Int. J. Production Economics 129 (2011) 14–22

Contents lists available at ScienceDirect

Int. J. Production Economics

journal homepage: www.elsevier.com/locate/ijpe

A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search

Ghasem Moslehi, Mehdi Mahnam*

Department of Industrial and Systems Engineering, Isfahan University of Technology, 84156-83111 Isfahan, Iran

ARTICLE INFO

Article history:
Received 25 June 2007
Accepted 20 July 2010
Available online 10 August 2010

Keywords:
Flexible job-shop scheduling
Multi-objective optimization
Particle swarm optimization
Local search

ABSTRACT

The job-shop scheduling problem is one of the most arduous combinatorial optimization problems. Flexible job-shop problem is an extension of the job-shop problem that allows an operation to be processed by any machine from a given set along different routes. This paper present a new approach based on a hybridization of the particle swarm and local search algorithm to solve the multi-objective flexible job-shop scheduling problem. The particle swarm optimization is a highly efficient and a new evolutionary computation technique inspired by birds' flight and communication behaviors. The multi-objective particle swarm algorithm is applied to the flexible job-shop scheduling problem based on priority. Also the presented approach will be evaluated for their efficiency against the results reported for similar algorithms (weighted summation of objectives and Pareto approaches). The results indicate that the proposed algorithm satisfactorily captures the multi-objective flexible job-shop problem and competes well with similar approaches.

1. Introduction

Job-shop scheduling problem (JSP) is a branch of production scheduling and one of the most arduous combinatorial optimization problems. The classical JSP consists of scheduling a set of jobs on a set of machines, subject to the constraint that each job has a specified processing order throughout. JSP is an NP-hard problem (Garey et al., 1976), so different heuristic and metaheuristic algorithms are considered for solving JSP. The flexible job-shop problem (FJSP) is an extension of the job-shop problem that allows an operation to be processed by any machine from a given set along different routes. Some applications of FJSP are in planning flexible manufacturing systems (FMS), chemical materials processing plants, and transportation systems. In addition to the common complexities inherent of JSP, the flexible job-shop problem poses an even greater complexity due to the need to determine the assignment of operations to machines.

The FJS problem consists of two sub-problems of routing and scheduling. The routing sub-problem assigns each operation to a machine among a set of machines authorized for each job. The scheduling sub-problem involves sequencing the operations assigned to the machines in order to obtain a feasible schedule that minimizes a predefined objective. Brucker and Schlie (1990) were the first to address the FJSP. They proposed a polynomial algorithm for solving the FJSP with two jobs, in which the machines capable of performing one operation have the same processing time. For solving problems with more than two jobs, different hierarchical and integrated approaches have been used. According to hierarchical approaches, assigning operations to machines and their sequencing on the machines are accomplished independently from each other whereas in integrated approaches, the two tasks are jointly accomplished.

Hierarchical approaches are based on the idea of decomposing the original problem in order to reduce its complexity. Brandimarte (1993) was the first to use decomposition for the FJSP. He solved the routing sub-problem using some existing dispatching rules and then focused on the scheduling sub-problem, which is solved using a tabu search heuristic. Tung et al. (1999) developed a similar approach for scheduling a flexible manufacturing system. Mati et al. (2001) proposed a greedy heuristic to deal simultaneously with assigning and sequencing sub-problems of the flexible job-shop model. The advantage of Mati's heuristic is its ability to take into account the assumption of identical machines. Kacem et al. (2002a, 2002b) used genetic algorithm for two multi-objective approaches using either of the weighted summation of objectives or Pareto approaches. The first one (Kacem et al., 2002a) is controlled by the assigned model generated through approach by localization (AL) and the second one (Kacem et al., 2002b) used a hybridization of evolutionary and fuzzy logic algorithms to solve the FJSP. Xia and Wu (2005) applied the combination of particle swarm optimization (PSO) and simulated annealing algorithm (SA) to solve the problem. Wu and Weng (2005) considered the problem with job earliness and tardiness objectives, and proposed a multi-agent scheduling

* Corresponding author. Tel.: +98 3113912550; fax: +98 3113915526.
E-mail address: mahnam@in.iut.ac.ir (M. Mahnam).

0925-5273/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ijpe.2010.08.004

method. Gao et al. (2008) have developed a new approach hybridizing genetic algorithm with variable neighborhood descent to exploit the "global search ability" of genetic algorithm and "the local search ability" of variable neighborhood descent for solving multi-objective problem. Zhang et al. (2009) proposed hybridizing the two optimization algorithms, particle swarm optimization (PSO) and tabu search (TS) algorithms, an effective hybrid approach for the multi-objective FJSP. Recently, Xing et al. (2009a) presented a simulation model framework to solve the multi-objective flexible job-shop scheduling problem by weighted summation method. They improved the sequencing performance by using ant colony optimization algorithm (ACO). Also, Xing et al. (2009b) developed their previous approach and proposed an efficient search method for the problem.

