Available online at www.sciencedirect.com

ScienceDirect

Progress in Natural Science 18 (2008) 1179–1183

Progress in Natural Science

www.elsevier.com/locate/pnsc

Short communication

# Hybrid particle swarm optimization for solving resource-constrained FMS

Dongyun Wang [*], Liping Liu

*Department of Electronic Engineering, Zhongyuan University of Technology, Zhongyuan Road 41#, Zhengzhou 450007, PR China*

Received 4 March 2008; received in revised form 18 March 2008; accepted 20 March 2008

**Abstract**

In this paper, an approach for resource-constrained flexible manufacturing system (FMS) scheduling was proposed, which is based on the particle swarm optimization (PSO) algorithm and simulated annealing (SA) algorithm. First, the formulation for resource-constrained FMS scheduling problem was introduced and cost function for this problem was obtained. Then, a hybrid algorithm of PSO and SA was employed to obtain optimal solution. The simulated results show that the approach can dislodge a state from a local minimum and guide it to the global minimum.

*Keywords:* Particle swarm optimization; Simulated annealing algorithm; FMS; Scheduling

## 1. Introduction

Resource-constrained flexible manufacturing system (FMS) scheduling problem is a typical non-polynomial (NP) problem. Historically, resource-constrained FMS scheduling problem was treated via classical math optimization or heuristic methods. The conventional methods [1,2], such as dynamic planning and expert system, for solving this NP problem cannot obtain reasonable solution. The math optimization algorithm is mainly based on the branch and bound method [3]. Because it is time-consuming and can only solve small problems, such algorithm has no attraction to practitioners. On the other hand, heuristic algorithms, which are quite good alternatives, have been developed greatly during the past decade, such as the simulated annealing [4], taboo search [5], and genetic algorithm [6]. Zhou [7] and Lo [8] have proposed a methods for Job-shop problem (JSP) with neural networks to minimize makespan. However, only local optimum can be obtained because of the weakness of learning algorithm. Recently, Wang and Zhang proposed a hybrid optimization strategy (GA-SA) [9] and Binato et al. proposed a greedy randomized adaptive search procedure for solving resource-constrained JSP scheduling problem [10].

## 2. Formulation of scheduling for FMS

### 2.1. Formulation

Scheduling for FMS is a reasonable allocation of constrained resource of FMS while minimizing a certain criterion. The resources are called machine, tool, fixture, etc. Therefore, scheduling problem in fact is constrained optimization problem. The restrains among operation of each job and precedence order of jobs completion time, which the order is provided by planning system of flexible assemble system (FAS), must be subject to optimization assembly performance.

The manufacturing system, which is to schedule, is assumed as follows:

[*] Corresponding author. Tel./fax: +86 371 67698792.
*E-mail address:* wdy1964@yahoo.com.cn (D. Wang).

1002-0071/$ - see front matter © 2008 National Natural Science Foundation of China and Chinese Academy of Sciences. Published by Elsevier Limited and Science in China Press. All rights reserved.
doi:10.1016/j.pnsc.2008.03.021

① Jobs $i$ ($i = 1,2,\ldots,n$) and operations $j$ ($j = 1,2,\ldots,$ n$_i$).
② $S$ ($s = 1,2,\ldots,m$) kinds of resource, the number of each resource is $r_s$.
③ The completion time of operations of jobs is subject to the precedence constraints.
④ The completion time of jobs is subject to the precedence constraints.
⑤ Minimize the total completion time.

The constraints are given by the following inequalities:

$$x_{ij} - x_{il} + t_{il} \leqslant 0$$
$$\text{for all } [j,l] \in R_i, \quad i = 1,2,\cdots,n,$$
$$j = 1,2,\cdots,n_i \tag{1}$$
$$x_{ij} - x_{il} + t_{il} \leqslant 0 \text{ or } x_{il} - x_{ij} + t_{ij} \leqslant 0;$$
$$\text{for all } [j,l] \in Q_i, \quad i = 1,2,\cdots,n,$$
$$j = 1,2,\cdots,n_i \tag{2}$$
$$x_{ij} - x_{il} + t_{il} \leqslant 0 \text{ or } x_{il} - x_{ij} + t_{ij} \leqslant 0;$$
$$\text{for all } [j,l] \in N_{sq}, \ s = 1,2,\cdots,n,$$
$$q = 1,2,\cdots,r_s \tag{3}$$
$$x_k - x_i \leqslant 0; \ \text{for all } [k,i] \in P \tag{4}$$
$$t_{ij} - x_{ij} \leqslant 0; \quad i = 1,2,\cdots,n,$$
$$j = 1,2,\cdots,n_i \tag{5}$$

