International Journal of Computational Intelligence and Applications

HYBRID FUZZY LOGIC-BASED PARTICLE SWARM OPTIMIZATION FOR FLOW SHOP SCHEDULING PROBLEM

S. H. LING*
Faculty of Engineering and Information Engineering, University of Technology, Sydney, NSW 2007, Australia.

F. JIANG
Faculty of Engineering and Information Engineering, University of New South Wales, NSW 2052, Australia.

H. T. NGUYEN
Faculty of Engineering and Information Engineering, University of Technology, Sydney, NSW 2007, Australia.

K. Y. CHAN
DEBI Institute, Curtin University of Technology, Perth, WA 6102, Australia.

This paper proposes a hybrid fuzzy logic-based particle swarm optimization (PSO) with cross-mutated operation method for the minimization of makespan in permutation flow shop scheduling problem. This problem is a typical non-deterministic polynomial-time (NP) hard combinatorial optimization problem. In the proposed hybrid PSO, fuzzy inference system is applied to determine the inertia weight of PSO and the control parameter of the proposed cross-mutated operation by using human knowledge. By introducing the fuzzy system, the inertia weight becomes adaptive. The cross-mutated operation effectively forces the solution to escape the local optimum. To make PSO suitable for solving flow shop scheduling problem, a sequence-order system based on the roulette wheel mechanism is proposed to convert the continuous position values of particles to job permutations. Meanwhile, a new local search technique namely swap-based local search for scheduling problem is designed and incorporated into hybrid PSO. Finally, a suite of flow shop benchmark functions are employed to evaluate the performance of the proposed PSO for flow shop scheduling problems. Experimental results show empirically that the proposed method outperforms significantly the existing hybrid PSO methods.

Keywords: Flow shop; Fuzzy logic; Particle swarm optimization; Roulette wheel mechanism, Scheduling.

1. Introduction

The flow shop scheduling problem is a very complex combinational optimization problem with many variations. Permutation flow shop scheduling problem consists of scheduling given jobs with same order at all machines. Flow shop scheduling

*Corresponding author. Email: Steve.Ling@uts.edu.au

problem is a Non-Polynomial (NP) hard problem, existing mathematical methods, i.e., integer programming, branch-and-prune method, dynamic programming, etc, are only applicable to small-scale problem and they are difficult to find the global solution of the problem. Thus, various global searching algorithms such as genetic algorithm (GA) [1] [2], Tabu search [3], differential evolution [4] and ant colony optimization [5] [6] are proposed to solve the flow shop scheduling problems. Recently, a new global searching algorithm namely particle swarm optimization (PSO) is widely used to different industrial areas such as power systems [7] [8] [9], parameters learning of neural networks [10] [25], control [12], inverse problem [13], modelling [14], and so on. Especially, PSO is effectively to tackle the flow shop scheduling problem [15] [16] [17] [18] [19] [20]. In the paper [17] [19], PSO is presented to solve the flow shop problem, where the smallest position value (SPV) rule is developed to enable the continuous particle swarm optimization to be applied to all classes of sequencing problems. In addition, a local search method namely variable neighborhood search (VNS) is proposed for embedding in the PSO method to solve the flow shop problems [19], and in the paper [17], simulated annealing based local search with multiple different neighborhoods is designed. In [18], a hybrid PSO is used to solve the flow shop scheduling problem where simulated annealing (SA) algorithm is incorporated into PSO. In the paper [20], a modified binary PSO algorithm is presented to solve the flow shop problem, which all particles are coded with binary number.

Particle swarm optimization (PSO) is inspired by the social behaviors of animals like fish schooling and bird flocking [21]. Comparing with other population based stochastic optimization methods, such as evolutionary algorithms, PSO has comparable or even superior search performance for many hard optimization problems with faster and more stable convergence rates [22]. Furthermore, PSO has memory, previously visited best positions in PSO are remembered, while in evolution algorithms, which are forgotten once the current population changes. However, observations reveal that PSO converges sharply in the early stage of the searching process, but saturates or even terminates in the later stage. It behaves like the traditional local searching methods that trap in local optima.

Recently, different hybrid PSO methods have been proposed to overcome the drawback of trapping in local optima. The hybrid PSO has been first proposed in 1998 [23], in which a standard selection mechanism is integrated with PSO. A new hybrid gradient descent PSO (HGPSO), which is integrated with gradient information to achieve faster convergence without getting trapped in local minima is proposed [24]. However, the computational demand of HGPSO is increased by the process of the gradient descent. In addition, it is poor to handle the multimodel problem that contain many local minima. In the paper [25], a hybrid PSO algorithm named HGAPSO is proposed, which incorporates GA's evolutionary operations of crossover, mutation and reproduction. In the paper [7], a hybrid PSO named HP-

SOM is proposed, in which a constant mutating space is used in mutation. In both HGAPSO and HPSOM, the solution space can be explored by performing mutation operations on particles along the search, and premature convergence is more likely to be avoided. However, the mutating space is kept unchanged all the time throughout the search, and the space for the permutation of particles in PSO is also fixed. It can be improved by varying the mutating space along the search. A hybrid PSO with wavelet mutation operation (HPSOWM) is proposed [26], which the mutating space is varying by applying wavelet theory. The solution quality and solution reliability (standard deviation upon many trials) are improved.

In order to make PSO be suitable to solve the combinational (scheduling) problem, a converting process from the continuous position values of particles to job permutations is used. Conventionally, the Smallest Position Value (SPV) rule [17] [18] [19] [28] [29] is actually a mapping from an N-dimensional continuous space to a sequence with N numbers. The corresponding sequence is represented by the order of the vector. As we know, the positions of the particles in PSO are updated that causes velocity and evolution. In some case, the positions of the particles are updated, the corresponding sequence is still unchange by using SPV rule [20]. This kind of inefficiency of the continuous space will result in the decrease in convergence rate. To overcome this drawback, a sequence-order system based on the roulette wheel mechanism is proposed in this paper. In this mechanism, the probability representation is instead of the position value which is represented by SPV rule.

In the paper [30], an improved version of PSO with an inertia weight factor is introduced. The aim of the inertia weight provides a balance between the global exploration and local exploitation and it is governed by a linear characteristic function. However, not all the optimization problems are linear. Thus, in this paper, a hybrid fuzzy logic-based PSO with cross-mutated operation is proposed. This proposed approach can be divided into two main parts. First, an adaptive inertia weight is proposed, which incorporates with the fuzzy inference system. Fuzzy logic is good in representing some expert knowledge and experience in some linguistic rule which can be easily understood by the human being. Using fuzzy inference to determine the inertia weight of PSO, such that, the characteristic of function of inertia weight becomes nonlinear. It will be shown that the nonlinear characteristic of the inertia weight performs more better solution quality in this paper. Second, a new operation namely cross-mutated (CM) operation is introduced. The CM operation effectively solves the drawback of PSO, which makes PSO easier to trap in the local optima. In CM operation, the control parameter is determined by fuzzy rules and the operation becomes adaptive. Additionally, a swap-based local search method is designed for the local exploration on a discrete job permutation space. This method is used to fine tune the sequence of the flow shop scheduling problem. In this paper, a suite of benchmark problems for permutation flow shop scheduling problem are used to illustrate the performance of the proposed approach. The resulting adap-

tive inertia weight and CM operation aid the proposed fuzzy logic-based PSO with cross-mutated operation to perform better solution quality and solution reliability compared with improved PSO (IPSO) [30] and other hybrid PSO methods [7 24 25 26] in solving a suite of benchmarking flow shop scheduling problems.