As mentioned before, the integrated approach is used by considering assignment and scheduling at the same time. Hurink et al. (1994) proposed an approach using tabu search algorithm in which re-assignment and re-scheduling are considered as two different types of moves. An integrated approach presented by Dauzère-Pérès and Paulli (1997) defined a neighborhood structure for the problem. Chen et al. (1999) also applied genetic algorithm to solve FJSP and introduced a new coding for each solution and different crossover and mutation operators to minimize the makespan for all jobs. Mastrolilli and Gambardella (2002) improved Dauzère-Pérès' tabu search and presented two neighborhood functions. Baykasoğlu (2002) proposed an approach based on linguistics, simulated annealing and a dispatching rule based heuristic. He defined FJSP with alternative process plans (operation sequences). Also Baykasoğlu et al. (2004) presented FJSP as a grammar and defined the productions in the grammar as controls and used multiple-objective tabu search to solve the multi-objective flexible job-shop scheduling problem. They selected makespan, total tardiness and load balance as the objective functions. Chan et al. (2006) used a genetic algorithm to solve FJSP under resource constraints which follows the integrated approach. Also, Tay and Ho (2008) investigated the potential use of genetic programming (GP) for evolving effective and robust composite dispatching rules for solving the multi-objective FJSP with respect to minimum makespan, mean tardiness, and mean flow time objectives.

In this paper, we propose an integrated multi-objective approach based on hybridization of particle swarm optimization and local search algorithm to solve the flexible job-shop scheduling problem. PSO allows an extensive search of solution space while the local search algorithm is employed to reassign the machines to operations and to reschedule the results obtained from the PSO, which will enhance convergence speed.

The remainder of the paper is organized as follows: In Section 2, we describe the assumptions and formulation of the flexible job-shop scheduling problem in detail. Section 3 describes the standard PSO algorithm and its application to multi-objective FJSP. In Section 4, we present our approach to solving the FJSP. Then, the experimental results are illustrated and analyzed in Section 5. Finally, Section 6 provides conclusion and suggestions for further study of this problem.

2. Problem formulation

In flexible job-shop problem, n jobs on m machines are scheduled so that each job i has n_i operations in the form of $O_{i,1}, O_{i,2}, \ldots, O_{i,n_i}$ and the set of machines $M=\{M_1, M_2, \ldots, M_m\}$ is available. For each operation $O_{i,j}$, there is a set of machines which are capable of performing it and $O_{i,j}$ should be assigned to one of these machines (routing). The second problem is sequencing the operations on each machine in order to optimize some criteria (scheduling). In this study, the objective is to find an assignment and a schedule to minimize makespan or maximal completion time by machines (F_1); total workload of the machines (F_2), which represents the total working time of all machines; and critical machine workload (F_3). Several multi-objective approaches such as Pareto and the weighted summation of objectives have been developed but we will focus only on the Pareto approach.

For the purposes of this study, we assume that machines are independent from each other and that set up times and move times between operations are negligible. Each job has a release time r_i, indicating the earliest possible time that the job can be processed and a due date d_i. Preemption is not allowed and each job can be processed only by one machine at a time. Also, each machine can at most perform one operation at a time and machines never break down and are available during the scheduling period. The processing time of an operation $O_{i,j}$ on machine k is predefined and no operation can be interrupted.

3. Particle swarm optimization algorithm

Particle swarm optimization (PSO) is an evolutionary computation and population based on the stochastic optimization technique proposed by Eberhart and Kennedy (1995). PSO because of its two main features, namely, optimization via social evolution and simplicity of use has become the focus of attention by many researchers. PSO requires only primitive and simple mathematical operators, and it is computationally inexpensive in terms of both memory requirements and time. Members of the swarm in PSO can be considered as agents searching through the solution space. Each single solution is like a 'bird' in the D-dimensional search space, called 'particle'. All particles have fitness values which are evaluated by the fitness function, and velocities which direct the flight of the particles. Initially, PSO consists of a randomly produced population and velocity in the user's considered domain. The swarm size is problem dependent with the sizes of 20–50 as the most common sizes (Hu et al., 2004). Then, the velocity is dynamically adjusted at each step according to the experience by itself and its colleagues as given by Eq. (1). The first part of Eq. (1) represents the inertia of the previous velocity. The second part is the 'cognition' part, representing individual thinking, and the third part is 'social' awareness, representing cooperation among the particles. The new particle position is found by adding the new velocity to the current position according to Eq. (2):

$$V_{i,t+1} = w\,V_{i,t} + Rand\,C_1(P_i - X_{i,t}) + rand\,C_2(P_g - X_{i,t}) \quad (1)$$

$$X_{i,\,t+1} = X_{i,\,t} + V_{i,\,t+1} \quad (2)$$

wherein i is the ith particle; $X_{i,t}$ is the position of particle i in iteration t; $V_{i,t}$ is the velocity of particle i in iteration t; P_i is the best previous position of particle i so far, also called *pbest* (memorized by every particle); and P_g is the best previous position among all the particles, which is called *gbest* (memorized in a common repository). w is inertial weight and its function is to balance global and local exploitations of the swarm. One of the most widely used methods of inertia weighting is linear decreasing, which is determined by the following equation:

$$w = w_{\max} - ((w_{\max} - w_{\min})/iter_{\max}) \times iter \quad (3)$$

where $w_{\max}$ is the initial value of weighting coefficient; $w_{\min}$, the final value of weighting coefficient; $iter_{\max}$, maximum number of iterations; and *iter* is the current iteration. C_1 and C_2 are two learning factors which control the influence of *pbest* and *gbest* on the search process and the most common value for them is

$C_1=C_2=2$. *Rand* and *rand* are two random numbers within the range of [0,1].

The procedure for standard PSO is summarized as follows:

Step 1: Initialize a population of particles with random positions and velocities in the *D*-dimensional problem space.

Step 2: Evaluate the objective values of all particles, set *pbest* of each particle equal to its current position, and set *gbest* equal to the position of the best initial particle.

Step 3: Update the velocity and position of particles according to Eqs. (1) and (2).

Step 4: Evaluate the objective values of all particles.