where $n$ is the number of jobs, $n_i$ is the number of operation for the $i$th job, $m$ is the number of resource, $r_s$ is the number of each resource, $R_i$ is the set of operation couple $[j,1]$ of job $i(i = 1,2,\ldots,n)$ with operation $j$ prior to operation $l$, $Q_i$ is the set of operation couple $[j,1]$ of job $i(i = 1,2,\ldots,n)$ with no precedence between operation $j$ and $l$, $I_i$ is the set of operation which can be assigned first, $i = 1,2,\ldots,n$, $t_{il}$ is the processing time of $l$ operation of the $i$th job, $l = 1,2,\ldots,n_j$, $i = 1,2,\ldots,n$, $x_{ij}$ is the completion time of $j$ operation of the $i$th job, $l = 1,2,\ldots,n_i$, $i = 1,2,\ldots,n$; $x_i$ is the completion time of operation which is the last one in scheduling sequences for job $I$; $N_{sq}$ is the set of operation occupying the $q$th resource of resource $s$; and $P$ is the set of job couple $[k,i]$ with job $i$ prior to job $k$.

The constraint (1) ensures that the precedent constraint relation of each job's operation is to be satisfied. The constraint (2) is to avoid the overlap (in time) between two operations of each job. The constraint (3) ensures that every single resource only can be used by one job. The constraint (4) is to satisfy the precedent relation of the jobs. The constraint (5) is to ensure that the starting time should be positive. Therefore, the mathematical formulation for solving resource-constrained FMS scheduling is described as follows:

$$\text{Minimize } \sum_{i=1}^{n} x_i$$

subject to constraints from (1) to (5).

## 3. Particle swarm optimization algorithm

Particle swarm optimization (PSO) is a parallel population-based computation technique proposed by Kennedy and Ebehart [11,12], which was motivated by the organisms behavior such as schooling of fish and flocking of birds. PSO can solve a variety of difficult optimization problems [13]. PSO's major difference from genetic algorithm (GA) is that PSO uses the physical movements of the individuals in the swarm and has a flexible and well-balanced mechanism to enhance and adapt to the global and local exploration abilities, whereas GA uses genetic operators. Another advantage of PSO is its simplicity in coding and its consistency in performance.

### 3.1. Standard particle swarm optimization

PSO is similar to the evolutionary algorithm, and its system is initialized with population (named swarm in PSO) of random solutions. Each individual or potential solution, named particle, flies in the multi-dimensional problem space with a velocity dynamically adjusted according to the flying experiences of its own and its colleagues. The global optimizing model proposed by Shi and Eberhart [14] is as follows:

$$V_{id} = W \times V_{id} + C_1 \times \text{Rand} \times (P_{\text{best}} - X_{id})$$
$$+ C_2 \times \text{Rand} \times (G_{\text{best}} - X_{id}) \tag{6}$$
$$X_{id} = X_{id} + V_{id} \tag{7}$$

where $V_{id}$ is the velocity of particle $i$, $X_{id}$ is the particle position, $W$ is the inertial weight, $C_1$ and $C_2$ are the positive constant parameters, Rand is the random functions in the range $[0,1]$, $P_{\text{best}}$ is the best position of the $i_{\text{th}}$ particle, and $G_{\text{best}}$ is the best position among all particles in the swarm.

For Eq. (6), the first part represents the inertia of the previous velocity. The second part is the cognition part, which represents the private thinking of itself. The third part is the social part, which represents the cooperation among particles. The process for implementing the PSO algorithm is as follows:

① Initialize a swarm of particles with random positions and velocities in the multi-dimensional problem space.
② For each particle, evaluate the desired optimization fitness function.
③ Compare particle's fitness value with particle's Pbest. If the current value is better than Pbest, then set Pbest value equal to the current value, and the Pbest position equal to the current position in the multi-dimensional space.
④ Compare the fitness evaluation value with the best swarm's fitness value obtained so far. If the current value is better than Gbest, then reset Gbest to the current particle's value.
⑤ Change the velocity and position of the particle according to Eqs. (6) and (7), respectively.
⑥ Loop to step (2) until a stop criterion is satisfied, usually a sufficiently good fitness or a specified number of generations.

