*International Conference on Modeling and Simulation*Coimbatore, 27-29 August 2007

OPTIMIZATION OF FLOWSHOP SCHEDULING WITH FUZZY DUE DATES USING A HYBRID EVOLUTIONARY ALGORITHM

M.S.N.Kiran Kumar[a], B.B.Biswal[b] , S.S.Mahapatra[c] and N.Jena[d]

International Conference on Modeling and Simulation
Coimbatore, 27-29 August 2007

OPTIMIZATION OF FLOWSHOP SCHEDULING WITH FUZZY DUE DATES USING A HYBRID EVOLUTIONARY ALGORITHM

M.S.N.Kiran Kumar[a], B.B.Biswal[b] , S.S.Mahapatra[c] and N.Jena[d]

[a] *Software engineer, I.B.M. Software Solutions, Bangalore*

[b] *Professor and Head, Dept of training and placement ,NIT, Rourkela, pin-769008,India;bbbiswal@nitrkl.ac.in*

[c] *professor, Dept of training and placement ,NIT, Rourkela, pin-769008,India;ssml@nitrkl.ac.in*

[d] *Dept of training and placement ,NIT, Rourkela,pin-769008; nilimajena@gmail.com*

Abstract: The concept of global work-place in the recent past has been the major driving factor to remain competitive by using the appropriate techniques in various aspects of manufacturing activities. Genetic Algorithm (GA) has proven to be a useful method of optimization for the combinatorial optimization problems. Particle Swarm Optimization (PSO), a new method of optimization, is able to accomplish the same goal as GA optimization in a faster way. The purpose of the present work is to investigate performance of the two algorithms when applied to flowshop scheduling with fuzzy due dates. An attempt has been made to hybridize the two algorithms in series. It is established that the hybrid algorithm (PSO-GA) produces better results on problems of larger size when compared to the individual performance of the algorithms.

Key words: Heuristics, flowshop scheduling, fuzzy due dates, Metaheuristics, Particle swarm optimization, Production scheduling.

1. INTRODUCTION

The flowshop scheduling problem determines the order of processing jobs over machines to optimize certain performance measures while all jobs have the same machine sequence. Since the flowshop problem was proposed, it has attracted much attention of researchers over decades. In recent decades, researchers have achieved substantial advances in scheduling techniques. Most scheduling problems are NP-hard problems. Researchers also notice that it is actually not very practical to spend too much effort on searching for an optimal solution in an industrial context because changes often occur to either the availability of resources or the set of jobs that have to be performed. Consequently, many heuristic algorithms have been developed to solve different scheduling problems. All these algorithms aim to reach near optimal solutions efficiently. As available from Panwalkar [1], heuristics for the flowshop scheduling problem have been proposed by Dannenbring [2], Campbell et al. [3], Framinan et al. [4], Allahverdi and Aldowaisan [5], Framinan and Leisten [6]. To achieve a better solution quality, modern meta-heuristics such as Simulated Annealing, Taboo Search, Genetic Algorithms, Ant Colony Optimization, and iterated local search have been presented for the flowshop scheduling problem with makespan minimization.

In the past, due dates and job processing times were usually treated as a crisp value. However, this assumption is not realistic in many cases. When studying scheduling problems in real world situations where some uncertain factors are incorporated into the problems, it is very difficult to get the definite value of due date and processing time. In such conditions it may be appropriate to consider fuzzy due date tolerating a certain amount of earliness or tardiness in the due date. Generally the fuzziness of due date greatly depends on the customer placing the order or the character of the product to be produced and it varies from order to order.

Efforts are made in this paper to implement PSO in flowshop scheduling considering fuzzy due dates. It is compared with genetic algorithm and finally it is proposed to use a PSO-GA series algorithm that performed better than either PSO or GA when applied to larger problems. The paper is organized as follows: Section 2 gives a formal definition of PFSP. In section 3, methodology of all the three algorithms is discussed, and computational results of test problems are shown in section 4. Finally section 5 summarizes the conclusions.