Step 5: For each particle, compare its current objective value with its *pbest* value. If the current value is better, then update *pbest* with the current position and objective value.

Step 6: Determine the best particle of the current whole population with the best objective value. If the objective value is better than that of *gbest*, then update *gbest* with the current best particle.

Step 7: If a stopping criterion is met, then output *gbest* and its objective value; otherwise, go back to Step 3.

3.1. Neighborhood structure

As mentioned earlier, *gbest* is the best global solution among all particles. *gbest* is an extreme situation of the *nbest* version in which all particles of the population are considered as neighbors of each particle. We can use *nbest* as the best position of *n* neighbor particles have achieved so far. The neighborhood of a particle is the social environment a particle encounters. So, a particle is affected only by its neighbors not all population. It is generally accepted that a larger neighborhood size will make the particles to converge faster, while a small neighborhood size will help prevent creation of premature or pre-convergence (Hu et al., 2004).

Kennedy and Mendes (2002) investigated various neighborhood structures and their influences on the performance of the algorithm. *ring* and *star* topologies are the most widely used neighborhood structures. In the ring topology, the neighborhood size is assumed to be three and each particle can communicate with its two nearest neighbors while in the latter topology the particle has communication with all neighbors. For determining the neighbors, simply a list of particles is generated regardless of their positions. This structure which is called social, has a simple definition which is a great advantage in discrete problems while the *geographical* neighborhood as a different structure need to compute the distances between particles and finding the positions of the nearest particles in a neighborhood. In this study the star and social neighborhood structure is employed which has an important effect on controlling the convergence of algorithm.

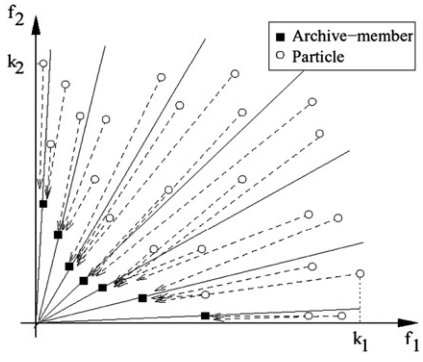

Fig. 1. Finding the best local guide with elitist strategy using sigma method (Mostaghim and Teich, 2003).

3.2. Multi-objective particle swarm optimization

In PSO, *pbest* and *nbest* (*gbest*) play crucial roles in guiding the particle's search. Transforming PSO to MOPSO requires a redefinition of *nbest* (*gbest*) selection rule in order to obtain an optimal front solution. Naturally, this selection is from the set of Pareto-optimal solutions. This problem is both difficult and important for attaining convergence and diversity of solutions. Since each particle of the swarm should select one of the Pareto-optimal solutions as its global best particle, the one selected is called the best local guide of that particle. So, the most important part of MOPSO is determination of the best local guide of particles. In this method, elitism strategy is considered in order not to lose the non-dominated solutions during generations. According to elitism strategy, the algorithm evaluates the particles in the population and compares members of the current population with members of the actual archive. Then, it surveys the particles to be considered for insertion into the archive and those to be removed. Thus, if no two points in the archive dominate each other, the archive is called *'domination-free'*. Fig. 1 shows how to find the best local guide through the elitist strategy using the sigma method.

The selection of the best local guide for each particle of the population from a set of Pareto-optimal solutions in different multi-objective particle swarm optimization methods has a great impact on the convergence and diversity of solutions. In the multi-objective PSO presented in this paper, we used the sigma method introduced by Mostaghim and Teich (2003). According to this method, a value σ_i is assigned to each point with coordinates (f_{1i}, f_{2i}) and σ is defined as $\sigma=(f_1^2-f_2^2)/(f_1^2+f_2^2)$. Thus, all points occurring on the line $f_2=af_1$ will have the same value for $\sigma=(1-a)^2/(1+a)^2$. In its general form, σ is a vector of C_2^k elements, where k is the dimension of the objective space. For example, for the three coordinatesf_1,f_2,f_3, σ is defined as

$$\vec{\sigma}=\frac{\begin{pmatrix} f_1^2-f_2^2 \\ f_2^2-f_3^2 \\ f_3^2-f_1^2 \end{pmatrix}}{(f_1^2+f_2^2+f_3^2)} \tag{4}$$

Different values of σ are shown in Fig. 2.

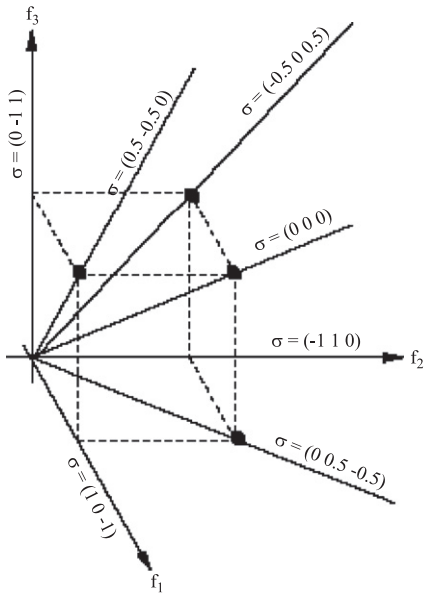

Fig. 2. Sigma method for a three objective pace (Mostaghim and Teich, 2003).

In order to find the best local guide, in the first step, we assign the value σ_j to each particle j in the archive. In the second stage, σ_i for particle i of the population is calculated. Then, the distance between particle i and all the archive points is calculated. Finally, particle k in the archive whose σ_k has the minimum distance from σ_i is selected as the best local guide for particle i. In other words, the particle with a sigma value closer to that of the archive member is selected as its best local guide. The sigma method can be used for different but positive objects. An important point to note here is that the range of equiponderance of objects has a great impact on the convergence and diversity of solutions.