This paper is organized as follows. Section 2 presents the formulation of flow shop scheduling problem. The implementation of the flow shop scheduling problem with hybrid PSO will be discussed in Section 3. In this section, the solution representation, hybrid fuzzy-based PSO with cross-mutated operation, and the swap local search method are discussed. In Section 4, a suite of flow shop benchmark problems will be given to evaluate the performance of the proposed method. Finally, a conclusion will be drawn in Section 5.

2. Permutation Flow shop scheduling problem formulation

The formulation of the permutation flow shop scheduling problem is described in this section. Suppose there is a set of n jobs to be processed in a set of m machines in the same order. The processing time $p_{j,k}$ for job j $(j = 1, 2, ..., n)$ on machine k $(k = 1, 2, ..., m)$ is given. The objective is to find a sequence for the processing of jobs in the machines to minimize the total completion time or makespan (C_{max}). There are several assumptions on this scheduling problem [31]: 1) Each job j can be processed at most on one machine k at the same time; 2) Each machine m can process only one job j at a time; 3)The processing of a job j on a machine k cannot be interrupted; 4) All jobs are independent and are available for processing at time 0; 5) The setup times of the jobs on machines are negligible; 6) The machines are continuously available; and 7) If the next machine on the sequence needed by a job is not available, the job can wait and join the queue at that machine.

Let $\pi = J_1, J_2, ..., J_n$ denote a permutation of all jobs, and $C(J_j, k)$ denotes the completion time of job J_j on the machine, then the completion time $C(J_j, k)$ can be calculated as follows:

$$C\left(J_1, 1\right) = p_{J_1,1} \tag{1}$$

$$C\left(J_j, 1\right) = C\left(J_{j-1}, 1\right) + p_{J_j,1}, j = 2, ..., n \tag{2}$$

$$C\left(J_1, k\right) = C\left(J_1, k-1\right) + p_{J_1,k}, k = 2, ..., m \tag{3}$$

$$C\left(J_j, k\right) = \max\left\{C\left(J_{j-1}, k\right), C\left(J_j, k-1\right)\right\} + p_{J_j,k}, j = 2, ..., n, k = 2, ..., m \tag{4}$$

$$C_{\max}\left(\pi\right) = C\left(J_n, m\right). \tag{5}$$

So, the flow shop problem with the makespan criterion is to find a permutation π^* in the set of all permutations Π such that,

$$C_{\max}\left(\pi^*\right) \leq C\left(J_n, m\right), \forall \pi \in \Pi \tag{6}$$

3. The hybrid fuzzy-based PSO for flow shop scheduling

3.1. *Solution representation: roulette wheel mechanism*

As mentioned in the introduction section, the existing solution representation method, smallest position value (SPV) rule has a drawback which is to decrease the convergence rate and performance. In this section, a sequence-order system based on the *roulette wheel mechanism* is proposed. In this method, the value of the continuous position of particles represents the probability of the higher ranking. In other words, the position of the higher value of the element of particles will have a higher chance to be select to job sequence list firstly. In this section, the process of the methodology is discussed and an example is given to illustrate the methodology.

3.1.1. *Methodology*

The process of the solution representation with roulette wheel mechanism is described as follows. Fig. 1 shows the roulette wheel mechanism.

(1) Given a continuous position (element) of the particle x_j, $j = 1, 2, \ldots, n$.
(2) Assign a probability q_j to the x_j such that: $q_j = \frac{x_j}{\sum_{j=1}^{n} x_j}$. Note that $\sum q_j$ should equal to 1.
(3) Evaluate a cumulative probability $\hat{q}_j$ for each x_j such as: $\hat{q}_j = \sum_{l=1}^{j} q_l$.
(4) Generate a set of floating-point numbers $d_h \in [0\ 1]$ randomly, $h = 1, 2, ..., \eta$; η is the number of generated floating-point number. These floating point numbers should be sorted with ascending order. Normally, the number of generated floating-point number η equals to $20 \times n$.
(5) Count the total number of the floating-point numbers for each j are in the range of $[\hat{q}_{j-1}\ \hat{q}_j]$ such as: $S_j = count\,(\hat{q}_{j-1} \leq d_h \leq \hat{q}_j)$, where $count()$ is a function to count how many statement is met the condition. Note that $\hat{q}_0 = 0$.
(6) Convert the continuous position of S_j to the sequence of job by using largest position value rule. For a given continuous position like [23, 12, 41, 6], the corresponding sequence is represented by the order of the value. For example, in decent order, 41 is the largest value, so the first of the sequence should be 3; the second largest value is 2, so the second job of sequence should be 1. Thus, we can obtain the corresponding job sequence π is [3, 1, 2, 4].

3.1.2. *Example*

The whole process is illustrated by the following example. The dimension of the particle of PSO is 5. The step-by-step process is listed as follows:

(1) Given a continuous position (element) of particle

$\mathbf{x} = [\mathbf{0.1}, \mathbf{0.7}, \mathbf{0.3}, \mathbf{0.5}, \mathbf{0.4}]$.

(2) Assign a probability from q_1 to q_5: $q_1 = \frac{x_1}{\sum_{j=1}^{5} x_j} = \frac{0.1}{2} = 0.05$, $q_2 = 0.35$, $q_3 = 0.15$, $q_4 = 0.25$, and $q_5 = 0.2$.

(3) Evaluate a cumulative probability $\hat{q}_1$ to $\hat{q}_5$: $\hat{q}_1 = \sum_{l=1}^{1} q_l = 0.05$, $\hat{q}_2 = 0.4$, $\hat{q}_3 = 0.55$, $\hat{q}_4 = 0.8$, and $\hat{q}_5 = 1$.

(4) Generate 100 floating-point numbers ($\eta = 20 \times 5 = 100$), $\mathbf{d}$ = [0.0099, 0.0118, 0.0153, 0.0185, 0.0196, 0.0579, 0.0648, 0.1365, 0.1389, 0.1509, 0.1730, 0.1763, 0.1897, 0.1934, 0.1987, 0.1988, 0.1991, 0.2026, 0.2028, 0.2311, 0.2523, 0.2714, 0.2722, 0.2844, 0.2897, 0.2987, 0.3028, 0.3046, 0.3093, 0.3412, 0.3420, 0.3529, 0.3704, 0.3784, 0.3795, 0.4057, 0.4103, 0.4186, 0.4289, 0.4447, 0.4449, 0.4451, 0.4565, 0.4660, 0.4692, 0.4860, 0.4966, 0.5028, 0.5226, 0.5252, 0.5341, 0.5417, 0.5466, 0.5681, 0.5936, 0.6038, 0.6068, 0.6154, 0.6213, 0.6449, 0.6602, 0.6614, 0.6721, 0.6813, 0.6822, 0.6946, 0.6979, 0.7027, 0.7095, 0.7271, 0.7373, 0.7382, 0.7468, 0.7621, 0.7919, 0.7948, 0.8132, 0.8180, 0.8214, 0.8216, 0.8318, 0.8381, 0.8385, 0.8462, 0.8537, 0.8600, 0.8757, 0.8801, 0.8913, 0.8936, 0.8939, 0.8998, 0.9169, 0.9218, 0.9318, 0.9355, 0.9501, 0.9568, 0.9797, 0.9883].

(5) Evaluate S_1 to S_5: For S_1, there are 5 floating-points numbers are in the range of 0 to 0.05 ($\hat{q}_0$ to $\hat{q}_1$), thus, $S_1 = 5$. For S_2, there are 30 floating-points numbers are within 0.05 to 0.4 ($\hat{q}_1$ to $\hat{q}_2$), thus, $S_2 = 30$. Repeat the process, we have $S_3 = 18$, $S_4 = 23$, and $S_5 = 24$.

