Available online at www.sciencedirect.com
ScienceDirect
Computers & Industrial Engineering 51 (2006) 791–808
computers & industrial engineering
www.elsevier.com/locate/dsw

A hybrid particle swarm optimization for job shop scheduling problem

D.Y. Sha [a,b,*], Cheng-Yu Hsu [b]

[a] Department of Business Administration, Asia University, 500 Liufeng Road, Wufong, Taichung 413, Taiwan, ROC
[b] Department of Industrial Engineering and Management, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan, ROC

Received 14 January 2006; received in revised form 14 September 2006; accepted 14 September 2006
Available online 27 October 2006

Abstract

A hybrid particle swarm optimization (PSO) for the job shop problem (JSP) is proposed in this paper. In previous research, PSO particles search solutions in a continuous solution space. Since the solution space of the JSP is discrete, we modified the particle position representation, particle movement, and particle velocity to better suit PSO for the JSP. We modified the particle position based on preference list-based representation, particle movement based on swap operator, and particle velocity based on the tabu list concept in our algorithm. Giffler and Thompson's heuristic is used to decode a particle position into a schedule. Furthermore, we applied tabu search to improve the solution quality. The computational results show that the modified PSO performs better than the original design, and that the hybrid PSO is better than other traditional metaheuristics.

Keywords: Job shop problem; Scheduling; Particle swarm optimization

1. Introduction

The job shop scheduling problem (JSP) is one of the most difficult combinatorial optimization problems. The JSP can be briefly stated as follows (French, 1982; Gen & Cheng, 1997). There are n jobs to be processed through m machines. We shall suppose that each job must pass through each machine once and once only. Each job should be processed through the machines in a particular order, and there are no precedence constraints among different job operations. Each machine can process only one job at a time, and it cannot be interrupted. Furthermore, the processing time is fixed and known. The problem is to find a schedule to minimize the makespan ($C_{\max}$), that is, the time required to complete all jobs.

Garey, Johnson, and Sethi (1976) demonstrated that JSP is NP-hard, so it cannot be exactly solved in a reasonable computation time. Many approximate methods have been developed in the last decade to solve

* Corresponding author. Tel.: +886 4 23323456x1936; fax: +886 4 2331 6699.
E-mail addresses: yjsha@mail.nctu.edu.tw (D.Y. Sha), cyhsu.iem92g@nctu.edu.tw (C.-Y. Hsu).

0360-8352/$ - see front matter
doi:10.1016/j.cie.2006.09.002

JSP, such as *simulated annealing* (SA) (Lourenço, 1995), *tabu search* (TS) (Nowicki & Smutnicki, 1996; Pezzella & Merelli, 2000; Sun, Batta, & Lin, 1995), and *genetic algorithm* (GA) (Bean, 1994; Gonçalves, Mendes, & Resende, 2005; Kobayashi, Ono, & Yamamura, 1995; Wang & Zheng, 2001). We applied a new evolutionary search technique – particle swarm optimization (PSO) – to solve the JSP in this paper.

The optimal JSP solution should be an active schedule. In an active schedule the processing sequence is such that no operation can be started any earlier without delaying some other operation (French, 1982). To reduce the search solution space, the tabu search proposed by Sun et al. (1995) searches solutions within the set of active schedules. In our algorithm, we applied Giffler and Thompson's (1960) heuristic to decode a particle position into a schedule. Furthermore, we applied a tabu search to improve the solution quality.

The background of particle swarm optimization (PSO) is introduced in the next section. In Section 3, we propose a hybrid PSO for the JSP. In Section 4, we test the hybrid PSO on Fisher and Thompson (1963) and Lawrence (1984) and Taillard (1993) test problems. Finally, conclusions and remarks for further works are given in Section 5.

2. The background of particle swarm optimization

Particle swarm optimization (PSO) was developed by Kennedy and Eberhart (1995). PSO is a population-based optimization algorithm. Each particle is an individual and the swarm is composed of particles. The problem solution space is formulated as a search space. Each position in the search space is a correlated solution of the problem. Particles cooperate to find out the best position (best solution) in the search space (solution space).

Particles move toward the pbest position and gbest position with each iteration. The pbest position is the best position found by each particle so far. Each particle has its own pbest position. The gbest position is the best position found by the swarm so far. The particle moves itself according to its velocity. The velocities are randomly generated toward pbest and gbest positions. For each particle k and dimension j, the velocity and position of particles can be updated by the following equations:

$$v_{kj} \leftarrow w \times v_{kj} + c_1 \times rand_1 \times (pbest_{kj} - x_{kj}) + c_2 \times rand_2 \times (gbest_j - x_{kj}) \quad (1)$$

$$x_{kj} \leftarrow x_{kj} + v_{kj} \quad (2)$$

In Eqs. (1) and (2), v_{kj} is the velocity of particle k on dimension j, and x_{kj} is the position of particle k on dimension j. The $pbest_{kj}$ is the pbest position of particle k on dimension j, and $gbest_j$ is the gbest position of the swarm on dimension j. The inertia weight w was first proposed by Shi and Eberhart (1998a, 1998b), and is used to control exploration and exploitation. The particles maintain high velocities with a larger w, and low velocities with a smaller w. A larger w can prevent particles from becoming trapped in local optima, and a smaller w encourages particles exploiting the same search space area. The constants c_1 and c_2 are used to decide whether particles prefer moving toward a pbest position or gbest position. The $rand_1$ and $rand_2$ are random variables between 0 and 1. The process for PSO is as follows:

Step 1: Initialize a population of particles with random positions and velocities on d dimensions in the search space.
Step 2: Update the velocity of each particle, according to Eq. (1).
Step 3: Update the position of each particle, according to Eq. (2).
Step 4: Map the position of each particle into solution space and evaluate its fitness value according to the desired optimization fitness function. At the same time, update pbest and gbest position if necessary.
Step 5: Loop to step 2 until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations.

The original PSO design is suited to a continuous solution space. For better suiting to combinatorial optimization problems, we have to modify PSO position representation, particle velocity, and particle movement. Zhang, Li, Li, and Huang (2005) proposed a PSO for resource-constrained project scheduling, and compared two kinds of position representation: (1) priority-based representation (particle position represented by prior-