4. Hybridization of MOPSO and local search to the FJS problem (MOPSO+LS)

In this paper, we use an integrated approach to the problem to the effect that each particle consists of the two parts, A and B, the former defining the routing policy of the problem (machine assignment to operations) and the latter indicating the sequence of the operations on each machine (operations priority).

4.1. Routing and scheduling

The value of part A assigns a priority level for each operation representing the machine selected for the operation. So, in the first step, we sequence the machines available for an operation according to the increasing order of processing time. Then, we get different priority levels for all machines processing the same operation. Because a different priority level is assigned to each machine, particle position corresponds to a machine assignment of all operations. Every bit of the particle position in part A should be an integer but after being computed by Eqs. (1) and (2), some bits may appear as real values. Therefore, we round off the real values to the nearest integer number. A stochastic particle position of assignment sub-particle A is represented in Fig. 3.

The second part of the particle which determines the scheduling of the solution and its structure is based on the priority (random key). In this form of the problem, each element in the particle represents an operation and the corresponding value of each element designates the priority of that operation. The important feature of random keys is that all offspring formed by the algorithm are feasible solutions. So, the PSO is used for this approach because the priorities are real numbers. Fig. 4 shows a priority-based particle as an illustration.

For the scheduling portion, we used the strategy presented by Gonçalves et al. (2005) in dealing with job-shop scheduling problem. The basic idea of this strategy is to control the delay times allowed for each operation. By controlling the maximum delay time allowed, one can reduce or increase the solution space. A maximum delay time equal to zero is equivalent to restricting the solution space to non-delay schedules. The procedure is based on a time-increment generation scheme. Each particle is made of $2n$ genes in the range of [0,1], where n is the number of operations. The first n genes are used as operation priorities and the second n genes are used to determine the delay times used in scheduling an operation. The delay time of operation j used by current iteration is calculated from the following equation:

$$Delay_j = X(n+j) \times 1.5 \times P_{\max} \quad (5)$$

where $Delay_j$ is the allowed delay time of operation j and $P_{\max}$ is the maximum processing time of all operations. In each iteration, the operation with the highest priority is selected among the operations whose precedence is met in the interval $[t;\ t+Delay_j]$ (set E). This operation will be scheduled at the earliest in terms of its release time and precedence and capacity constraints. Then, the time t is updated and the procedure is repeated. This algorithm continues until all operations are scheduled.

4.2. Local search algorithm

In general, the hybridization of different heuristics provides more efficient search methods since the two goals of exploration and exploitation, simultaneously. Exploration allows extensive search to determine the part of the solution space that has a higher chance of containing the global optimum whereas exploitation refines the search and focuses on a special part of the space. Exploration can be obtained via population-based heuristics such as genetic algorithms (GA), ant colony optimization (ACO), particle swarm optimization (PSO), etc; while exploitation would be obtained by local search heuristics such as hill climbing (HC), simulated annealing (SA), and tabu search. An instance of this kind of combination, using genetic algorithm and local search, is considered by Tseng and Lin (2010). In this paper, a local search algorithm is applied to each particle in order to reassign machines to operations while taking account of the scheduling accomplished for each particle. The implementation of this algorithm is after evaluating the particles so it does not cause to miss previous solutions. This algorithm begins with identifying the critical path in the solution obtained from the scheduling procedure. Then critical operations are assigned to machines with shorter processing times and earlier finish times considering the release time, precedence and capacity constraints. This procedure is continued until the makespan improve. So, the assignment and scheduling vectors should be corrected. The priority of an operation is changed such that it lies between the priority of the preceding and following operations on the reassigned machine. Obviously, the local algorithm decreases the objects especially in primary iterations and increases the convergence speed. This procedure is shown in Fig. 5.

4.3. Proposed algorithm

The general structure of the proposed algorithm is shown in Fig. 6. As seen in the stepwise procedure, in each iteration, the MOPSO algorithm updates the particles based on their *pbest* and *nbest* and then they are improved by the local search algorithm. Since it may occasionally happen that the algorithm is easily trapped

Particle k: X_k	0.81	0.67	0.23	...	0.34	0.94	0.07	0.47

Fig. 4. Particle-represented priorities.

	Job1		Job2			Job 3	
operation	1	2	1	2	3	1	2
particle position	5	4	2	5	1	3	1
processing machine	M_2	M_3	M_1	M_4	M_2	M_4	M_2

Fig. 3. A stochastic particle position representation (Xia and Wu, 2005).

into the local optima, mutation operator, based on the two parameters m_1 and m_2, is employed in the intermediate stages. This operator chooses the particles with a probability of m_1 and randomly changes each dimension of the particle with a probability of m_2. Using neighborhood (*nbest*) is another method of preventing the particle from pre-convergence; social neighborhood is, thus, applied