(6) Convert the continuous position of $S_1 - S_5$ to the sequence of job. Now, $\mathbf{S}$ = [5, 30, 18, 23, 24]. By using largest position value rule, the job sequence π should be [2 5 4 3 1].

When the element of the particle in PSO is updated, the probability representation is also updated to reflect the order of the job sequence. Considering the previous example, where the continuous position is [0.1, 0.7, 0.3, 0.5, 0.4], by using SPV rule, the smallest position value is 0.1, so the job 1 is assigned to be the first job; the second smallest value is 0.3, so the job 3 is assigned to be the second job and so on. Finally, the job sequence π should be [1 3 5 4 2]. Now, the value of the element of particle is updated in next generation and the updated continuous position is [0.1, 0.7, 0.3, 0.55, 0.35]. By using SPV rule, the job sequence π should be [1 3 5 4 2], which is no different. This kind of inefficiency of the continuous space will result in the decrease in convergence rate and degrade the performance of optimization process. Because of this drawback, the proposed roulette wheel mechanism is used to overcome this drawback and performs faster convergence than SPV rule. With the same example in SPV rule, the updated continuous position is [0.1, 0.7, 0.3, 0.55, 0.35]. Assuming the set of the random number is unchanged. The S_1 to S_5 (in Step (5)) is changed to $\mathbf{S}$ = [5, 30, 18, 27, 20] and finally the job sequence π should be [2 4 5 3 1]. We can see that the job sequence is updated in the second and third

job.

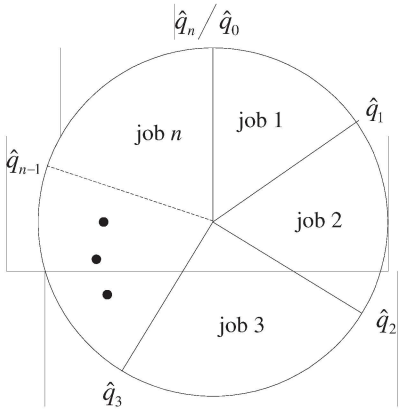


Fig. 1: The roulette wheel mechanism

3.1.3. *Hybrid fuzzy logic-based particle swarm optimization*

Particle swarm optimization (PSO) is a stochastic optimization method which was developed in 1995 [21]. It models the processes of the sociological behaviors associated with the bird flocking and the fish schooling. It uses a number of particles that constitute a swarm. Each particle traverses the search space looking for the global optimum. Recently, an improved PSO is proposed [30] where constriction and inertia weight factors are introduced and the searching ability is improved in comparison with the standard PSO [21]. The process diagram of the PSO with constriction and inertia weight factors (IPSO) is shown in Fig.2. In this paper, a fuzzy logic-based particle swarm optimization with cross-mutated operation namely FPSOCM is proposed to solve flow shop scheduling problem and shown in Fig. 3. The details of both IPSO and FPSOCM will be discussed as follows.

3.1.4. *PSO with constriction and inertia weight factors (IPSO)*

From Fig.2, $X(t)$ is denoted as a swarm at the t-th iteration. Each particle $\mathbf{x}^i(t) \in X(t)$ contains n elements $x_j^i(t) \in \mathbf{x}^i(t)$ at the t-th iteration, where $i = 1, 2, \ldots,$

γ and $j = 1, 2,\ldots, n$; γ denotes the number of particles in the swarm and n is the dimension of a particle. First, the particles of the swarm are initialized and then evaluated by a defined cost (objective) function. The evaluation cost value is represented by $f\left(\mathbf{x}^i(t)\right)$. In the same time, current generation number t is initially set at 0. The objective of PSO is to minimize the cost values of particles iteratively. The swarm evolves from iteration $t+1$ by repeating the processes as shown in Fig.2. The standard PSO [21] operation is discussed as follows.

In PSO, there has one important term and called velocity. The velocity is corresponding to the flight speed in the search space. The velocity $v_j^i(t)$ and the position $x_j^i(t)$ of the j-th element of the p-th particle at the t-th generation can be calculated using the following formulae:

$$v_j^i(t) = 2 \cdot r_1 \cdot \left(p_j^i - x_j^i(t-1)\right) + 2 \cdot r_2 \cdot \left(g_j - x_j^i(t-1)\right) \tag{7}$$

and

$$x_j^i(t) = x_j^i(t-1) + v_j^i(t) \tag{8}$$

where
$p^i = \left[p_1^i\ p_2^i, \ldots p_n^i\right]$ and $g = \left[g_1\ g_2, \ldots g_n\right]$; the best position of a particle i is represented as p^i; the position of the best particle among all the particles is represented as g; r_1 and r_2 return a uniform random number in the range of [0,1]. In [30], an improved version of PSO (IPSO) is presented, where the constriction factor and inertia weight factor are introduced. Here, when the PSO with constriction factor and inertia weight factor is used, (7) will be changed to:

$$v_j^i(t) = k \cdot \left\{\omega(t) \cdot v_j^i(t-1) + \varphi_1 \cdot r_1 \cdot \left(p_j^i - x_j^i(t-1)\right) + \varphi_2 \cdot r_2 \cdot \left(g_j - x_j^i(t-1)\right)\right\} \tag{9}$$

where ω is an inertia weight factor; φ_1 and φ_2 are acceleration constants; k is a constriction factor derived from the stability analysis of equation (9) to ensure the system to be converged but not prematurely [30]. Mathematically, k is a function of φ_1 and φ_2 as reflected in the following equation:

$$k = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \tag{10}$$

where $\varphi = \varphi_1 + \varphi_2$ and $\varphi > 4$.

IPSO utilizes p^i and g to modify the current search point in order to avoid the particles moving in the same direction, but to converge gradually toward p^i and g. A suitable selection of the inertia weight ω provides a balance between the global and local explorations. Generally, $\omega(t)$ is governed by the following equation:

$$\omega(t) = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{T} \times t \tag{11}$$

where t is the current iteration number, T is the total number of iteration, $\omega_{\min}$ and $\omega_{\max}$ are the upper and lower limits of the inertia weight, and normally set to 0.1 and 1.1 respectively [26,?].

In (9), the particle velocity is limited by a maximum value $v_{\max}$. The parameter $v_{\max}$ determines the resolution with which regions are to be searched between the present position and the target position. This limit enhances the local exploitation of the problem space and it realistically simulates the incremental changes of human learning. If $v_{\max}$ is too high, particles might fly past good solutions. If $v_{\max}$ is too small, particles may not explore sufficiently beyond local solutions. Empirically, $v_{\max}$ is often set at 10% to 20% of the dynamic range of the element on each dimension.

A new swarm $X(t)$ is updated after the velocity of all particles are updated. To ensure all particle elements x_j^i in $X(t)$ fall within the range $\left[\rho_{\mathbf{min}}, \rho_{\mathbf{max}}\right]$, the following conditions are considered as follows: If $x_j^i(t) > \rho_{\max_j}$, then the updated $x_j^i(t)$ should equal to $\rho_{\max_j}$. Similarly, If $x_j^i(t) < \rho_{\min_j}$, then the updated $x_j^i(t)$ should equal to $\rho_{\min_j}$. Here, $\rho_{\mathbf{min}} = \left[\rho_{\min_1}\ \rho_{\min_2}\ \cdots\ \rho_{\min_n}\right]$ and $\rho_{\mathbf{max}} = \left[\rho_{\max_1}\ \rho_{\max_2}\ \cdots\ \rho_{\max_n}\right]$; $\rho_{\min_j}$ and $\rho_{\max_j}$ are minimum and maximum values of $x_j^i(t)$ respectively and $j = 1, 2, ..., n$.