### 3.2. Setting parameters

In Eq. (6), inertial weight ($W$) is an important parameter to search ability of PSO algorithm. A large inertia weight facilitates research in a new area while a small inertia weight facilitates fine-searching in the current search area. Suitable selection of the inertia weight provides a balance between global exploitation, and results in less iteration on average to find a sufficiently good solution. Therefore, linearly decreasing the inertia weight from a relative large value to a relatively small value through the course of PSO run, PSO tends to have more local search ability near the end of run. In this study the inertia weight is set to be the following equation:

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{I_{\max}} \times I \tag{8}$$

where $w_{\max}$ is the initial value of weighting coefficient, $w_{\min}$ is the final value of weight coefficient, $I_{\max}$ is the maximum number of iterations or generation, and $I$ is the current inertia or generation number.

### 3.3. Fitness function

Fitness is used as performance evaluation of particles in the swarm. Fitness is usually represented with a function $f$: $s \rightarrow R^+$ ($s$ is the set of candidate schedules and $R^+$ is the set of positive real values). Mapping an original objective function value to a fitness value that represents relative superiority of particles is a

$$\begin{aligned} E = & \sum_{i=1}^{n} x_{ij} + \sum_{i=1}^{n} \sum_{j} \sum_{l} H_1 * \underset{j,l \in R_i}{F_1 (x_{ij} - x_{jl} + t_{jl})} \\ & + \sum_{i=1}^{n} \sum_{j} \sum_{l} H_2 * (\min \underset{j,l \in Q_i}{(F_1(x_{ij} - x_{jl} + t_{jl}), F_1(x_{jl} - x_{ij} + t_{ij}))}) \\ & + \sum_{s=1}^{m} \sum_{q=1}^{r_s} \sum_{j} \sum_{l} H_3 * (\min (F_1 (x_{ij} - \underset{j,l \in N_{sqi}}{x_{jl} + t_{jl}}), F_1 (x_{jl} - x_{ij} + t_{ij}))) \\ & + \sum_{k} \sum_{i} H_4 * F_1(\underset{k.i \in P}{x_k} - x_i) + \sum_{i=1}^{n} \sum_{j=1i}^{n_i} H_5 * F_1 (x_{ij} - t_{ij}) \end{aligned}$$

feature of evaluation function. In resource-constrained FMS scheduling problem, the objective function is to minimize the cost function. We can propose the following cost function for resource-constrained FMS scheduling problem:

where $H_1$, $H_2$, $H_3$, $H_4$, $H_5$ are positive constants which are usually largely depending on the problem to be solved. Actually, it is a penalty function of resource-constrained FMS scheduling problem

$$F(x) = \begin{cases} e^x; \ x > 0 \\ 0; \ x \leqslant 0 \end{cases}$$

## 4. Simulated annealing algorithm

Simulated annealing (SA) algorithm is a meta-strategy local search method that attempts to avoid producing the poor local maximum inherent in the steepest ascent method. It employs additional random acceptance strategy that allows occasional downhill moves to be accepted with certain probabilities [15]. SA algorithm has produced good results for many scheduling problems [16].

In SA algorithm, the improvements are obtained by choosing another solution $s'$ that belongs to the neighborhood $N(s_0)$ of the current solution $s_0$. When the current solution changes from $s_0$ to $s'$, the objective function will also change, namely, $\Delta = E(s') - E(s_0)$. For the minimization problem, if $\Delta < 0$, the new solution $s'$ will be accepted. If $\Delta > = 0$, the new solution will be accepted with the probability $\exp(-\Delta/t)$, where $t$ is the temperature. Generally, the algorithm starts from a high temperature, and then the temperature gradually decreases. At each temperature, the search will be performed for a certain number of iterations, which is called the temperature length. When the termination condition is satisfied, the algorithm will stop.

### 4.1. Control parameters selection

SA algorithm generally must be carefully designed because the choice of its parameters might affect the quality of the solution and computation. In general, a slow search will lead to better solutions. However, the slow search tends to consume more computation time. Therefore, it is necessary to take a tradeoff between them.