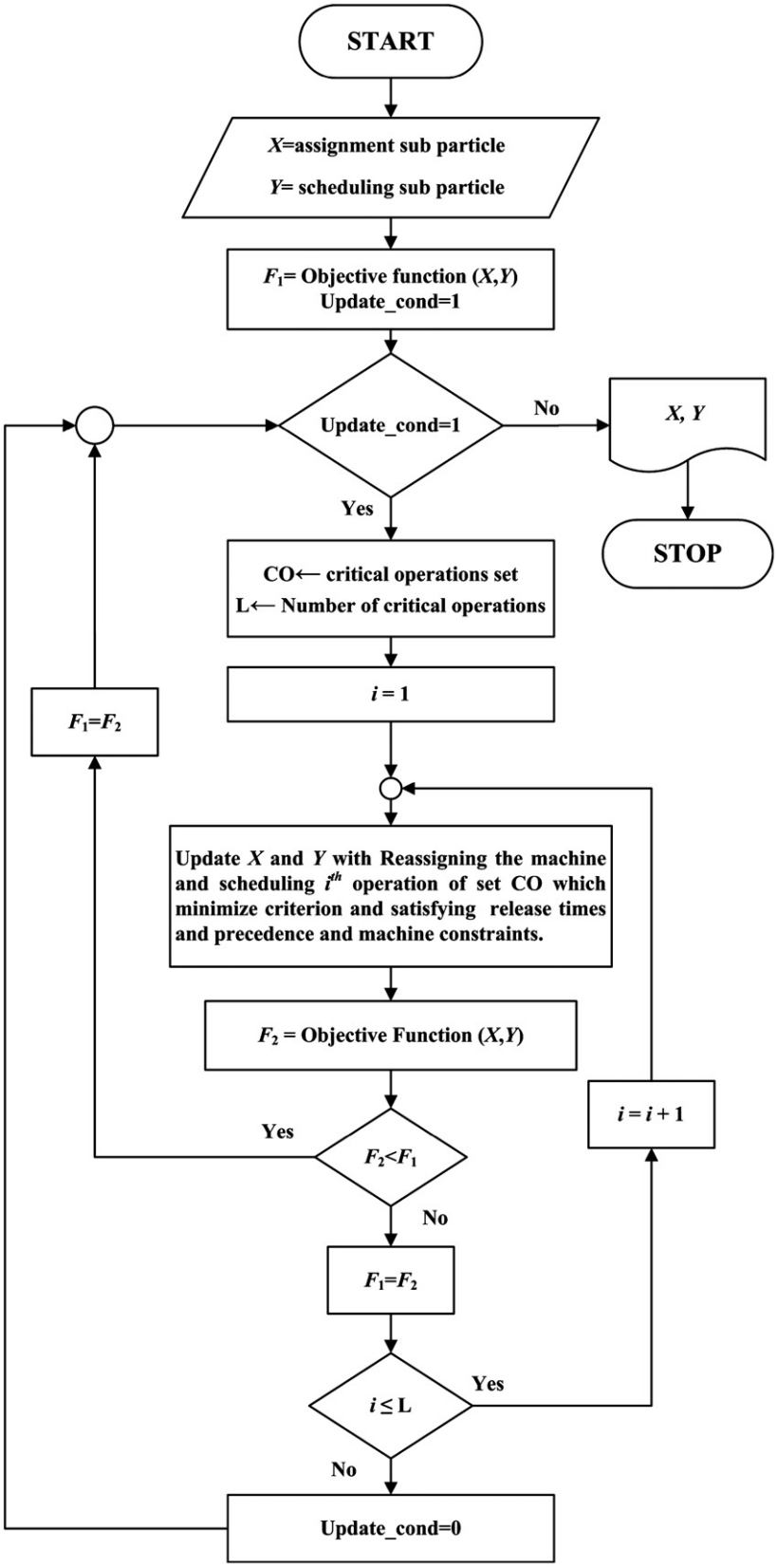

Fig. 5. Local search algorithm scheme.

in this approach. But for providing the possibility of searching boundaries between neighborhoods, if after g_1 iterations, the Pareto-optimal solutions show no improvement, then global neighborhood will replace the local one. Also, if the algorithm reaches the local optima in the second stage – i.e., the result is not improved after g_2 iterations – the swarm will be randomly reproduced again.

Because the PSO algorithm is sensitive to the primary swarm, it would be better to use some heuristic method to find the proper