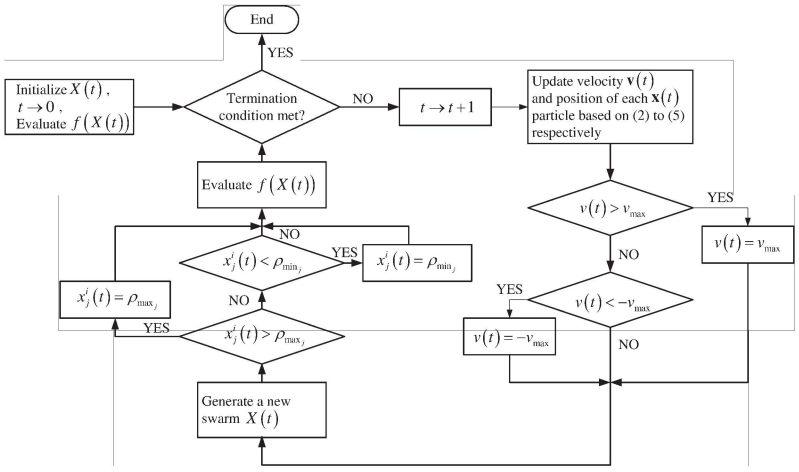


Fig. 2: The process diagram of IPSO

3.1.5. *Fuzzy logic-based PSO with cross-mutated operation*

In this section, the fuzzy logic-based PSO with cross-mutated operation for flow shop scheduling is presented. The process diagram is shown in Fig. 3. In this proposed method, there are two main contributions that will enhance the performance of

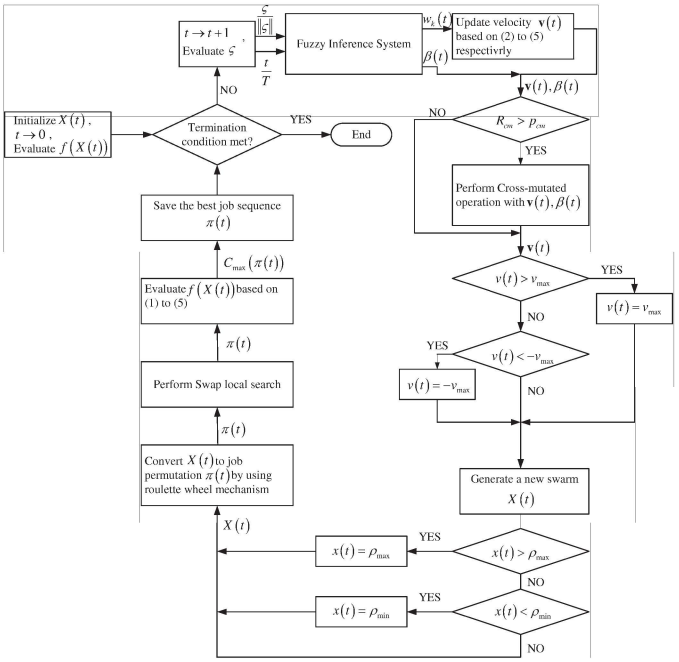


Fig. 3: The process diagram of FPSOCM for flow shop scheduling

searching compared to other PSO methods. Firstly, an adaptive inertia weight $\omega_k(t)$ is proposed to improve the solution quality of searching. Secondly, cross-mutated (CM) operation is given to solve the drawback of IPSO that it is easier to trap into local minima. From this figure, we can see that the adaptive inertia weight $\omega_k(t)$ and the control parameter $\beta(t)$ of CM operation are determined by a fuzzy inference system. The details of the operation of FPSOCM is given in the next subsection.

3.1.6. *Adaptive inertia weight*

In IPSO, the inertia weight is used to provide a balance between the global exploration and local exploitation. A linear inertia weight is governed by the equation (11). From this equation, when the value of t/T is smaller which implies that it is doing global exploration. Similarly, higher value of t/T implies that it is doing fine-tuning (local exploitation). We can see that the characteristic of this function is linear. However, some of the optimization problems are nonlinear. Thus, an adaptive

(nonlinear) inertia weight is proposed to enhance the performance of the searching. In this paper, an adaptive inertia weight $\omega_k(t)$ is proposed, which is incorporated with a fuzzy inference system. In a fuzzy inference system, there are two inputs and two outputs. One of the output is adaptive inertia weight $\omega_k(t)$, the other one is control parameter $\beta(t)$ of CM (It will discussed in later). The inputs of the fuzzy system are $\varsigma(t)/\|\varsigma(t)\|$ and t/T. $\varsigma(t)/\|\varsigma(t)\|$ is a normalized standard deviation of cost value among all the particles. A larger value of $\varsigma(t)/\|\varsigma(t)\|$ implies that the variance between each cost value of particle $f(X^p(t))$ is larger. Conversely, when the value of $\varsigma(t)/\|\varsigma(t)\|$ is smaller, that means the variance between each cost value of particle $f(X^i(t))$ is smaller. In other words, the location of each particles are closer. The formulation of $\varsigma(t)/\|\varsigma(t)\|$ is defined as follows:

$$\varsigma(t) = \sqrt{\frac{1}{\gamma}\sum_{i}^{\gamma}\left(f\left(\mathrm{x}^i(t)\right) - \bar{f}\left(\mathrm{x}^i(t)\right)\right)^2} \tag{12}$$

where

$$\bar{f}\left(\mathrm{x}^i(t)\right) = \frac{1}{\gamma}\sum_{i=1}^{\gamma} f\left(\mathrm{x}^i(t)\right) \tag{13}$$

$\|\cdot\|$ denotes the l_2 vector norm.

The adaptive inertia weight $\omega_k(t)$ is governed by the following fuzzy rules:

$$\text{Rule } j: \text{IF } \varsigma(t)/\|\varsigma(t)\| \text{ is } \mathrm{N}_1^j, \text{ AND } t/T \text{ is } \mathrm{N}_2^j, \text{ THEN } \omega_k(t) = \sigma_j,$$
$$j = 1, 2, \ldots, \varepsilon \tag{14}$$

where N_1^j and N_2^j are fuzzy terms of rule j, ε denotes the number of rules, $\sigma_j \in \left[\omega_{min}\ \omega_{min}\right]$ is the singleton to be determined. In this paper, ω_{min} and ω_{min} are set at 0.1 and 1.1 respectively [26]. The final value of $\omega_k(t)$ is given by:

$$\omega_k(t) = \sum_{j=1}^{\varepsilon} m_j(t)\sigma_j \tag{15}$$

where

$$m_j(t) = \frac{\mu_{\mathrm{N}_1^j}\left(\varsigma(t)/\|\varsigma(t)\|\right) \times \mu_{\mathrm{N}_2^j}(t/T)}{\sum_{j=1}^{r_m}\left(\mu_{\mathrm{N}_1^j}\left(\varsigma(t)/\|\varsigma(t)\|\right) \times \mu_{\mathrm{N}_2^j}(t/T)\right)} \tag{16}$$