Control parameters were set according to the problem characteristics. Through many experiments, we found that the solutions and running time are both better when the initial temperature is set according to the maximal difference in fitness value between any two neighboring solutions. The length of temperature denotes the number of moves made at the same temperature, and generally, it is set according to the size of neighborhood solutions for a given solution. In SA optimization process, the temperature is gradually lowered. It is well-known that the method that specifies temperature with the equation $t_k = \lambda t_{k-1}$ is often a good choice and can provide a tradeoff between computational time and good solutions. The smaller the cooling rate $\lambda$ is, the quicker the temperature descends. To terminate the algorithm, we select the termination temperature $t_{\text{end}}$, when the current temperature $t < t_{\text{end}}$ the algorithm will stop. Generally, $t_{\text{end}}$ is a small value.

## 5. Hybrid PSO algorithm

PSO algorithm is problem-independent, which means that little specific knowledge relevant to given problem is required. What we have known is just the fitness evaluation for each solution. This advantage makes PSO more robust than many other search algorithms. However, PSO, as stochastic search algorithm, is prone to lack global search ability at the end of a run. PSO may fail to required optimum in case when the problem to be solved is too complicated and complex. SA employs certain probability to avoid becoming trapped in a local optimum and search process can be controlled by the cooling schedule. By designing the neighborhood structure and cooling schedule of SA, we can control the search process and avoid individuals being trapped in local optimum more efficiently. Thus, a hybrid algorithm of PSO and SA is presented as follows:

```
Begin
Step 1. Initialization
    (1.1) Initialize swarm size, each particle's position and velocity.
    (1.2) Evaluate each particle's fitness function.
    (1.3) Initialize $G_{best}$ position with the lower fitness particle in swarm.
    (1.4) Initialize $P_{best}$ position with copy of particle itself.
    (1.5) Initialize $w_{max}$, $w_{min}$, $I_{max}$, $C_1$, $C_2$, maximal generation, and generation = 0.
    (1.6) Determine $T_0$, $T_{end}$, and $\lambda$.

Step 2. Computation
    (2.1) PSO
    While (the maximum of generation is not satisfied)
    Do{generation++;
        Generate next swarm by Eqs. (6) and (7)
        Evaluate swarm {Find new $G_{best}$, $P_{best}$; Update $G_{best}$ of swarm and $P_{best}$ of particle;}
        }
    (2.2) SA
    For $G_{best}$, particle $s$ of swarm

{$T_k = T_0$;
    While ($T_k > T_{end}$)
    Do {Generate a neighbor solution $s'$ from $s_0$;
        Compute fitness of $s'$
        Evaluate $s'$ {$\Delta = E(s') - E(s_0)$;
          If (min[1, exp($-\Delta/T_k$) > random[0, 1]])
        {Accept $s'$}
          Update the best solution found do far if possible;}
    $t_k = \lambda t_{k-1}$;
          }
}
Step 3. Output optimization results.
End
```

It can be seen that PSO provides initial solution for SA during the hybrid process. Such hybrid algorithm can be converted to general PSO by omitting SA unit, and it can be converted to traditional SA by setting swarm size to one particle. PSO implements easily and reserves the generality of PSO and SA. Moreover, such hybrid algorithm can be applied to many combinatorial optimization problems by simple modification.

## 6. Simulation result

The initial coefficients are as follows:

The parameter for SA: $T_0 = 10{,}000$; $\lambda = 0.98$; $H_1 = H_2 = H_3 = H_4 = 100$, $H_5 = 1$.

For PSO algorithm, $w_{max} = 0.9$, $w_{min} = 0.4$, $c_1 = 0.3$, $c_2 = 0.5$, swarm size is set to be 30 and the maximum of iterative generations $I_{max}$ is set to be 300.

Simulations were successfully done on 2 jobs/3 machines scheduling problem given in Ref. [7]. Fig. 1 shows the data of 2 jobs/3 machines scheduling problem. Fig. 2 shows the optimum scheduling, the order of completion time is $x_2 = 10$, $x_1 = 15$, $x_3 = 19$, the maximum machine occupy time is 19, and the total completion time is 44. It is better than the results of Ref. [7]. The same simulations were also done on the $10 \times 10$ problem in Ref. [7]. The results, with the maximum machine occupation time of 95 and the total completion time of 867, is better than the results of Ref. [7], with maximum machine occupation time of 98 and the total completion time of 889.

