1993, 22, pp:158-183.

[4] Liouane N, Saad I, Hammadi S, Borne P, (2007), "Ant Systems & Local Search Optimization for Flexible Job-Shop Scheduling Production", *International Journal of Computers*, Communications & Control, 2007, pp: 2174-184.

[5] Lawler, E., J. Lenstra, A. Rinnooy Kan, and D. Shmoys. (1993) Sequencing and scheduling: algorithms and complexity. In A.H.G. Rinnooy Kan S.C Graves and P.H. Zipkin, editors, *Logistics of Production and Inventory*, volume 4, Chapter 9, pages 445–522. Elsevier, 1993.

[6] Eberhart R. C. and J. Kennedy, (1995), "A new optimizer using particle swarm theory", *Proceedings of the Sixth International Symposium on Micromachine and Human Science*, Nagoya, Japan, pp. 39-43, 1995.

[7] Kennedy J. and R. C. Eberhart, (1995), "Particle Swarm Optimization", *Proceedings of IEEE International Conference on Neural Networks*, Piscataway, N. J., 1995, pp. 1942-1948.

[8] Kennedy J., R. C. Eberhart and Y. Shi, (2001), „*Swarm intelligence*", San Francisco: Morgan Kaufmann Publishers, 2001.

[9] Ge H. W., Y. C. Liang, Y. Zhou, Guo, X.C., (2005), A Particle Swarm Optimization-based Algorithm for Job-shop Scheduling Problems, *Int. J. Computational Methods*, Vol. 2, No. 3, 2005, 419-430

[10] 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. *Computers and Industrial Engineering*, 56 (4), pp. 1309-1318.

[11] Glover F (1990) Tabu search-part II. *ORSA J. Comput.* 2:4–32

[12] Pirlot M (1996) General local search methods. *European Journal of Operational Research*, 92:493–511.

[13] Glover F. and M. Laguna, *Tabu Search*, Kluwer: Boston, MA, 1997.

[14] Saidi-Mehrabad M., Fattahi P., (2007), "Flexible job shop scheduling with tabu search algorithms", *International Journal of Advanced Manufacturing Technology*, March 2007, 32(5):563-570.

[15] Li J., Q. Pan and Y. Liang, (2010), "An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems", *Computers & Industrial Engineering*, Vol. 59, Issue 4, November 2010, pp. 647-662.

[16] Li J., Q. Pan, S. Xie and J. Liang, (2010), "A Hybrid Pareto-Based Tabu Search for Multi-objective Flexible Job Shop Scheduling Problem with E/T Penalty", *Advances in Swarm Intelligence, Lecture Notes in Computer Science*, Vol. 6145, pp. 620-627.

[17] Jia S. and Z.-H. Hu, (2014), "Path-relinking Tabu search for the multi-objective flexible job shop scheduling problem", *Computers & Operations Research*, Vol. 47, July 2014, pp. 11-26.

[18] Brucker, P., (2007) The Job-Shop Problem: Old and New Challenges, 3-rd Multidisciplinary International Scheduling Conference, Paris, 28.-31. August, 2007. http://www.mistaconference.org/2007/papers/The%20Job%20Shop%20Problem%20Old%20and%20New%20Challenges.pdf

[19] J. P. Watson and J. Christopher Beck, "Problem Difficulty for Tabu Search in Job-Shop Scheduling", Elsevier Science, 2002, vol. 21.

[20] Hongwei Ge, Wenli Du, Feng Qian, "A Hybrid Algorithm based on PSO and SA for Job Shop Scheduling", http://www.paper.edu.cn.