Hybrid Fuzzy Adaptive Particle Swarm Optimization Algorithm for Fuzzy Job Shop Scheduling Problem (FJSSP)

Aylin Pakzad
Department of industrial Engineering
Shahid Bahonar University of Kerman Jomhouri Boulevard
Kerman, Iran

Malek Tajadod
Department of industrial Engineering
Shahid Bahonar University of Kerman Jomhouri Boulevard
Kerman, Iran

ABSTRACT
The processing time for each job in JSSP is often imprecise in many real world applications. Therefore, JSSP with fuzzy processing time was addressed in this paper. Triangular fuzzy numbers were used to describe the fuzzy processing time. In this paper, a hybrid particle swarm optimization (HPSO) algorithm was presented for solving JSSP. The quality of the PSO algorithm final solution depends on two factors: the quality of initial solutions and adjustment of PSO parameters. In this study, to improve the quality of initial solutions, a constructive greedy randomized adaptive search procedure (GRASP) algorithm was proposed. Furthermore, in order to adjust HPSO parameters, a fuzzy interference system was applied to compute these parameters at each iteration of HPSO. Therefore, the presented algorithm in this study was called hybrid fuzzy adoptive PSO (HFAPSO). Benchmarks with fuzzy processing time were used for testing the presented algorithm.

General Terms
Fuzzy job shop scheduling problem, Greedy randomized adaptive search procedure, Hybrid Particle Swarm Optimization.

Keywords
Fuzzy interference system, Fuzzy job shop scheduling problem, Greedy randomized adaptive search procedure, Hybrid Particle Swarm Optimization.

1. INTRODUCTION
The job-shop scheduling problem (JSP) has been studied for more than 50 years in both academic and industrial environments. Jain and Meeran [1] provided a concise overview of JSPs over the last few decades and highlighted the main techniques. Garey et al [2] demonstrated that JSPs are NP-hard; hence finding an exact solution in a reasonable computational time is impossible, so using evolution algorithm is unavoidable. In this reason, many researchers used evolution algorithm to solve the problems, such as genetic algorithm [3], simulated annealing [4] and particle swarm optimization [5]. According to last researches in JSSP, hybrid evolution algorithms had better result than evolution algorithms. Some new valid algorithm operators are proposed. Particle swarm optimization combined with simulated annealing is used to find the minimum makespan in the job shop scheduling environment [6]. One stream of research investigated the job shop scheduling problem where uncertainty spotted Such as: The first significant application that considers the uncertainty in time parameters proposed by P. Fortemps [7]. Some approaches specifically consider fuzzy processing time. Deming Lei [8], proposed Fuzzy job shop scheduling problem (FJSSP) with availability constraints.

In this paper, we focus on constructing a hybrid particle swarm optimization algorithm to achieve the better solution for FJSSP. This algorithm contains GRASP as constructive algorithm that constructs initial solution for PSO algorithm. Also to make suitable FJSSP in practice, we suppose that all of the processing times are uncertain (fuzzy) and for better adjustment of the parameters in PSO algorithm, fuzzy adaptive PSO has been used. The remaining sections of this paper are organized as follows: section 2 describes methodology. For solving the objective function of FJSSP, some related fuzzy set operations are proposed in section 2.1. GRASP algorithm for constructing initial solution of PSO algorithm is described in section 2.2. In section 2.3, we present PSO algorithm and adjusted its parameters with fuzzy interference system. Hybrid Fuzzy Adoptive PSO algorithm is presented in section 2.4. Computational result and conclusions come in sections 3 and 4 respectively.

2. METHODOLOGY
2.1 Related Fuzzy Set Operations
The processing time for each job in JSSP is often imprecise in many real world applications. Therefore, job shop scheduling problem with fuzzy processing time is addressed in this paper. The processing time is described by triangular fuzzy numbers. The objective of FJSSP is finding a schedule to minimize the makespan ($C_{\max}$). Our objective function for jth job on ith machine consists of two parts: obtaining the maximum of two fuzzy numbers (completion time of jth job on (i-1)th machine and completion time previous job on machine i), and processing time of job j on machine i. For computing the objective function (minimum makespan), we should minimize the summation of these two parts.