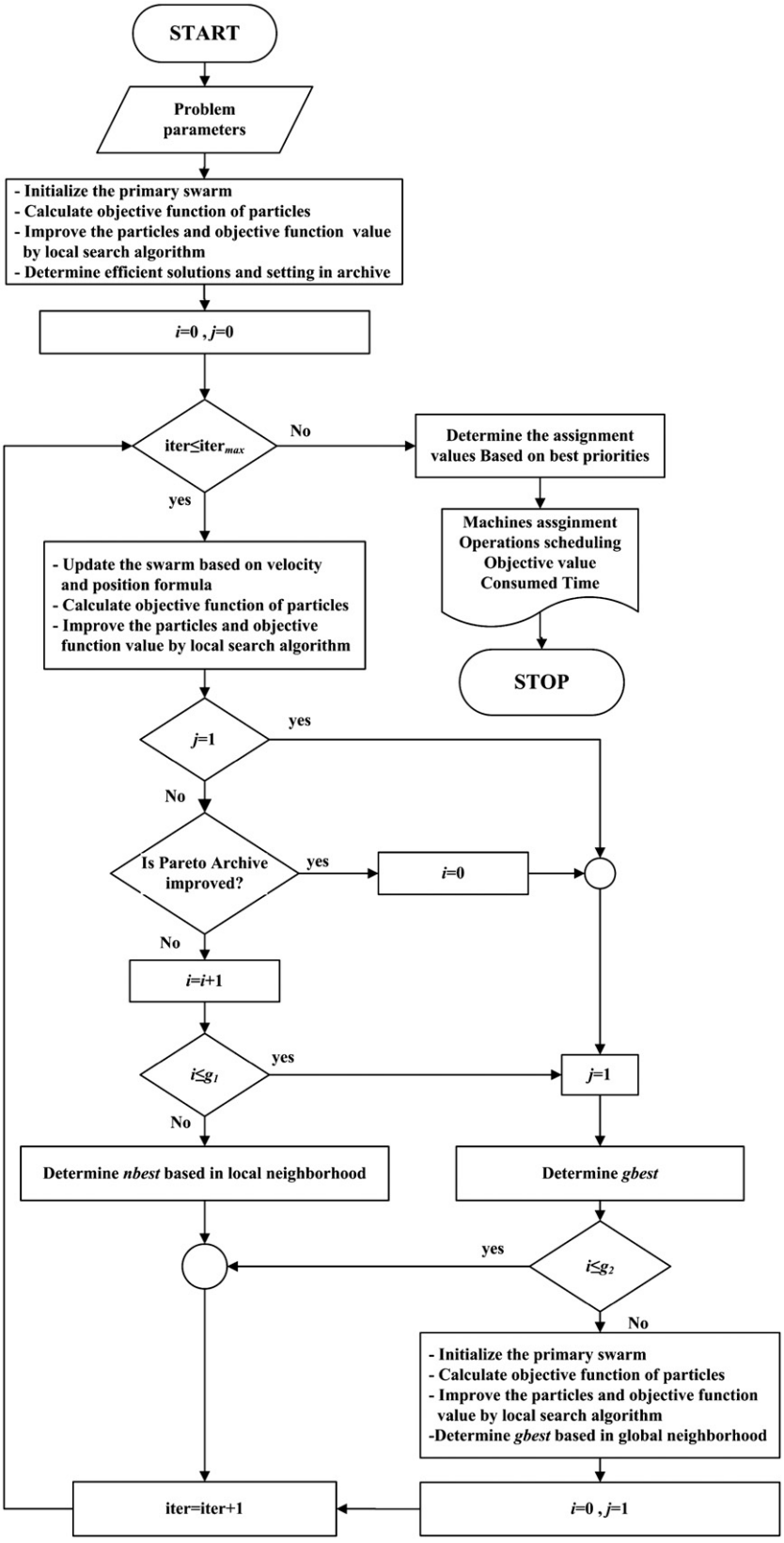

Fig. 6. The scheme of MOPSO+LS algorithm.

initial points. One of the favorable assignments could be the first operation priority (the machine with the lowest processing time), which is identified as a dispatching rule. Also, a greedy heuristic algorithm as introduced by Kacem et al. (2002a) is employed to assign appropriate machines to operations. This procedure enables us to assign each operation to the right machine while taking into account the processing times and workloads of machines according to which we have already assigned operations. Below is a brief summary of the assignment algorithm (Kacem et al., 2002a):

Assignment procedure:

P_{ijk}: processing time of operation $O_{i,j}$ on machine k
S_{ijk}: Assignment of operation $O_{i,j}$ to machine k

Initialization: Set all elements of S to 0 and recopy P in P'.
For $i=1$ to n
  For $j=1$ to n_i
    Min$=\infty$;
    position$=1$;
    For $k=1$ to m
      IF ($P' <$ Min) Then
        Min$=P'_{ijk}$;
        position$=k$;
      EndIF
    EndFor
    $S_{i,j,\text{position}}=1$;
    For $j'=j+1$ to n_i
      $P'_{i,j',\text{position}}=P_{i,j,\text{position}}+P'_{i,j',\text{position}}$;
    EndFor
    For $i'=i+1$ to n
      For $j'=1$ to $n_{i'}$
        $P'_{i',j',\text{position}}=P_{i,j,\text{position}}+P'_{i',j',\text{position}}$;
      EndFor
    EndFor
EndFor

5. Computational results

Computational experiments were carried out to compare the approaches and to evaluate the efficiency of our proposed method. The algorithms were implemented in Matlab 7 on a Pentium IV running at 2 GHz on the Windows XP operating system. In this section, the presented approach is compared with other methods proposed for solving the FJSP. Other algorithms are categorized in two groups of weighting summation of objectives and Pareto approaches, each of which is separately inspected below due to their different natures.

5.1. Weighting summation of objectives approach.

AL+CGA (Kacem et al., 2002a), PSO+SA (Xia and Wu, 2005), GA+VND (Gao et al., 2008), PSO+TS (Zhang et al., 2009), SACO (Xing et al., 2009a) and ESM (Xing et al., 2009b) algorithms are included in the group of approaches applied to FJSP that used the weighted summation of objectives. In these algorithms, three instance problems based on practical data in three small (8×8), medium (10×10), and large sizes (15×10) were employed as benchmark problems. Different release times are not considered by these algorithms; so, release time values in the proposed algorithm (MOPSO+LS) are considered zero. Because the computation times, maximum runs, and weights are not reported in the literature, an accurate comparison seems unachievable. However, the comparison of the approach presented here with other algorithms shows that the priority-based approach is competitive with all others. Even if the proposed algorithm did not compete well with other algorithms using the weighting summation of objectives, this could not be considered as its weakness. The values were selected for parameters are shown in Table 1. Also, thirty runs are made for each case to check the consistency. Table 2 show the results obtained from MOPSO+LS and from previous methods for all instances. The best solutions are indicated. The results indicate that ESM algorithm has better results among previous algorithms. However it is because that this algorithm was run with different weight sets for different objectives and the best results were presented. In the 8×8 problem, the solutions of MOPSO+LS are dominating AL+CGA in all iterations. Also the solution of other algorithms is among MOPSO+LS solutions but never dominates. So, in small-sized problems, our proposed approach has no dominating solution but is not dominated by them, either. In the 10×10 problem, our algorithm often has better or at least equiponderant solutions compared with others. The MOPSO+LS algorithm was able to dominate previous algorithms for problems of the size 15×10 and this result is based on the best solution found in all iterations. In addition, results indicate that in 70% of the iterations, there is always one better solution or at least one equal to the best previous solution. Finally, although we were not able to compare

Table 1
MOPSO+LS algorithm parameters in comparison with weighted summation methods.