2. PROBLEM FORMULATION

2.1. Formulation of Flowshop Problem

If n jobs (j=1, 2… n) are to be sequenced through m machines (k=1, 2, 3… m) then the problem is to find the best permutation of jobs $\pi = \{\pi_1, \pi_2, \pi_3, ..., \pi_n\}$ to be valid for each machine. Given a job permutation $\pi = \{\pi_1, \pi_2, \pi_3, …., \pi_n\}$and P_{kj} indicating the processing times of job j on machine k, the calculation of completion time for n-job m-machine problem is given as follows:

$$C(\pi_j, k) = \max\{C(\pi_{j-1}, k), C(\pi_j, k-1) + P_{\pi j, k}\} \quad (1)$$

The performance measure of the sequence, expressed as makespan, is defined as:

$$C_{max}(\pi) = C(\pi_n.m) \quad (2)$$

Hence, the Flowshop problem is to find a permutation π_{best} such that

$$C_{max}(\pi_{best}) \leq C(\pi_n.m) \quad (3)$$

2.1.1 Flowshop Scheduling with Fuzzy due Dates

The concept of fuzzy due dates and the formulation of fuzzy scheduling problems were introduced by Ishii et al. [7]. In their fuzzy scheduling problems, the membership function of a fuzzy due date assigned to each job represents the grade of satisfaction of a decision maker for the completion time of that job. The following linear membership function was employed for representing a fuzzy due date of job j [7]:

$$\mu_j(C_j) = \begin{cases} 1 & \text{if } C_j \leq d_j^L, \\ 1-(C_j - d_j^L)/(d_j^U - d_j^L) & \text{if } d_j^L < C_j < d_j^U, \\ 0 & \text{if } d_j^U \leq C_j. \end{cases} \quad (4)$$

Where C_j is the completion time of j and $\mu(C_j)$ is the membership function of the fuzzy due date of that job. From the membership function d_j^L and d_j^U can be viewed as the tightest (i.e., earliest) due date and the loosest (i.e., latest) due date of job j, respectively. The full satisfaction ($\mu_j(C_j)=1$) is attained if $C_j \leq d_j^L$, and the grade of satisfaction is greater than 0 when $C_j \leq d_j^U$. In some particular situations full satisfaction is not attained if the completion time is too early. In such trapezoidal membership function is used to represent the grade of satisfaction of the decision maker.

2.1.2 Fuzzy due Dates

A fuzzy due date is associated with each job and is represented by a fuzzy set on R^+ (the positive part of real numbers). Since the fuzzy due date of each job represents the satisfaction grade of a decision maker, the shape of its membership function should be chosen according to the preference of the decision maker. While this paper deals only with linear membership function, there is no restriction on the shape of the membership function of each fuzzy due date.

2.2 Fuzzy Scheduling Problem

The total grade of satisfaction over *n* jobs is considered as the scheduling criterion. The maximization problem of the total grade of satisfaction can be written as:

$$Maximize \quad f_{sum}(x) = \sum_{j=1}^{n} \mu_{xj}(C_j(x)). \tag{5}$$

The problem is to find the sequence *x* that maximizes the total grade of satisfaction f_{sum}. Rewriting the problem with sequence *x* of the n jobs, we get:

$$Maximize \quad f_{sum}(x) = \sum_{j=1}^{n} \mu_{xj}(C_j(x)). \tag{6}$$

3 OVERVIEW OF PSO AND GA

3.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is one of the latest evolutionary optimization methods inspired by nature. PSO is based on the metaphor of social interaction and communication such as bird flocking and fish schooling. PSO is distinctly different from other evolutionary-type methods in a way that it does not use the filtering operation (such as crossover and/or mutation) and the members of the entire population are maintained through the search procedure so that information is socially shared among individuals to direct the search towards the best position in the search space. In a PSO algorithm, each member is called *particle*, and each particle moves around in the multidimensional search space with a velocity, which is constantly updated by the particle's own experience and the experience of the particle's neighbors or the experience of the whole swarm. Since PSO was first introduced to optimize various continuous nonlinear functions by Kennedy and Eberhart [8], it has been successfully applied to a wide range of applications such as mass-spring system, task assignment, supplier selection and ordering problem, and automated drilling. Although the applications of PSO on combinatorial optimization problems are still limited, PSO has its merit in the easy implementation and computational efficiency.