Let (a,b,c) and (d,e,f) are triangular fuzzy numbers. In this paper, for approximation the maximum (v) of two triangular fuzzy numbers, upper approximation in the shape of trapezoid fuzzy number was used as shown in equation (1):

$$(a,b,c) \vee (d,e,f) = (\min(a,d), \min(b,e), \max(b,e), \max(c,f)) \quad (1)$$

2.2 GRASP Algorithm
Till now we have described FJSSP and said some related fuzzy set operations, in this section we explain GRASP algorithm that is used in our hybrid PSO algorithm. PSO is evolutionary algorithm and the quality of initial solution has

significant role in quality of ultimate solution. The initial solutions for classic PSO algorithm were determined randomly, but here the GRASP algorithm proposed to construct them. The proposed GRASP algorithm has 7 steps which are describing as follow:

Step1: Use $_J p_j = \{M_{1j} = i_1, M_{2j} = i_2, \ldots, M_{mj} = i_m\}$ that defines the job j process flow, as input data for making job-path matrix that the rows of it show jobs and the column of it shows machines. In other words, every set of $_J p_j$ comes in rows of matrix.

Step2: Sum the set up and process times for all jobs. Sort jobs on every machine based on SSPT (shortest setup and processing time first) rule and show the rank of every job with $SSPT_{ij}$, (if equivalent takes place in time for two jobs on one machine, give equivalent rank). After that, define $LSPT_{ij}$ (regarding LSPT (longest setup and processing time first) rule) in this shape $LSPT_{ij} = SSPT_{i(nej+1)}$ for all i and j in $SSPT_{ij}$. Mahmodi Nejad and Mashinchi ranking method [9] is used to rank the fuzzy sets and transform them to crisp numbers.

Step3: Compute f_{ij} for j= 1,2,…,n and i= 1,2,…,m. (every jobs has lesser f_{ij}, has more priority for processing on machine i):

If $jobpath_{ji} = 0$, put f_{ij} a large number and If $jobpath_{ji} \neq 0$, compute f_{ij} according to relation (2). In relation (2), w_1, w_2, w_3, are weight coefficient and adjust the importance of elements in it.

$$f(o_{ij}) = W_1 M_{ij} + w_2 SSPT_{ij} + W_3 LSPT_{ij} \quad (2)$$

Step4: Put all jobs in the list that called A(i) and sort them based on ascendant regarding f_{ij}, then put γ percentage from beginning of A(i) in a list called B(i), the jobs in B(i) are the best candidate jobs for allocating to machine i. then put s:=1, t:=1.

Step5: Put the amount of two list A(i) and B(i), in two new list $A'(i)$ and $B'(i)$ .then put i:=1 and p:=1.

Step6: If $B'(i)$ is not been emptied yet, randomly select operation p of machine i from $B'(i)$, otherwise select the job that has minimum amount of f_{ij} in $A'(i)$ - $B'(i)$. Eliminate the operation (job) was selected from $B'(i)$ or $A'(i)$ - $B'(i)$. Then survey these conditions:

- If i = m and p < n, put i =1 and p = p+1 and repeat step6.
- If i < m and p<=n, put i = i +1 and repeat step6.
- If i = m and p:= n, go to next step.

Step7: Save the computed solution as S_s and convert it to an active schedule. This conversion is done by exact G&T algorithm was presented by Giffler and Thompson's heuristic [10], and compare the objective value of it with the best objective value that achieved so far. If $C_{\max}(s_s) < C_{\max}(s_s^*)$, replace it with previous S^*. Then if S < N_s and t <= nIter, put S = S+1 and go to step 5. N_s is the number of solution that is required and nIter is the number of required repetition of algorithm. If S = N_s and t < nIter, put s: =1 and t = t+1 and go to step 5 and if S = N_s and t = nIter, stop.