$n\times m$	Swarm size	Neighborhood size	Iterations	$W_{\max}$	$W_{\min}$	C_1	C_2	g_1	g_2
8×8	70	35	65	0.9	0.4	0.8	1.2	7	15
10×10	70	35	65	0.9	0.4	0.8	1.2	7	15
15×10	100	33	100	0.9	0.4	0.8	1.2	14	18

Table 2
Comparison of MOPSO+LS with weighted summation objectives approaches.

Size ($n\times m$)	Method	AL+CGA		PSO+SA		GA+VND	PSO+TS		SACO		SME				MOPSO+LS				
	Objects	1	2	1	2	1	1	2	1	2	1	2	3	4	1	2	3	4	5
8×8	Makespan (F_1)	15	16	15	16	14	15	14	15	14	15	17	14	16	15	16	14	16	17
	Total workload (F_2)	79	75	75	73	77	75	77	76	77	75	73	77	77	75	73	77	78	77
	Max workload (F_3)	–	–	12	13	12	12	12	12	12	12	13	12	11	12	13	12	11	11
10×10	Makespan (F_1)	7	–	7	–	7	7	–	7	8	8	7	8	–	8	7	8	7	–
	Total workload (F_2)	45	–	44	–	43	43	–	42	42	41	42	42	–	41	42	42	43	–
	Max workload (F_3)	5	–	6	–	5	6	–	6	5	7	6	5	–	7	6	5	5	–
15×10	Makespan (F_1)	23	24	12	–	11	11	–	11	11	11	–	–	–	12	11	–	–	–
	Total workload (F_2)	95	91	91	–	91	93	–	91	93	91	–	–	–	93	91	–	–	–
	Max workload (F_3)	11	11	11	–	11	11	–	11	10	11	–	–	–	10	11	–	–	–

Pareto and the weighting summation of objectives approaches in a proper manner, we were able to conclude that our proposed approach not only is equiponderant with the best previous algorithms, but also comprises better results.

5.2. Pareto approach

Kacem et al. (2002b) applied the Pareto approach to solve the multi-objective FJSP by hybridization of evolutionary algorithms and fuzzy logic (FL+EA). They also evaluated their approach in instance problems of different sizes with four samples of 4×5, 10×7, 10×10 and 15×10. Kacem et al. (2002b) considered different release times as follows:

Instance $1(4\times5): r_1=3, r_2=5, r_3=1$ and $r_4=6$.

Instance $2(10\times7): r_1=2, r_2=4, r_3=9, r_4=6,$
$r_5=7, r_6=5, r_7=7, r_8=4, r_9=1$ and $r_{10}=0$.

Instance $3(10\times10): r_j=0$ for all j.

Instance $4(15\times10): r_1=5, r_2=3, r_3=6, r_4=4,$
$r_5=9, r_6=7, r_7=1, r_8=2, r_9=8, r_{10}=0,$
$r_{11}=14, r_{12}=13, r_{13}=11, r_{14}=12$ and $r_{15}=5$.

The proposed algorithm parameter values in this comparison are shown in Table 3. Unfortunately, they did not report average values of computation time, maximum iterations, and quality of the solutions in different runs. Table 4 indicates the comparison of the solutions by the two approaches. It is observed that the solutions are improved for all problem sizes. Although Table 4 presents only the best solutions yielded by MOPSO+LS, all answers obtained from all runs of MOPSO+LS were better than those by FL+EA.

Table 3
MOPSO+LS algorithm parameters in comparison with Pareto approach.

$n\times m$	Swarm size	Neighborhood size	Iterations	W_{max}	W_{min}	C_1	C_2	g_1	g_2
4×5	70	35	65	0.9	0.4	0.8	1.2	7	15
10×7	75	35	75	0.9	0.4	0.8	1.2	7	15
10×10	80	35	90	0.9	0.4	0.8	1.2	7	15
15×10	100	33	120	0.9	0.4	0.8	1.2	14	18

Table 4
Comparison of MOPSO+LS and FL+EA.

Size ($n\times m$)	Method	FL+EA					MOPSO+LS		
	Objects	1	2	3	4	5	1	2	3
4×5	Makespan (F_1)	18	18	16	16	–	16	16	–
	Total workload (F_2)	32	33	35	34	–	32	33	–
	Max workload (F_3)	8	7	9	10	–	8	7	–
10×7	Makespan (F_1)	16	15	18	17	16	16	15	15
	Total workload (F_2)	60	61	63	64	66	60	61	62
	Max workload (F_3)	12	11	10	10	10	12	11	10
10×10	Makespan (F_1)	8	7	8	–	–	8	7	7
	Total workload (F_2)	41	45	42	–	–	41	44	42
	Max workload (F_3)	7	5	5	–	–	7	5	6
15×10	Makespan (F_1)	24	23	–	–	–	23	–	–
	Total workload (F_2)	91	95	–	–	–	91	–	–
	Max workload (F_3)	11	11	–	–	–	11	–	–