$\mu_{\mathrm{N}_1^j}\left(\varsigma(t)/\|\varsigma(t)\|\right)$ and $\mu_{\mathrm{N}_2^j}(t/T)$ are the membership function corresponding to N_1^j and N_2^j respectively. Noting that the value of $\omega_k(t)$ will replace the value of $\omega(t)$ in (9) to produce a new $v_j^i(t)$ with adaptive inertia weight. Thus, the new velocity will be changed to

$$v_j^i(t) = k \cdot \left\{\omega_k(t) \cdot v_j^i(t-1) + \varphi_1 \cdot r_1 \cdot \left(p_j^i - x_j^i(t-1)\right) + \varphi_2 \cdot r_2 \cdot \left(g_j - x_j^i(t-1)\right)\right\} \tag{17}$$

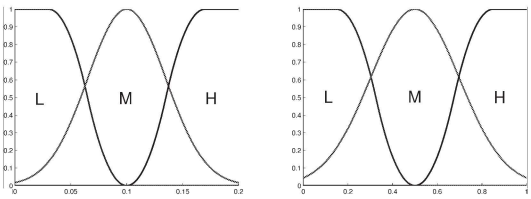
In the proposed algorithm, there are 3 membership functions for each input and shown in Fig. 4. The 3 fuzzy terms are namely L (Low), M (Middle), and H (High). Based on the characteristic of $\varsigma(t)/\|\varsigma(t)\|$ and t/T, there are 9 linguistic IF-THEN fuzzy rules and the fuzzy rule table for determine $\omega_k(t)$ is shown in Fig. 5 (a). The rationale of the selected fuzzy rules for determine $\omega_k(t)$ is described as follows: the value of $\omega_k(t)$ is determined by fuzzy inputs $\varsigma(t)/\|\varsigma(t)\|$ and t/T. The value of t/T represents the iteration stage (smaller value of t/T represents the searching process in the early stage, a larger value of t/T represents the searching process in the later stage). The value of $\omega_k(t)$ should be higher as the value of t/T is smaller (in early stage) implies that a larger value of velocity of particle element is given for global searching. Similarly, a larger value of t/T implies that a higher value of velocity of particle element for local searching. Because of it, the values of $\omega_k(t)$ at t/T is "L", are larger than t/T is set at"M". For the same reason, the values of $\omega_k(t)$ at t/T is "M" are larger than that as t/T is "H". The value of $\omega_k(t)$ should be smaller as t/T increase in order to reduce the value of velocity of particle element for fine-tuning.

As mentioned before, $\varsigma(t)/\|\varsigma(t)\|$ is a normalized standard deviation of cost value among all the particles. A larger value of $\varsigma(t)/\|\varsigma(t)\|$ implies that the variance between each cost value of particle $f(X^i(t))$ is larger. In other words, the location of each particle is far away. At t/T is "L", which the searching process is in early stage and when $\varsigma(t)/\|\varsigma(t)\|$ is "H", that implied the location of each particle is far away in early stage. Thus, a larger value of $\omega_k(t)$ should be given for global exploration. In a special case, when $\varsigma(t)/\|\varsigma(t)\|$ is "L", that implies the location of each particle is closed in early stage.

We also set $\omega_k(t)$ to a larger value, because there has a high chance that the solutions are trap into local optima (it is affected from the smaller value of $\varsigma(t)/\|\varsigma(t)\|$). Therefore, a larger value of $\omega_k(t)$ is given to force the particle escape the local optima. As the value of $\varsigma(t)/\|\varsigma(t)\|$ is "M", we set the value of $\omega_k(t)$ is slight smaller than that $\varsigma(t)/\|\varsigma(t)\|$ is "L" and $\varsigma(t)/\|\varsigma(t)\|$ is "H" in early stage (t/T is "L"). When t/T is "M", which the searching process is in middle stage. The rationale of the suggested value of $\omega_k(t)$ is similar to rules that when When t/T is "L". However, there is one difference that when $\varsigma(t)/\|\varsigma(t)\|$ is "L", the value of $\omega_k(t)$ is smaller than $\varsigma(t)/\|\varsigma(t)\|$ is "H". It is because the optimal solution may be found in middle stage where a smaller value of $\omega_k(t)$ is given. At t/T is "H", which the searching process is in later stage . In this stage, the searching process is undergo fine-tuning process (local exploitation) to find the optimal solution. Because of it, when the value of $\varsigma(t)/\|\varsigma(t)\|$ is "L" that implied the position of most particles are closed and near the best solution. Thus, a smallest value of $\omega_k(t)$ is given.

3.1.7. *Cross-mutated operation*

In this section, a cross-mutated (CM) operation is introduced. This new CM operation is merged the idea of crossover and mutation operation of genetic algorithm. The aim of the CM operation is forcing the particle to escape the local optima.



(a) fuzzy input 1: $\varsigma/\|\varsigma\|$ (b) fuzzy input 2: t/T

Fig. 4: Membership functions a) (x-axis: $\varsigma(t)/\|\varsigma(t)\|$,y-axis: $\mu_{N_1}(\varsigma(t)/\|\varsigma(t)\|)$) and b) (x-axis: t/T ,y-axis: $\mu_{N_2}(t/T)$)

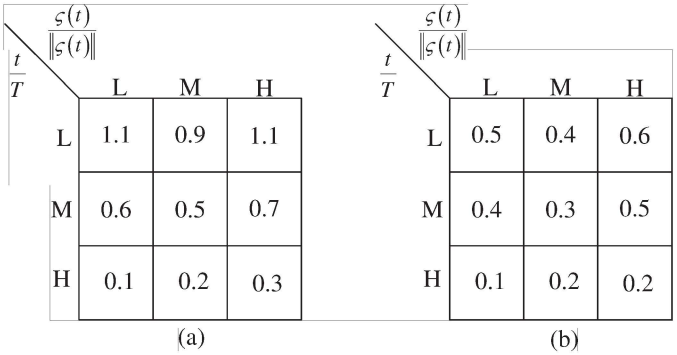


Fig. 5: The fuzzy rule table for determining (a) $w_k(t)$, and (b) $\beta(t)$

Furthermore, a control parameter $\beta(t)$ is introduced into the CM operation, and the operation becomes more adaptive. This control parameter is governed by some fuzzy rules with the human knowledge. By introducing the CM operation, the performance of the proposed PSO method is improved.

The details of the CM operation is as follows. Every velocity of the particle element in the swarm will have a chance to undergo CM operation, which is governed by a probability of cross-mutated operation, $p_{cm} \in [0\ 1]$, defined by the user. For each velocity of the particle element, a random number R_{cm} between 0 and 1 will be generated such that if it is less than or equivalent to p_{cm}, the CM operation will take place on that element. The sensitivity of the p_{cm} with experimental results and the analysis will be discussed later. The choice of the p_{cm} will affect the quality of the solution.

The resulting velocity of the particle element under the CM operation is given by:

$$\bar{v}_j^i(t) = \{1 - \beta(t)\} v_j^i(t) + \beta(t)\tilde{v}_j^i(t) \tag{18}$$

where

$$\tilde{v}_j^i(t) = \frac{1}{2}\left\{\rho_{\max_j} + \rho_{\min_j}\right\} + \frac{1}{4}\left\{r_3 \cdot \left(\rho_{\max_j} + \rho_{\min_j}\right)\right\}. \tag{19}$$

$v_j^i(t)$ is determined by (17), $\tilde{v}_j^i(t)$ is a random velocity of particle element and the value of this velocity is randomly generated and bounded with 0.25 of the dynamic range of the particle element. $r_3 \in \left[-1\ 1\right]$ is an uniform random number. In (18), the resulting velocity of particle element $\bar{v}_j^i(t)$ is combined with the information of $v_j^i(t)$ and $\tilde{v}_j^i(t)$. This information exchanging process is like as crossover operation. However, in CM operation, $\bar{v}_j^i(t)$ is changed (mutated) one-by-one which is like as mutation operation. Therefore, we called it is cross-mutated (CM) operation.