2.3 Fuzzy Adaptive Particle Swarm Optimization Algorithm

Particle swarm optimization (PSO) is a novel evolutionary algorithm that was proposed by Kennedy and Eberhart for [11]. In classic PSO, the PSO parameters are often held constant or linearly changed for the entire run of a PSO, although this approach will not produce optimal results in many cases.in this study, a fuzzy system is applied to adjust the inertia weight and learning factors with best fitness (BF) and number of generations for unchanged best fitness (NU) as the input variables, and the inertia weight (w) and learning factors (c_1 and c_2) as output variables. In this study, to design a FAPSO applicable to a wide range of problems, the ranges of BF and NU are normalized into [0, 1.0].

The model of FAPSO is shown in Figure 1. The fuzzy system consists of four principal components: fuzzification, fuzzy rules, fuzzy reasoning and defuzzification, which are described as following.

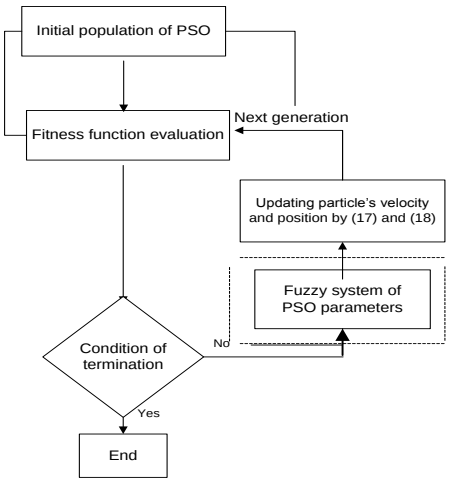

Figure 1: The Model of FAPSO

2.3.1 Fuzzification

Triangle membership functions are used for every input and output as illustrated in Figure 2,3 and 4 PS (positive small), PM (positive medium), PB (positive big) and PR (positive bigger) are the linguist variables for the inputs and outputs.

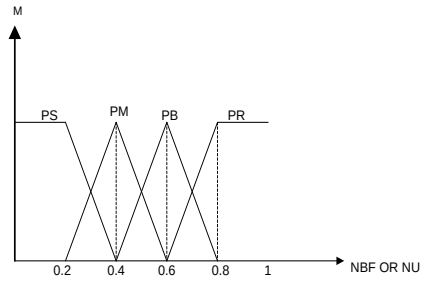

Figure 2: Membership Functions of Inputs and Outputs NBF or NU

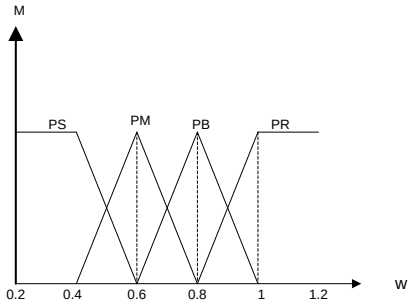

Figure 3: Membership Functions of Inputs and Outputs w

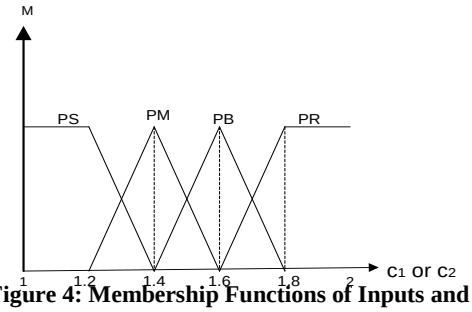

Figure 4: Membership Functions of Inputs and Outputs $c_1$or c_2

2.3.2 Fuzzy Rules

The Mamdani-type fuzzy rule is used to formulate the conditional statements that comprise fuzzy logic. For example

R_i: IF (NBF is PB) and (NU is PM), THEN (w is PB), (c_1 is PM) and (c_2 is PM)

The fuzzy rules in Tables 1–3 are used to adjust the inertia weight (w) and learning factors (c_2 and c_1), respectively. Each rule represents a mapping from the input space to the output space.