3.1.1 The Basic PSO Algorithm

PSO is an optimization strategy generally employed to find a global minimum. The basic PSO algorithm begins by scattering a number of particles in the function domain space. Each particle is essentially a data structure that keeps track of its current position *x* and its current velocity *v*. additionally, each particle remembers. The lbest (local best) position it has obtained in the past, denoted *p*. The best of these values among all particles (the global best remembered position) is denoted *g*. In PSO each single solution is a bird and it is called a particle. After finding the l-best values and g-best values the particle updates as shown in Eq 7 and 8 from Fatih Tasgetiren et al. [9].

$$v[] = w*v[] + c_1\, rand()*(pbest[] - present[]) + c_2* rand()*(gbest[] - present[]) \tag{7}$$

$$Present\ [] = present\ [] + v\ [] \tag{8}$$

Usually c_2, c_1=2, and rand ( ) is a random number between 0 and 1. The parameter *w* is to control the impact of the previous velocities on the current velocity. The velocities of particles on each dimension are clamped to v_{max}. If the sum of accelerations would cause the velocity on

that dimension to exceed v_{max}, which is a parameter specified by the user, then the velocity on that dimension is limited to v_{max}.

3.2 Genetic algorithm

GA begins its search from a randomly generated population of designs that evolve over successive generations (iterations). GA employs three operators to propagate its population from one generation to another to perform its optimization process. The first operator is the "Selection" operator that mimics the principal of "Survival of the Fittest". The second operator is the "Crossover" operator, which mimics mating in biological populations. The crossover operator propagates features of good surviving designs from the current population into the future population, which will have better fitness value on average. The last operator is "Mutation", which promotes diversity in population characteristics. The mutation operator allows for global search of the design space and prevents the algorithm from getting trapped in local minima.

In the genetic algorithm the population size is fixed at 20 and the mutation operator is fixed at 0.5 as in Hisao Ishibuchi [10] [11]. The stopping condition specified is 10000 evaluations of solutions.

4. PSO-GA SERIES ALGORITHM

In order to combine the advantages of both PSO and GA, a PSO-GA series algorithm is used to solve the problems mentioned earlier. Generally PSO converges faster than GA. Initially PSO explores a larger area and then gradually slows down and do finer search. So some times they may be trapped at local optimum and may not be able to find optimal solution. Hence if some other algorithm can be used at the point of slowing down of search of PSO, better results may be obtained in very much less time.

GA was used in series to PSO. PSO was init ially used to solve certain iterations and the schedules or solutions found by PSO by that time are taken as initial population to GA and the search is continued till the maximum iterations. PSO was continued up to 500 iterations and then GA is continued up to maximum iterations. As mentioned earlier max iterations are taken as 10,000.

5. SIMULATION RESULTS

PSO is applied to five different sized problems: 10x10 (jobs x machines), 20x10, 40x10, 50x10 and 100x10. For each size of problem six different problems were generated with fuzzy due dates and then they are tested and the average values are tabulated as shown. The due dates were generated according to the Hisao Ishibuchi et al. [10]. In this algorithm the population is fixed at 20 and the social, cognitive parameters are taken as $c_1=c_2=2$. Initial inertia weight is varied between w_0 = 0.95 to 0.99 and is never decreased below 0.40. Finally, the decrement factor • is taken as 0.975. In the genetic algorithm the population size is fixed at 20 and the mutation operator is fixed at 0.5. These parameters were obtained by preliminary computer simulations. The stopping condition was taken as 10000 evaluations. The values of Grade of satisfaction obtained in all the three algorithms are listed in Table 1.

Table 1: Comparison of all the algorithms for different problems

Sl. No	PSO 20x10	GA 20x10	PSO-GA 20x10	PSO 40x10	GA 40x10	PSO-GA 40x10	PSO 50x10	GA 50x10	PSO-GA 50x10	PSO 100x10	GA 100x10	PSO-GA 100x10
1	18.662	18.470	18.000	35.930	34.482	35.210	46.000	46.000	46.000	90.831	93.000	94.000
2	18.000	17.759	18.000	37.669	36.000	36.000	47.922	48.000	48.000	90.000	95.000	95.816

3	17.847	17.643	17.847	34.271	38.000	37.862	45.984	46.98	47.628	88.748	90.856	92.845
4	18.328	18.353	18.410	36.964	36.409	36.129	47.000	47.23	47.122	88.000	94.000	95.000
5	17.917	18.014	18.014	35.000	34.200	37.280	6.819	46.93	47.000	88.910	91.000	93.000
6	17.500	17.533	17.500	37.669	36.986	36.050	47.000	47.21	47.072	90.835	93.925	94.987
Avg	**18.042**	**17.962**	**17.962**	**36.136**	**36.022**	**36.422**	**46.787**	**47.06**	**47.137**	**89.554**	**93.130**	**94.275**