6. Conclusion

In this paper, a new approach based on a hybridization of particle swarm algorithm and a local search algorithm was presented to solve the multi-objective flexible job-shop scheduling problem with different release times. Although the optimization is not guaranteed in this approach, appropriate and competitive solutions are obtained at satisfactory computation times. The efficiency of the new approach was compared against the results reported from other algorithms (weighting summation of objectives and Pareto) to evaluate the proposed algorithm. The results indicate that the proposed algorithm is an effective and competitive approach compared to the multi-objective flexible job-shop problem. MOPSO+LS algorithm is compared to weighting summation methods even without considering different release times and it was able to produce new satisfactory answers in addition to those already generated by other methods. For medium-sized problems, remarkable results are obtained and dominating solutions are generated with high frequency. For large-sized problems, the previous good solutions were repeated while in some runs, the benchmark was also exceeded. Compared to Pareto approaches, the results obtained in this study were far better. Although the proposed algorithm passed the tests satisfactorily, more computational study will be required to test the efficiency of the proposed technique. Generally, PSO is a new optimization algorithm with great potentials for its efficiency to be enhanced. Furthermore, application of other methods adapting the PSO algorithm to discrete and multi-objective spaces is another area recommended for future research.

Acknowledgement

The authors would like express gratitude to anonymous referees for their valuable comments on reviewing this paper.

References

Baykasoğlu, A., 2002. Linguistic-based meta-heuristic optimization model for flexible job shop scheduling. International Journal of Production Research 40, 4523–4543.
Baykasoğlu, A., Özbakir, L., Sönmez, A., 2004. Using multiple objective tabu search and grammars to model and solve multi-objective flexible job shop scheduling problems. Journal of Intelligent Manufacturing 15, 777–785.
Brandimarte, P., 1993. Routing and scheduling in a flexible job shop by taboo search. Annals of Operations Research 41, 157–183.
Brucker, P., Schlie, R., 1990. Job-shop scheduling with multi-purpose machines. Computing 45, 369–375.
Chan, F.T.S., Wong, T.C., Chan, L.Y., 2006. Flexible job-shop scheduling problem under resource constraints. International Journal of Production Research 44, 2071–2089.
Chen, H., Ihlow, J., Lehmann, C., 1999. A genetic algorithm for flexible job shop scheduling problem. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1120–1125.
Dauzère-Pérès, S., Paulli, J., 1997. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. Annals of Operations Research 70, 281–306.
Eberhart, R., Kennedy, J., 1995. A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39–43.
Gao, J., Sun, L., Gen, M., 2008. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Computers and Operations Research 35, 2892–2907.
Garey, M.R., Johnson, D.S., Sethi, R., 1976. The complexity of flow shop and job shop scheduling. Mathematics of Operations Research 1, 117–129.
Gonçalves, J.F., Mendes, J.J.M., Resende, M.G.C., 2005. A hybrid genetic algorithm for the job shop scheduling problem. European Journal of Operational Research 167, 77–95.
Hu, X., Shi, Y., Eberhart, R., 2004. Recent advances in particle swarm. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 90–97.
Hurink, J., Jurisch, B., Thole, M., 1994. Tabu search for the job shop scheduling problem with multi-purpose machines. Operations Research Spectrum 15, 205–215.

Kacem, I., Hammadi, S., Borne, P., 2002a. Approach by localization and multi objective evolutionary optimization for flexible job-shop scheduling problems. IEEE Transactions on Systems, Man, and Cybernetics, 1–13.

Kacem, I., Hammadi, S., Borne, P., 2002b. Pareto-optimality approach for flexible job-shop scheduling problems. Hybridization of evolutionary algorithms and fuzzy logic. Mathematics and Computers in Simulation 60, 245–276.

Kennedy, J., Mendes, R., 2002. Population structure and particle swarm performance. In Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1671–1676.

Mastrolilli, M., Gambardella, L.M., 2002. Effective neighborhood functions for the flexible job shop problem. Journal of Scheduling 3, 3–20.

Mati, Y., Rezg, N., Xie, X., 2001. An integrated greedy heuristic for a flexible job shop scheduling problem. IEEE International Conference on Systems, Man and Cybernetics,2534–2539.

Mostaghim, S., Teich, J.R., 2003. Strategies for finding local guides in multi-objective particle swarm optimization (MOPSO). In Proceedings of the IEEE Swarm Intelligence Symposium, pp. 26–33.

Tay, J.C., Ho, N.B., 2008. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Computers and Industrial Engineering 54, 453–473.

Tseng, L.Y., Lin, Y.T., 2010. A genetic local search algorithm for minimizing total flow time in the permutation flow shop scheduling problem. International Journal of Production Economics 127(1), 121–128.

Tung, L.F., Lin, L., Nagi, R., 1999. Multi-objective scheduling for the hierarchical control of flexible manufacturing systems. The International Journal of Flexible Manufacturing Systems 11, 379–409.

Wu, Z., Weng, M.X., 2005. Multiagent scheduling method with earliness and tardiness objectives in flexible job shops. IEEE Transactions on System, Man, and Cybernetics-Part B 35 (2), 293–301.

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

Xing, L-N., Chen, Y-W., Yang, K-W., 2009a. Multi-objective flexible job shop schedule: design and evaluation by simulation modeling. Applied Soft Computing 9, 362–376.

Xing, L-N., Chen, Y-W., Yang, K-W., 2009b. An efficient search method for multi-objective flexible job shop scheduling problems. Journal of Intelligent Manufacturing 20 (3), 283–293.

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