Table 1: Fuzzy Rules for Inertia Weight

	w	NU			
		PS	PM	PB	PR
NBF	PS	PS	PM	PB	PB
	PM	PM	PM	PB	PR
	PB	PB	PB	PB	PR
	PR	PB	PB	PR	PR

Table 2: Fuzzy Rules for Learning Factor C_2

C_2		NU			
		P S	P M	P B	P R
NB	P	P	P	P	P
F	S	R	B	M	M
	P	P	P	P	P
	M	B	M	S	S
	P	P	P	P	P
	B	M	M	S	S
	P	P	P	P	P
	R	M	S	S	S

Table 3: Fuzzy Rules for Learning Factor C_1

C_1		NU			
		P S	P M	P B	P R
NB	P	P	P	P	P
F	S	R	B	B	M
	P	P	P	P	P
	M	B	M	M	S
	P	P	P	P	P
	B	B	M	S	S
	P	P	P	P	P
	R	M	M	S	S

2.3.3 Fuzzy Reasoning

The fuzzy control strategy is used to map from the given inputs to the outputs. Mamdani's fuzzy inference method is used in this paper. The AND operator is typically used to combine the membership values for each fired rule to generate the membership values for the fuzzy sets of output variables in the consequent part of the rule. In addition, the various output fuzzy sets derived from different fired rules are aggregated into a single output fuzzy set by OR operator.

To obtain a deterministic control action, a defuzzification strategy is required. It will be illustrated at a later point.

2.3.4 Defuzzification

The method of centroid (center-of-sums) is used for defuzzification. Defuzzified value is directly acceptable values of PSO parameters.

In brief, the fuzzy system is an effective tool to represent and utilize human knowledge that is too complex for mathematical approaches.

2.4 Hybrid Fuzzy Adaptive Particle Swarm Optimization Algorithm (HFAPSO) for FJSSP

Hybrid FAPSO algorithm is summarized in 10 steps which are describing as follow:

Step 1: Receive data and parameters. Provide the list of 0, 1, 2, …, (n-1), and call it F. provide two others list from 1, 2, …, n for every machine i, that every numbers demonstrates one job. Call these lists P_i and Pi'.

Step 2: With using GRASP algorithm was described in section 2.2, construct initial solution in the measure of popsize.

Step 3: Suppose every achieved sequences of GRASP on machine i, as permutation and convert the sequence of jobs

on every machines in initial solution to Factoradic number with using Factoradic base [12]. Then create m-dimension vector from non-negative integer number. This vector demonstrates particle position in algorithm. After creating m-dimension vector for every solution (r), put it as the first vector of particle position as equation (3).

$$x_{r1}^{k} = \{x_{r1}^{i}, x_{r1}^{i}, \ldots, x_{rD}^{i}\}, \quad \forall r = 1, 2, \ldots, popsize \qquad (3)$$

Step 4: Create randomly m-dimension vector from integer number in $[0 - x_{ri}^{1}, (n!-1) - x_{ri}^{1}]$, for every solution (r), put it as the first vector of particle velocity as equation (4). Then go to step 6.

$$v_{r1}^{k} = \{v_{r1}^{i}, v_{r1}^{i}, \ldots, v_{rD}^{i}\}, \quad \forall r = 1, 2, \ldots, popsize \qquad (4)$$

Step 5: Update ith dimension of every particle velocity vector in kth iteration with relation (5).