## 7. Conclusions

We have formulated a resource-constrained FMS scheduling problem. A novel hybrid algorithm based on SA and PSO has been proposed, which the optimum can be obtained. The approach is flexible. It incorporates different machines, different job lengths. The performance of hybrid PSO algorithm is evaluated in comparison with the other optimization methods. The result shows that the new algorithm is more effective and efficient.

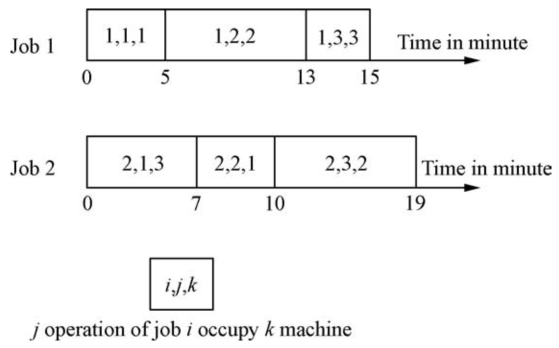

Fig. 1. Data of 2 jobs/3 machines scheduling problem.

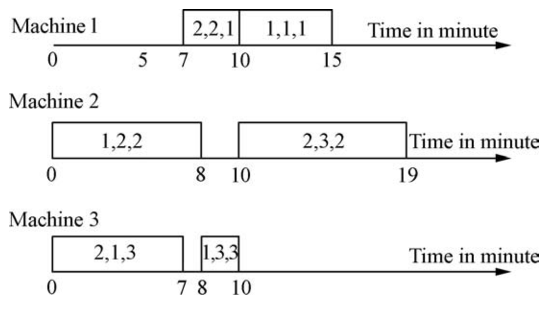

Fig. 2. GANTTE of scheduling problem shown in Fig. 1.

## Acknowledgements

This work was supported by Natural Science Foundation of Henan Province (Grant No. 072300410400) and China Scholarship Council Foundation (Grant No. 2006331).

## References

[1] Rodammer FA, Whit KP. A recent survey of production scheduling. IEEE Trans Syst Man Cybern 1998;18(8):841–51.

[2] French S. Sequencing and scheduling: an introduction to the mathematics of job-shop. New York: Wiley; 1988.

[3] Lageweg BJ, Lenstra JK, Rinnooy AG. Job-shop scheduling by implicit enumeration. Manage Sci 1977;24(4):441–50.

[4] Laarhoven PJ, Arts EH, Lensra JK. Job shop scheduling by simulated annealing. Oper Res 1992;40(1):113–25.

[5] Croce FD, Tadei R, Volta G. A genetic algorithm for the job shop scheduling problem. Comput Oper Res 1995;22(1):15–24.

[6] Dell'Amico M, Trubian M. Applying tabu search to the job shop scheduling problem. Ann Oper Res 1993;40:231–52.

[7] Zhou DN. Scaling neural network for job-shop scheduling. In: Proc. IJNN90, vol. 3; 1990. p. 889–92.

[8] Lo ZP. Scheduling neural network for flexible manufacture systems. In: Proc. IEEE Int. Conf. Robotics & Automation, Sacramento; 1991. p. 818–23.

[9] Wang L, Zhang D. An effective hybrid optimization strategy for job shop scheduling problem. Comput Oper Res 2001;28(6):585–96.

[10] Binato S, Hery WJ, Looewenstrn DM, et al. AGRASP for Job shop scheduling. In: Essays and surveys in meta-heuristics. Boston: Kluwer Academic Publishers; 2001. p. 59–79.

[11] Kennedy J, Ebehart R. Particle swarm optimization. In: Proceeding of the 1995 IEEE international conference on neural network; 1995. p. 1942–8.

[12] Ebehart R, Kennedy J. A new optimizer using particle swarm theory. In: Proceeding of the sixth international symposium on micro machine and human science; 1995. p. 39–43.

[13] Salman A, Ahmad I, Madani SA. Particle swarm optimization for task assignment problem. Microprocess Microsyst 2002;26(8): 363–71.

[14] Shi Y, Ebehart R. Empirical study ofparticle swarm optimization. In: Proceeding congress on evolutionary computation; 1999. p. 1945–50.

[15] Hasan M, Osman IH. Local search algorithm for the maximal planar layout problem. Int Trans Oper Res 1995;2(1):89–106.

[16] Xia WJ, Wu ZM. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling. Comput Ind Eng 2005; 48(2):409–25.