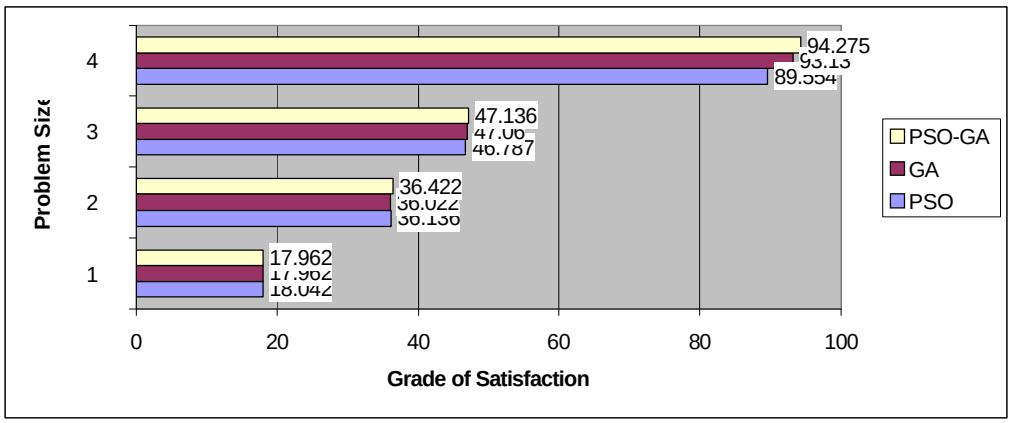

Fig 1: Comparison of three algorithms for different problems

The three algorithms PSO, GA and PSO-GA are compared with one another for the small and large sized problems. From the Fig 1, it can be observed that PSO alone performs better than GA for problem sizes 20x10 and 40x10. But PSO fails for 50x10 and 100x10 problems. PSO-GA algorithm dominates GA in all the cases where PSO could not do better. However, in the larger problems (50x10, 100x10), GA and PSO-GA performed better than PSO. For larger problem (100x10), PSO-GA is undoubtedly the better algorithm. In the case of remaining problems, PSO-GA performs better but the difference between grade of satisfaction of PSO-GA and GA is very less and so we now compare these methods with iterations taken for obtaining optimal solution.

Table 2 indicates that in most of the cases PSO-GA outperforms GA and finds the optimal solution in less number of iterations. The grade of satisfaction obtained is same in case of small sized problems and in the large problems, PSO-GA performed better than GA alone. So, it can be said that PSO-GA is an efficient technique that can be used for solving flowshop scheduling with fuzzy due-dates. A graph (Fig 2) is plotted with iterations on x-axis and grade of satisfaction on y-axis To compare the optimal solution (grade of satisfaction) found by the three algorithms with respect to iterations,. From the figure it is observed that PSO is weak in finding the optimal solution. PSO initially searches very fast and then converges between 3000 and 4000 iterations for a particular problem of size 100x10. GA performs better than PSO but finally PSO-GA is the best of all algorithms.

Table 2: Comparison of performance of GA and PSO-GA w.r.t to iterations required

S.no	PSO-GA 10x10 Iter	GA 20x10 Iter	PSO-GA 20x10 Iter	GA 40x10 Iter	PSO-GA 40x10 Iter	GA 50x10 Iter	PSO-GA 50x10 Iter	GA 100x10 Iter	PSO-GA 100x10 Iter
1	32	5959	117	9937	8467	6013	4461	8336	8500

2	37	3131	2292	3387	5339	8809	1672	9869	9727
3	48	7198	2280	9143	2329	9610	6636	8982	9293
4	523	7979	835	8503	9387	8404	6761	9949	9886
5	597	4169	2744	9024	5859	4583	2028	7452	9895
6	47	8019	417	5990	8370	9168	6035	9426	9667