$$v_{ri}^{k} = w * v_{ri}^{k} + c_1 * rand_1 * \left[pbest_{ri}^{k} - x_{ri}^{k} \right] + c_2 * rand_2 * \left[gbest_{gi}^{k} - x_{ri}^{k} \right]$$
$$\forall i = 1, 2, \ldots, m \quad \forall r = 1, 2, \ldots, popsize \qquad (5)$$

k is the number of the iteration, w is inertial weight, if v_{ri}^{k} be in $[0 - x_{ri}^{1}, (n!-1) - x_{ri}^{1}]$, go to step 6. Otherwise, if v_{ri}^{k} is less than $(0 - x_{ri}^{1})$, put: $v_{ri}^{k} = rand \times (0 - x_{ri}^{k})$ and if v_{ri}^{k} is more than $((n!-1) - x_{ri}^{1})$, put : $v_{ri}^{k} = rand \times ((n!-1) - x_{ri}^{k})$. Note that $rand$ is random number in $[0,1]$.Then go to step 6.

Step 6: Update ith dimension of every particle position vector in kth iteration with relation (6).

$$x_{ri}^{k+1} = [x_{ri}^{k} + v_{ri}^{k}], \forall i = 1, 2, \ldots, m, \forall r = 1, 2, \ldots, popsize \qquad (6)$$

k is the number of iteration and x_{ri}^{k+1} is new vector of particle position r. Then go to next step.

Step 7: At first put all amount of P_i in Pi′. consider the position of every particle. Convert ith dimension of Pi′ to Factoradic number (i =1, 2, …, m). Now, convert the Factoradic number to permutation regarding to F and Pi′. Then constitute the solution of every particle with these permutations for every machine. This solution is priority base that should be converted to active schedule. This conversion is done by G&T algorithm [10]. Then calculate objective value, pbest and gbest for every particle and save all yield active schedule in S_r and go to step 8.

Step 8: Map the position of each particle into the solution space and evaluate its fitness value according to the desired optimization fitness function. Simultaneously update the pbest and gbest positions if necessary.

Step 9: Regarding fitness value (gbast) and the number of iteration in which gbest was fixed, apply the presented fuzzy system in section 2.3 and calculate PSO parameters. Then go to step 10.

Step 10: Survey termination condition:

1- If k > N_{pso}

2- Time of applying algorithm becomes more than predefined time.

If none of these condition indefeasible, put k = k+1 and go to step 5. Otherwise save the schedule, corresponding active schedule of P_g= (p_{g1},…,p_{gm})(the best yield position by all particles), as the best solution of HFAPSO algorithm in S^{*}.

3. COMPUTAIONAL RESULT

To test the effectiveness of our approach, some instances are needed. A method [13] is used to fuzzify some of the famous crisp benchmarks. For each crisp duration x, a three-point triangular fuzzy number is built. The first point is drawn randomly from the interval $[\delta_1 x, x]$, where δ_1 < 1. The center point is set equal to x, and the third point is drawn randomly from the interval $[x, \delta_2 x]$, where δ_2 >1. We set δ_1 = 0.85 and δ_2 = 1.25 in our benchmarks. The famous benchmarks of Ft and La are selected to fuzzify the processing time and to investigate the performance of the improved HFAPSO.

To measure the effectiveness and viability of HFAPSO, results are compared with GA and GPSO [14]. To perform the comparison experiments with same running time, the population size and the generation for HFAPSO are 40 and 2000. The experiments were conducted for 10 independent runs to evaluate the performance of HFAPSO on the FJSSP.

The programming was done in MATLAB and the programs were run on Pentium 3G. Table4 summarizes the experimental results. BFV is the best value of the run. WFV is the worst value of the run. AFV is the average of the best values of the run.