In (18), the control parameter $\beta(t)$ provides a balance to control the resulting velocity $\bar{v}_j^i(t)$ converge toward $v_j^i(t)$ or $\tilde{v}_j^i(t)$. If $\beta(t)$ is approaching 0, the $\bar{v}_j^i(t)$ will tends to the $v_j^i(t)$. Conversely, when $\beta(t)$ is approaching 1,the $\bar{v}_j^i(t)$ will tends to the $\tilde{v}_j^i(t)$. The proposed random velocity $\tilde{v}_j^i(t)$ in (19) has ability to force the particle element to escape the local optima with a random movement. In this operation, the value of the $\beta(t)$ is governed by nine fuzzy-rules. The fuzzy inference system is same as before we mentioned. The inputs of the fuzzy system are same. They are $\varsigma(t)/\|\varsigma(t)\|$ and t/T. Now, there are 9 linguistic fuzzy rules and the fuzzy rule table for determine $\beta(t)$ is shown in Fig. 5 (b).

The rationale of the selected fuzzy rules is similar to the rules of adaptive inertia weight in last sub-section. The brief description of the choosing fuzzy rules for CM operation is given as follows: the value of $\beta(t)$ is determined by $\varsigma(t)/\|\varsigma(t)\|$ and t/T, and some fuzzy rules. As mention before, the value of the t/T represents the iteration stage. We can see as t/T is "L", where the searching process is in early stage and as t/T is "H", where the searching process is in later stage, the values of $\beta(t)$ in early stage are larger than the values of $\beta(t)$ in later stage. It is because a significant random velocity (higher value of $\beta(t)$ in (18)) provides a global exploration in early stage. Conversely, the effect of the random velocity should be reduced in later stage for fine-tuning (local exploitation) with smaller value of $\beta(t)$.

Till now, we have discussed about the effect of the $\varsigma(t)/\|\varsigma(t)\|$, the concept is same as the rules of adaptive inertia weight. In the early stage, when $\varsigma(t)/\|\varsigma(t)\|$ is "L", that implies the location of each particle is closed. Thus, we need to set the value of $\beta(t)$ to be larger than that when $\varsigma(t)/\|\varsigma(t)\|$ is "M". The reason lies in the fact that a higher chance that the solution being trapped into a local optimum. Conversely, in the later stage, the searching process undergoes fine-tuning process (local exploitation) to find the optimal solution. Because of this, when the value of $\varsigma(t)/\|\varsigma(t)\|$ is "L" that implies the position of most particles are closed (similar)

and near the best solution. Thus, a smallest value of $\beta(t)$ is given.

After the CM operation, an updated swarm is generated. Next, convert all the particles in the swarm to job permutation $\pi(t)$ by using the sequence-order system based on the roulette wheel mechanism. This mechanism is presented on section 3.1. After going through the process of the sequence-order system, a set of job sequence will be formed and then undergo swap based local search operation for local exploration on a job permutation space.

3.2. *Swap based-Local search*

Local searches are very important for the exploitation. In this paper, a swap-based local search method is designed for fine-tuning on the flow shop scheduling problem. The operation of this local search is simply to swap the selected two job order number. The operation of swap based local searching method is described as follows.

(1) Given a sequence of jobs π, where $\pi = J_1, J_2, ..., J_{i-1}, J_i, J_{i+1}, ...J_n$.
(2) Randomly picks 2 jobs, i.e., J_i and J_{i+1} .
(3) Swap the selected 2 jobs.

After the process, the new sequence of jobs should be J_1, J_2, ..., J_{i-1}, J_{i+1}, J_i, ...J_n. An example of swap based-local search method is given in Fig.6. In this figure, the job 7 and the job 1 are selected to undergo the swap process. After going through the swap operation, a new job sequence is generated. By using (1)-(5), a updated makespan C_{max} will be evaluated. The optimization process will repeat until a termination condition is met. The iterative process will be terminated when a defined number of iteration is met.

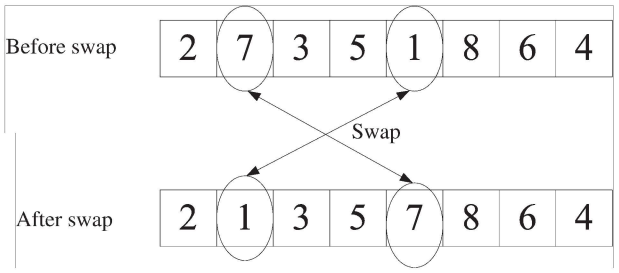


Fig. 6: The example of the swap-based local search method.

4. Computational results

To test the performance of the proposed FPSOCM to flow shop scheduling problem, computational simulation is carried out with a suite of flow shop benchmark problems [32]. In this paper, 8 benchmark problems [32] from OR-Library are given.

4.1. *Experimental setup*

The performance of HPSOWM [26], HPSOM [7], HGAPSO [25], HGPSO [24], IPSO [30], and the proposed FPSOCM on solving the flow-shop benchmarking problems are evaluated.

The following simulation conditions are in use:

For All PSOs:

- Swarm size (γ): 100
- Number of runs: 20
- Acceleration constant φ_1: 2.05
- Acceleration constant φ_2: 2.05
- Maximum velocity v_{max}: 0.2
- Initial population: it is generated uniformly at random

For FPSOCM:

- Probability of cross-mutated operation (p_{cm}):
  p_{cm} = 0.05 for Car3, Car8;
  p_{cm} = 0.01 for Car1, Car2, Car4, Car5, Car6, Car7,;

For HPSOWM, HPSOM, and HGAPSO:

- Probability of mutation operation (p_m): it is chosen by trial and error through experiments for good performance for all functions. (p_m= 0.2 for Car1 - Car8)

For HPSOWM:

- Shape parameter of the wavelet mutation: it is chosen by trial and error through experiments for good performance for all functions
- Parameter g of the wavelet mutation: 10000

For HGPSO:

- Learning rate: it is chosen by trial and error through experiments for good performance for all functions

For HGAPSO:

- Probability of crossover operation: 0.8

4.2. *Results*

In this section, the results for the 8 flow shop problem are given to validate the FPSOCM, which is proven to perform well on flow shop scheduling problem. The experimental results in terms of the average relative error (ARE), best relative error (BRE), and worst relative error (WRE) are summarized in Table 1. Relative error (RE) is defined as $(C^* - C_{\max})/C^*$ where C^* is the optimal value known so far and $C_{\max}$ is the value given by the algorithm. ARE is the average relative error of the 20 runs of the result, which is given by ARE=$\sum_{i=1}^{20} RE_i$. BRE and WRE are the best RE and worst RE among all the run respectively. The number of iterations 100 is adopted in all approaches.

From the table, we can see that FPSOCM algorithm obtains a smaller ARE, BRE and WRE in all problems. In problem Car6 which is a 8 jobs 9 machines flow shop scheduling problem, the ARE of the FPSOCM is 0. Comparing with other algorithms, only FPSOCM can reach the optimal value. In Problem Car3, the ARE of FPSOCM is 0.06 which imply around 2.5-6 times improvement compared with other algorithm. Similarly, in Problem Car5, the ARE of FPSOCM is 0.019 which imply around 8-18 times improvement.

