Expert Systems with Applications 36 (2009) 7027–7032

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

An efficient flow-shop scheduling algorithm based on a hybrid particle swarm optimization model

I-Hong Kuo [a], Shi-Jinn Horng [a,b,c,f,*], Tzong-Wann Kao [d], Tsung-Lieh Lin [a], Cheng-Ling Lee [e], Takao Terano [c], Yi Pan [f]

[a] Department of Electrical Engineering, National Taiwan University of Science and Technology, 106 Taipei, Taiwan
[b] Department of Electronic Engineering, National United University, 36003 Miao-Li, Taiwan
[c] Department of Computational Intelligence and Systems Science, Tokyo Institute of Technology, Japan
[d] Department of Electronic Engineering, Technology and Science Institute of Northern Taiwan, Taipei, Taiwan
[e] Department of Electro-Optical Engineering, National United University, 36003 Miao-Li, Taiwan
[f] Department of Computer Science, Georgia State University, Atlanta, GA, United States

ARTICLE INFO

Keywords:
Flow-shop scheduling problem
Particle swarm optimization
Random-key encoding scheme
Individual enhancement scheme

ABSTRACT

In this paper, a new hybrid particle swarm optimization model named HPSO that combines random-key (RK) encoding scheme, individual enhancement (IE) scheme, and particle swarm optimization (PSO) is presented and used to solve the flow-shop scheduling problem (FSSP). The objective of FSSP is to find an appropriate sequence of jobs in order to minimize *makespan*. *Makespan* means the maximum completion time of a sequence of jobs running on the same machines in flow-shops. By the RK encoding scheme, we can exploit the global search ability of PSO thoroughly. By the IE scheme, we can enhance the local search ability of particles. The experimental results show that the solution quality of FSSP based on the proposed HPSO is far better than those based on GA [Lian, Z., Gu, X., & Jiao, B. (2008). A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan. *Chaos, Solitons and Fractals, 35,* 851–861.] and NPSO [Lian, Z., Gu, X., & Jiao, B. (2008). A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan. *Chaos, Solitons and Fractals*, *35*, 851–861.], respectively.

1. Introduction

Flow-shop scheduling problem (FSSP) is a combinatorial optimization problem and it is an NP-complete problem (Garey, Johnson, & Sethi, 1976). The objective of FSSP is to find a permutation schedule that minimizes the maximum completion time of a sequence of k jobs in an m-machine flow-shop. Every job has m operations, and every machine has k jobs. Every job executes its operations on every machine in the order of (M_1, M_2, ..., M_m). Every operation of a job cannot be preemption. Every machine can only execute an operation of a job at the same time. FSSP has interested many researchers during the past decades, and many heuristic methods have been developed, such as Johnson's rule (Johnson, 1954), Palmer's slope index (Palmer, 1965), CDS (Campbell, Dudek, & Smith, 1970), and NEH (Nawaz, Enscore, & Ham, 1983). The heuristic methods get the solutions by the pre-defined rules, but the rules are not suitable for every environment. So some researchers made use of evolution algorithms to get the solutions, such as Taillard's tabu search method (Taillard, 1990), Ogbu and Smith's simulated annealing algorithm (Ogbu & Smith, 1990), Reeves's genetic algorithm (Reeves, 1995), and Lian et al.'s particle swarm optimization algorithm (Lian, Gu, & Jiao, 2008). In this paper, we focus on exploiting particle swarm optimization algorithm to get the solutions for FSSP.

Particle swarm optimization (PSO) is a novel computational evolution model that was developed by Kennedy and Eberhart (1995). They simulated birds' swarm behavior in PSO, and made every particle in the swarm move according to its experience and the best particle's experience. After the evolution ends, the best particle in the swarm is the best solution for the input problem.

In this paper, we propose a new hybrid swarm optimization algorithm named HPSO in order to solve FSSP. We use the continuous version of PSO to be the main component of HPSO because of its excellent searching ability in the continuous space. PSO's global searching ability is better in comparison with the local searching ability; for the sake of improving the PSO's local searching ability, an individual enhancement scheme is also proposed as the other component of HPSO. In addition to the schemes described above, we consider the influence of time-varying coefficients of PSO. The

* Corresponding author. Address: Department of Electronic Engineering, National United University, No. 1, Lien-Da, Kung-Ching Li, 36003 Miao-Li, Taiwan. Tel.: +886 2 27376700; fax: +886 2 27301081.
E-mail address: horngsj@yahoo.com.tw (S.-J. Horng).

0957-4174/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2008.08.054

experimental results show that HPSO outperforms GA (Lian et al., 2008), and NPSO (Lian et al., 2008) in solution quality for FSSP, respectively.

The paper is organized as follows: A brief overview of FSSP and PSO are described in Sections 2 and 3, respectively. Section 4 discusses the details of the HPSO algorithm. Section 5 discusses the experimental results obtained by the HPSO algorithm. Finally, Section 6 summarizes the contribution of this paper and conclusions.