Table 4: Performance Comparison between Three Algorithms

problem		FT06	Ft10	La01	La03	La05
GPSO	BFV	56.08	981.3	684.4	608.4	606
	WFV	55.75	1044.3	694.9	627	606
	AFV	56.28	1024.8	685.7	615.8	606
GA	BFV	59.28	1069	698.1	637.3	606
	WFV	62.44	1134.1	736	667.9	615.3
	AFV	61.35	1094.6	713.8	651.3	608.9
HFAPSO	BFV	58.32	1023.6	687.3	618.3	606
	WFV	61.25	1126.4	705.7	642.5	609.2
	AFV	59.86	1053.1	694.2	627.5	607.1

4. CONCLUTIONS

In this paper for solving FJSSP, a new HFAPSO algorithm was presented. In the presented HFAPSO, for constructing initial solution of PSO, GRASP algorithm was applied and for tuning FAPSO parameters, fuzzy interference system was used.

Likewise for representation of FJSSP solutions, we used priority based list and also used Factoradic base for adopting the discrete solution of FJSSP with particles position in FAPSO. Additionally, for converting priority base of jobs on machine s, exact G&T algorithm was used.

The objective value ($C_{\max}$), was computed through predefined ranking method. Furthermore, a new method for approximating maximum of two fuzzy numbers was presented. At the end of this paper, benchmark with fuzzy processing time used for testing the presented algorithm and solutions of the algorithm compared with GA and GPSO. For achieving better solutions, results show that more study about used parameters (such as γ, w, c_1, c_2, NBF, NU, N_s, nIter and N_{pso}) in the proposed hybrid algorithm is necessary.

5. REFERENCES

[1] A. S. Jain, and S. Meeran, "Deterministic job-shop scheduling: Past, present and future", European Journal of Operational Research, 1999, 113(2). pp. 390–434.

[2] M. R. Garey, D. S, Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling", Mathematics of Operations Research, 1976, 1(2),pp. 117–129.

[3] B. J. Park, H. R. Choi, and H. S. A. Kim, "A hybrid genetic algorithm for the job shop scheduling problems", Computers and Industrial Engineering, 2003, 45(4), pp. 597–613.

[4] P.J.M.V. Laarhoven, E.H.L. Aarts, and J.K. Lenstra, "Job shop scheduling by simulated annealing", Operations Research, 1992, 40(1), pp. 113–125.

[5] Ge, H.W. Du, W. and Qian, F. 2007. A hybrid algorithm based on particle swarm optimization and simulated annealing for job shop scheduling. In Proceedings of the third international conference on natural computation, 3, pp. 715– 719.

[6] X.J. Wei, and Z.M. Wu, "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems", Computers and Industrial Engineering, 2005, 48(2), pp. 409–425.

[7] P. Fortemps, " Job shop scheduling with imprecise durations: a fuzzy approach", IEEE Transactions on Fuzzy Systems, 1997, 5(4), pp. 557–569.

[8] L. Deming, "Fuzzy job shop scheduling problem with availability constraints", Computers & Industrial Engineering, 2010, 58(4), pp. 610–617.

[9] A. Mahmodi Nejad, and M. Mashinchi,"Ranking fuzzy numbers based on the areas on the left and the right sides of fuzzy number", Computers and Mathematics with Applications, 2011, 61(2), pp. 431–442.

[10] J. Giffler, and G. L. Thompson, "Algorithms for solving production scheduling problems", Operations Research, 1960, 8(4), pp. 487–503.

[11] Kennedy, J. and Eberhart, R. 1995. Particle Swarm OptimizationIn: Proceedings of the 1995 IEEE international conference on neural networks. New Jersey: IEEE Press. pp. 1942–1948.

[12] Knuth, D. 1997. Semi numerical algorithms (3rd ed). The art of computer programming (Vol. 2). Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

[13] A.G. Omar, "A bi-criteria optimization: minimizing the integral value and spread of the fuzzy makespan of job shop scheduling problems", Applied Soft Computing, 2003, 2, pp. 197–210.

[14] Q. Niu, B. Jiao, and X. Gu, "Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time", Applied Mathematics and Computation, 2008, 205(1), pp. 148–158.