To illustrate the solution stability achieved by the algorithms, the standard deviations (STD) of the results obtained by all algorithms are shown in Table 1. A smaller standard deviation implies that the algorithm produces a more stable solution. If the algorithm produces a better mean value and smaller standard deviation, it implies that this algorithm can obtain solutions with better performance in terms of solution quality and solution stability. From Table 1, all the algorithms can found the optimal value (BRE=0). However, comparing with the other algorithms and the proposal FPSOCM, the ARE and STD obtained by the other algorithms are worse than those obtained by the FPSOCM. By unitizing the proposed fuzzy inference system into the FPSOCM, the inertia weight, which is governed by a set of fuzzy rules, can be adapted with the diversity of the positions of the particles. Therefore, unlike the other algorithms, the proposed fuzzy inference system aids the FPSOCM to generate more stable solutions. Apart from this mechanism, the cross-mutated operation is proposed to unitize to the proposed FPSOCM. The cross-mutated operation is governed by some fuzzy rules, which effectively force the solution to escape the local minimum. With the proposed adaptive initial weight and the proposed cross-mutated operation, the FPSOCM performs better in terms of solution quality (best mean value) and solution stability (smallest standard deviation) than the other tested algorithms. As conclude, FPSOCM algorithm for flow shop scheduling problem to minimize makespan is more effective.

In order to demonstrate the statistical significance of differences between the results obtained by the proposed algorithm and the other tested algorithms, t-tests were conducted. A t-test table are summarized in Table 2. The t-test is a statistical method to evaluate the significant difference between two algorithms. The t-value will be negative if the first algorithm is better than the second, and positive if it is

(A)

Method		Problem			
		Car1	Car2	Car3	Car4
		$n, m = 11, 5$; $C^* = 7038$	$n, m = 13, 4$; $C^* = 7166$	$n, m = 12, 5$; $C^* = 7312$	$n, m = 14, 4$; $C^* = 8003$
FPSOCM	ARE	0	0	0.06	0
	BRE	0	0	0	0
	WRE	0	0	0.602	0
	STD	0	0	0.185	0
HPSOWM	ARE	0	0	0.157	0
	BRE	0	0	0	0
	WRE	0	0	1.203	0
	STD	0	0	0.329	0
HPSOM	ARE	0	0	0.260	0
	BRE	0	0	0	0
	WRE	0	0	1.504	0
	STD	0	0	0.520	0
HGAPSO	ARE	0	0	0.356	0
	BRE	0	0	0	0
	WRE	0	0	1.504	0
	STD	0	0	0.555	0
HGPSO	ARE	0	1.036	0.345	0.012
	BRE	0	0	0	0
	WRE	0	2.930	1.504	0.062
	STD	0	1.357	0.519	0.025
IPSO	ARE	0	0	0.343	0
	BRE	0	0	0	0
	WRE	0	0	1.792	0
	STD	0	0	0.556	0

(B)

Method		Problem			
		Car5	Car6	Car7	Car8
		$n, m = 10, 6$; $C^* = 7720$	$n, m = 8, 9$; $C^* = 8505$	$n, m = 7, 7$; $C^* = 6590$	$n, m = 8, 8$; $C^* = 8366$
FPSOCM	ARE	0.019	0	0	0
	BRE	0	0	0	0
	WRE	0.389	0	0	0
	STD	0.087	0	0	0
HPSOWM	ARE	0.161	0.153	0	0124
	BRE	0	0	0	0
	WRE	0.622	0.764	0	1.960
	STD	0.234	0.314	0	0.447
HPSOM	ARE	0.282	0.115	0	0
	BRE	0	0	0	0
	WRE	1.308	0.764	0	0
	STD	0.384	0.280	0	0
HGAPSO	ARE	0.549	0.076	0	0.152
	BRE	0	0	0	0
	WRE	1.308	0.764	0	1.960
	STD	0.422	0.235	0	0.457
HGPSO	ARE	0.194	0.223	0	0.294
	BRE	0	0	0	0
	WRE	0.609	0.764	0	1.960
	STD	0.213	0.359	0	0.718
IPSO	ARE	0.343	0.483	0	0.144
	BRE	0	0	0	0
	WRE	1.308	2.787	0	1.960
	STD	0.430	0.663	0	0.451

n: Number of jobs

m: Number of machines

C^*: The optimal value known so far

Table 1: Comparison results of FPSOCM, HPSOWM, HPSOM, HGAPSO, HGPOS, and IPSO. All results are averaged ones over 20 runs (A: Car1-4, B: Car5-8).

	HPSOWM	HPSOM	HGAPSO	HGPSO	IPSO
Car1	N/A	N/A	N/A	N/A	N/A
Car2	N/A	N/A	N/A	-3.4149	N/A
Car3	-1.1455	-1.6169	-2.2588	-2.0783	-2.1567
Car4	N/A	N/A	N/A	-2.1795	N/A
Car5	-2.5393	-2.9813	-5.4946	-2.4724	-3.3000
Car6	-2.1795	-1.8311	-1.4530	-2.1498	-3.2590
Car7	N/A	N/A	N/A	N/A	N/A
Car8	-1.2393	N/A	-1.4885	-1.5540	-1.4282

Table 2: t-value between FPSOCM and the others PSO methods (N/A means both algorithms can found the optimal result).

poorer. The t-value is defined as: $t = \frac{\bar{\alpha_1} - \bar{\alpha_2}}{\sqrt{\sigma_1^2/(\xi+1) + \sigma_2^2/(\xi+1)}}$ where $\bar{\alpha_1}$ and $\bar{\alpha_2}$ are the mean value of the first method and the second method respectively, σ_1 and σ_2 are the standard deviations of the first method and the second method respectively, ξ is the degree of freedom.

When the t-value is smaller than -1.33 (degree of freedom = 19), there is a significant difference between the two algorithms with a 80% confidence level. When the t-value is smaller than -2.09 and -2.86 (degree of freedom = 19), there is a significant difference between the two algorithms with a 95% and 99% confidence level. From the table, we can see that FPSOCM is statistically significant. Firstly, all the t-values are negative which implied that the FPSOCM is better than the others. Secondly, most of the t-values are smaller than -1.33 which implied that the proposed algorithm is a significant with a 80% confidence level and some of the t-values are smaller than -2.09 and -2.86 which implied that the FPSOCM is a significant with a 95-99% confidence level.

To further present the performance of the proposed FPSOM, a comparison was made between the proposed algorithm with results of SGA, PM-NONEH, SGA+NEH and EM listed in the work [34] [35] [36]. The results are listed in Table 3. As shown, FPSOM with its 20 executions, has less ARE than the other concurrent algorithms, according to this further test, the proposed FPSOM with wave mutation outperforms conventional algorithms with regard to the accuracy.

5. Conclusion

In this paper, we have proposed a hybrid particle swarm optimization which incorporates a fuzzy system and a new cross-mutated operation to tackle job shop scheduling problem. An adaptive inertia weight of PSO is presented. It is determined by a fuzzy system. Moreover, in order to solve the local optimum problem of PSO, a new operation namely cross-mutated operation is proposed. With these

Problem	SGA	PM-NONEH	SGA+NEH	EM	FPSOM
	ARE	ARE	ARE	ARE	ARE
Car1	0.27	0	0	0.21	0
Car2	4.07	0	2.93	2.55	0
Car3	2.95	0.17	1.21	2.19	0.06
Car4	2.36	0	0.07	1.95	0
Car5	1.46	0.045	1.14	1.27	0.019
Car6	1.86	0.038	2.82	1.34	0
Car7	1.57	0	1.36	1.12	0
Car8	2.59	0.032	0.03	1.05	0

Table 3: Comparison results on ARE of SGA, PM-NONEH, SGA+NEH, EM and FPSOCM.

proposed operations, the solution quality is improved. The performance of the proposed hybrid PSO is compared with other hybrid PSOs with respect to the same set of flow scheduling problems. The results demontrate the effectiveness of the proposed algorithm for the flow shop scheduling problem.