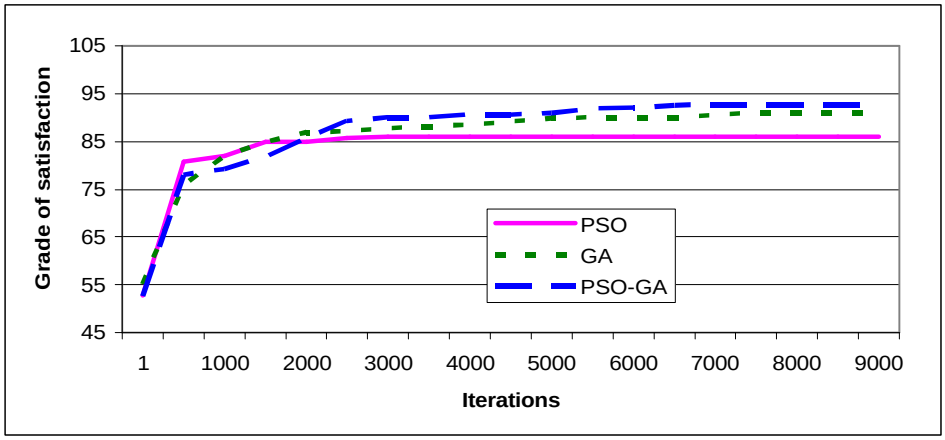

Figure 2: Variation of Grade of satisfaction with number of iterations

6.SUMMARY AND CONCLUSIONS

Observing the above results, it can be concluded that for the smaller problems any of the three methods can be used but as the number of iterations taken to find the solution is less in case of PSO, it should be used. Hence it can be said that for small sized problems PSO is a better option to use. But for the larger problems, PSO cannot find optimal solution because it gets trapped at the local optima and in this GA performs better than PSO. So, PSO is to be used in combination with some other algorithm to reach the optimal solution. PSO-GA series algorithm solves this problem of premature convergence and finds better optimal solutions. PSO-GA algorithm in most of the cases outperforms GA or obtains the same result as that of GA. The iterations in which the optimal solution is obtained is also lesser than that of GA. Hence PSO-GA can be used as an effective tool for solving larger flowshop scheduling problems with fuzzy due dates. It can be finally recommended to use PSO for smaller sized problems and to use PSO-GA series algorithm for medium and large sized problems.

PSO algorithm is effective and efficient technique used to solve the flowshop scheduling. The performance of the algorithm is good but it suffers from premature convergence. Therefore, in the case of complex or bigger problems, it cannot find better optimal solutions than GA. In order to overcome the limitations of PSO and that of GA, a hybrid algorithm (PSO-GA) is used. The results obtained by implementing this algorithm are promising and better than both the techniques in most of the cases. Thus, it can be concluded that PSO-GA is an efficient technique for solving flowshop scheduling with fuzzy due dates, especially when the problem size is large.

7. REFERENCES

1. Panwalkar S.S., Iskander W.,1977. ''A Survey of Scheduling Rules''. *Operations Research*, vol. **25**, no. 1
2. Dannenbring. D.G.,1997. ''An evaluation of flow shop sequencing heuristics''. *Management Science*, **23(11)**, 1174-1182.

3. Campbell, H.G., Dudek R.A ., Smith M.L.,1970. ‘’A heuristic algorithm for the n job, m machine sequencing problem’’. *Management Science*, **16(10)**, 630-637.
4. Framinan, J.M., Leisten. R., 2003, ‘’An efficient constructive heuristic for flow time minimization in permutation flow shops’’. *OMEGA*, **31**, 311-317.
5. Allahverdi, A., Aldowaisan., 2002. ‘’New heuristics to minimize total completion time in *m*-machine flow shops’’. *International Journal of Production Economics*, **77**, 71-83.
6. Framinan, J.M., R. Leisten & R. Ruiz-Usano. (2005). Comparison of heuristics for flow time minimization in permutation flow shops. Computers and Operations Research v 32, n 5, 1237-1254.
7. Ishii, H., M. Tada & T. Masuda. (1992). Two scheduling problems with fuzzy due dates. Fuzzy Sets and Systems, vol. 46, 339-347.
8. James Kennedy & Russel Eberhart. (1995). Particle Swarm Optimization. IEEE Transactions, 1942-1948.
9. Fatih Tasgetiren, M. (2004). Particle Swarm Optimization Algorithm for Makespan and Total Flow-time Minimization in Permutation Flowshop Sequencing Problem. Proceedings (Lecture Notes in Computer Science), Vol.3172, 382-389.
10. Hisao Ishibuchi., Tadahiko Murata & Hideo Tanaka. (1994). Genetic Algorithms and neighborhood algorithms for fuzzy flowshop scheduling problems. Fuzzy sets and systems, vol-67, p.81-100.
11. Hisao Ishibuchi., Tadahiko Murata & Kyu-Hung Lee. (1996). Relations between Conventional Scheduling Problems and Fuzzy Scheduling Problems. Proceedings of 35th conference on Decision and control, Kobe, Japan, Dec 1996.