2. Flow-shop scheduling problem

The objective of the flow-shop scheduling problem (FSSP) is to find an appropriate permutation schedule for jobs that minimizes the maximum completion time. Every job has m operations. The flow-shop has m machines. Every operation of any job must follow the same machine sequence $M_1, M_2, \ldots, M_m$. That is, the rth operation of job i is performed by machine M_r with fixed processing time $T(r, i)$, $1 \leqslant r \leqslant m$, and $1 \leqslant i \leqslant k$. Every machine can execute only an operation at the same time. The operations of a job cannot be preemption. All machines execute jobs in the order of the pre-defined permutation schedule. The equation of computing the maximum completion time of the permutation schedule is given as follows:

$$\begin{aligned} C(1,\ 1) &= T(1,\ 1) \\ C(1,\ i) &= C(1,\ i-1) + T(1,\ i) \\ C(r,\ 1) &= C(r-1,\ 1) + T(r,\ 1) \\ C(r,\ i) &= \max(C(r,\ i-1), C(r-1,\ i)) + T(r,\ i) \end{aligned} \tag{1}$$

In Eq. (1), $T(r, i)$ means the execution time of the rth operation of the ith job on machine M_r, $C(r, i)$ means the maximum completion time of the ith job on machine M_r, $1 \leqslant r \leqslant m$, and $1 \leqslant i \leqslant k$. If m is equal to the total number of machines, and k is equal to the total number of jobs, then $C(m, k)$ means *makespan*. We give an example of FSSP in Table 1, where the total number of jobs is 5, and the total number of machines is 2. The *makespan* of the example is listed in Table 2, that is, $C(2, 5) = 54$.

3. Particle swarm optimization

Particle Swarm Optimization (PSO) is a computational evolution model developed by Kennedy and Eberhart (1995) that can search solution of input problem in the virtual space. PSO is composed by a number of particles in a swarm. Every particle can move its position in the virtual search space, just like a bird flying in the sky. Every particle always remembers its best position in the experience. When a particle moves to another position, it must refer to its best experience and the best experience of all particles in the swarm. A particle's movement is based on Eqs. (2) and (3):

Table 1
The execution time for every operation of every job on every machine

	Job_1	Job_2	Job_3	Job_4	Job_5
M_1	3	4	7	8	10
M_2	12	15	6	10	8

Table 2
Makespan of the example shown in Table 1

	Job_1	Job_2	Job_3	Job_4	Job_5
M_1	3	7	14	22	32
M_2	15	30	36	46	54

$$V_{id} = \omega \times V_{id} + C_1 \times Rand() \times (P_{id}^{best} - P_{id}) + C_2 \times Rand() \times (P_{gid}^{best} - P_{id}) \tag{2}$$

$$P_{id} = P_{id} + V_{id} \tag{3}$$

In Eq. (2), V_{id} means a particle's one-step movement distance, and it is limited to [*VMAX*, *VMAX*] in which *VMAX* is the max movement distance in one step. The variable ω is named inertial weight which can increase or decrease a particle's one step movement distance. C_1 is the self confidence coefficient which means how much this particle believes in its self experience. *Rand()* is a function which can generate a random number between 0 and 1. C_2 is the neighbor confidence coefficient which means how much this particle believes in its neighbor's experience. P_{id} means the position of the *id*th particle. P_{id}^{best} means P_{id}'s best experience. P_{gid}^{best} means the *gid*th particle's best experience, which is the best experience of all particles in the swarm. A particle's next position in virtual search space is given by adding the one-step distance in V_{id} to its current position P_{id}, according to Eq. (3).

Instead of using the fixed coefficients ω, C_1, and C_2 in Eq. (2), it is possible to improve the solution quality by varying them with time (Ratnaweera, Halgamuge, & Watson, 2004; Shi & Eberhart, 1999). In Shi and Eberhart (1999), the researchers observed that if the value of ω is varying with time from high to low, PSO can find a better solution for some problems. In Ratnaweera et al. (2004), the researchers observed that if the value of C_1 is varying with time from high to low, and the value of C_2 is varying with time from low to high, PSO can find a better solution for some problems. In this paper, we are the first one to discuss the influence of the time-varying coefficients ω, C_1, and C_2 for FSSP.

4. The HPSO algorithm

In this paper, we present a new hybrid swarm optimization algorithm named HPSO to solve FSSP. The HPSO algorithm is composed by the following parts: random-key (RK) encoding scheme, individual enhancement (IE) scheme, and the particle swarm optimization (PSO) component.

4.1. RK encoding scheme

The RK encoding scheme can transform a position in the RK virtual space and the FSSP solution space. Every position in the RK virtual space is a vector of real numbers. Every position in the FSSP solution space is a vector of integers. Suppose there is a position A in the RK virtual space that is $(\lambda_1, \lambda_2, \ldots, \lambda_k$, where λ_i, $1 \leqslant i \leqslant k$, is the corresponding weight of job i. We then sort these weights in ascending order with the job indexes. The sorted job indexes correspond to a position in the FSSP solution space. For example, if there is a position A in the RK virtual space that is (0.3, 0.5, 0.2, 0.4), the RK encoding scheme can encode it to (3, 1, 4, 2) which is a position in the FSSP solution space corresponding to a job sequence 3, 1, 4, 2. In this example, a real number of position A corresponds to a job's weight. That is, job 1's weight = 0.3, job 2's weight = 0.5, job 3's weight = 0.2, job 4's weight = 0.4. When we sort these weights in ascending order with the job indexes, we get (3, 1, 4, 2), which is a position in the FSSP solution space.

4.2. Representation of individual

The individual is called particle in the PSO component. We represent a particle as a vector of real numbers. The real numbers of a particle mean a position in the RK virtual space. A position in the RK virtual space can be transformed to a position in the FSSP