References

1. C. R. Reeves, A genetic algorithm for flowshop sequencing,*Comput. Oper. Res.* **22 (1)** (1995) 5-13.
2. G. I. Zobolas, C. D. Tarantilis and G. Ioannou, Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm,*Computer and operations research* **36** (2009) 1249-1267.
3. J. S. Chen, J. C. H. Pan and C. K. Wu, Hybrid tabu search for re-entrant permutation flow-shop scheduling problem, *Expect System with Application* **34** (2008) 1924–1930.
4. G. Onwubolo and D. Davendra, Scheduling flow shops using differential evolution algorithm,*Eur. J. Oper. Res.* **171** (2006) 6740-692.
5. C. Rajendran and H. Ziegler, Two ant-colony algorithms for minimizing total flowtime in permutation flowshops,*Comput. & Ind. Engg.* **48** (2005) 789-797.
6. K.C. Ying and C. J. Liao, Aa ant colony system for premutation flow-shop sequencing, *Comput. Oper. Res.* **31** (2004) 791-801.
7. A. A. E. Ahmed, L. T. Germano and Z. C.Antonio, A hybrid particle swarm optimization applied to loss power minimization, *IEEE Trans. Power Syst.* **20 (2)** (2005) 859-866.
8. J. Hazra and A. K. Sinha, Congestion Management Using Multiobjective Particle Swarm Optimization, *IEEE Trans. Power Syst.* **22 (4)** (2007) 1726-1734.
9. J. G. Vlachogiannis and K. Y. Lee, A comparative study on particle swarm optimization for optimal steady-state performance of power systems, *IEEE Trans. Power Syst.* **21 (4)** (2006) 1718-1728.
10. A. Chatterjee, K. Pulasinghe, K. Watanabe and K. Izumi, A particle-swarm-optimized fuzzy-neural network for voice-controlled robot systems, *IEEE Trans. Ind. Electron.* **52 (6)** (2005) 1478-1489.

11. C. F. Juang, A hybrid genetic algorithm and particle swarm optimization for recurrent network design, *IEEE Trans. Syst. Man and Cybern. B* **34 (2)** (2004) 997-1006.
12. Y. Song, Z. Chen and Z. Yuan, New chaotic PSO-based neural network predictive control for nonlinear process, *IEEE Trans. Neural Networks* **18 (2)** (2007) 595-601.
13. S. L. Ho, S. Y. Yang, G. Z. Ni and K. F. Wong, An improved PSO method with application to multimodal functions of inverse problems, *IEEE Trans. Magnetics* **4** (2007) 1597-1600.
14. R. Marinke, E. Araujo, Ld. S. Coelho and I. Matiko, Particle swarm optimization (PSO) applied to fuzzy modeling in a thernal-vacuum system, in *Proc. 5th Int. Conf. Hybrid Intelligent Systems*, (Rio de Janeiro, Brazil, 2005) 67–72.
15. Z. Lian, X. Gu and B. Jiao, A novel particel swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan, *Chaos, Solitons and Fractals* **35** (2008) 851-861.
16. B. Liu, L. Wang and Y. H. Jin, An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers, *Comput. Oper. Res.* **35** (2008) 2791-2806.
17. B. Liu, L. Wang and Y. H. Jin, An effective PSO-based memetic algorithm for flow shop scheduling, *IEEE Trans. Syst. Man and Cybern. B*, **37** (2007) 18-27.
18. K. Sun and G. Yang, An effective hybrid optimization algorithm for the flow shop scheduling problem, *Proc. 30th IEEE Conf. Decision and Control*, (New Orleans, USA, 1995) 1942–1948.
19. M. F. Tasgetiren and Y. C. Liang and M. Sevkli and G. Gencyilmaz, A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem, *Eur. J. Oper. Res.*, **177** (2007) 1930-1947.
20. L. Yuan and Z. D. Zhao, A modified binary particle swarm optimization algorithm for permutation flowshop problem, *Proc. 6th Int. Conf. on Machine Learing and Cybernetics*, (Hong Kong, 2007) 902–907.
21. J. Kennedy and R. Eberhart, Particle swarm optimization, *Proceedings of IEEE International Conference on Neural Networks*, (Piscataway, NJ, 1995) 1942–1948.
22. J. Kennedy, R. Eberhart and Y. Shi, *Swarm Intelligence* (Morgan Kaufmann Publishers, San Fransisco, USA, 2001)
23. P. Angeline, Using selection to improve particle swarm optimization, *Proc. IEEE Congress on Evolutionary Computing*, (Anchorage 1998) 84–89.
24. M. M. Noel and T. C. Jannett, Simulation of a new hybrid particle swarm optimization algorithm, *Proc. 36th Southeastern Symposium on System Theory*, (Atlanta, USA, 1994) 150–153.
25. C. F. Juang, A hybrid genetic algorithm and particle swarm optimization for recurrent network design, *IEEE Trans. Syst. Man and Cybern. B*, **34 (2)** (2004) 997-1006.
26. S. H. Ling, H. H. C. Iu, K. Y. Chan, H. K. Lam, C. W. Yeung and F. H. F. Leung, Hybrid particle swarm optimization with wavelet mutation and its industrial applications, *IEEE Trans. Syst. Man and Cybern. B*, **38 (3)** (2008) 743-763.
27. S. H. Ling, H. H. C. Iu, F. H. F. Leung, and K. Y. Chan, Improved hybrid PSO-based wavelet neural network for modelling the development of fluid dispensing for electronic packaging, *IEEE Trans. Industrial Electronic*, **55 (9)** (2008) 3447-3460.
28. B. A. Norman and A. E. Smith, Random keys genetic algorithm with adaptive penalty function for optimization of constrained facility layout problems, *Proc. IEEE Int. Conf. on Evolutionary Computation 1997*, (1997) 407–411.
29. S. Xu and J. C. Bean, A genetic algorithm for scheduling parallel non-identical batch processing machines, *Proc. IEEE Symposium on Computational Intelligence in Scheduling*, (Honolulu, Hawaii, 2007) 143–150.
30. R. C. Eberhart and Y. Shi, Comparing inertia weights and constriction factors in

particle swarm optimization, *Proc. IEEE Congress on Evolutionary Computing*, (2000) 84–88.
31. Baker, *Introduction to Sequencing and Scheduling* (John Wiley and Sons, New York, 1974)
32. J. Carlier, Ordonnancements a contraintes disjonctives, *R.A.I.R.O. Recherche Operationelle/Oper. Res.*, **12 (4)** (1978) 333-351.
33. C. R. Reeves, A genetic algorithm for flowshop sequencing, *Computers and Operations Research,* **22 (1)**, (1995) 5–13.
34. H. Liu, L. Gao, Q. Pan, A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem, *Journal Expert Systems with Applications,* **38 (4)**, (2011), 4348-4360.
35. L. Wang and D. Z. Zheng, An effective hybrid heuristic for flow shop scheduling, *Int. J. Adv. Manuf. Technol.,* **21 (1)**, (2003), 38-44.
36. L. Bo, et al., An Effective PSO-Based Memetic Algorithm for Flow Shop Scheduling, *IEEE Transactions on Systems, Man, and Cybernetics, Part B:* **vol. 37,** (2007), 18-27.