Expert Systems with Applications 37 (2010) 2629–2636

Contents lists available at ScienceDirect

Expert Systems with Applications

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

An efficient job-shop scheduling algorithm based on particle swarm optimization ☆

Tsung-Lieh Lin [a], Shi-Jinn Horng [a,b,c,e,*], Tzong-Wann Kao [d], Yuan-Hsin Chen [c], Ray-Shine Run [c], Rong-Jian Chen [c], Jui-Lin Lai [c], I-Hong Kuo [a]

[a] Department of Electrical Engineering, National Taiwan University of Science and Technology, 106 Taipei, Taiwan
[b] Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, 106 Taipei, Taiwan
[c] Department of Electronic Engineering, National United University, 36003 Miao-Li, Taiwan
[d] Department of Electronic Engineering, Technology and Science Institute of Northern Taiwan, Taipei, Taiwan
[e] Department of Computer Science, Georgia State University, United States

ARTICLE INFO

Keywords:
Job-shop scheduling problem
Particle swarm optimization
Multi-type individual enhancement scheme
Random-key encoding scheme
Simulated annealing

ABSTRACT

The job-shop scheduling problem has attracted many researchers' attention in the past few decades, and many algorithms based on heuristic algorithms, genetic algorithms, and particle swarm optimization algorithms have been presented to solve it, respectively. Unfortunately, their results have not been satisfied at all yet. In this paper, a new hybrid swarm intelligence algorithm consists of particle swarm optimization, simulated annealing technique and multi-type individual enhancement scheme is presented to solve the job-shop scheduling problem. The experimental results show that the new proposed job-shop scheduling algorithm is more robust and efficient than the existing algorithms.

1. Introduction

The job-shop scheduling problem (JSSP) is one of the existing combinatorial optimization problems and it has been demonstrated to be an NP-hard problem (Garey, Johnson, & Sethi, 1976; Lawer, Lenstra, Rinooy Kan, & Shmoys, 1993). In the job-shop scheduling problem, each one of n jobs ($n \geqslant 1$) must be processed passing through m machines ($m \geqslant 1$) in a given sequence. The sequence of m machines is different for each different job and cannot be changed during the processing. When one job was processed on a machine, it can be considered as one operation, each job j ($1 \leqslant j \leqslant n$) needs a combination of m operations $(o_{j1}, o_{j2}, \ldots, o_{jm})$ to complete the work. One operation is processed on one of m machines, and just only one operation can be processed at a time. Any job cannot interrupt the machine that is processing one operation of another job. Each machine can process at most one operation at the same time. The main objective of the job-shop scheduling problem is to find a schedule of operations that can minimize the maximum completion time (called makespan) that is the completed time of carrying total operations out in the schedule for n jobs and m machines.

JSSP can be applied to the manufacture processing and effects really the production time and the cost of production for a plant. During the past few decades, JSSP has attracted many researchers to develop algorithms. Because JSSP is an NP-hard problem, it is difficult to develop a perfect algorithm to find a solution within a reasonable time especially for higher dimensions. Recently, many researchers made use of evolution algorithm to solve the problem, such as tabu search method (Nowicki & Smutnicki, 2005; Ponnambalam, Aravindan, & Rajesh, 2000), genetic algorithm (Goncalves, Mendes, & Resende, 2005; Park, Choi, & Kim, 2003; Wang & Zheng, 2001; Watanabe, Ida, & Gen, 2005), simulated annealing (Van Laarhoven, Aarts, & Lenstra, 1992; Steinhöel, Albrecht, & Wong, 1999; Suresh & Mohanasundaram, 2005), ant colony (Udomsakdigool & Kachitvichyanukul, 2008; Zhou, Li, & Zhang, 2004) and particle swarm optimization (Ge, Du, & Qian, 2007; Ge, Sun, Liang, & Qian, 2008; Lian, Gu, & Jiao, 2006). In this paper, we focus on exploiting particle swarm optimization algorithm to achieve the better solution for JSSP.

Particle swarm optimization (PSO) is developed by Kennedy and Eberhart (Kennedy & Eberhart, 1995). The position of one particle is corresponding to a solution of the solving problem. Liking a bird that flies to the food, one particle moves its position to a better solution according to the best particle's experience and its own experience. Every particle moves iteratively until the end of iterations. We call this process as evolution process. At the end of iterations, the position of best particle is the best solution of the solving problem. The original developed PSO is designed to search

☆ This work was supported in part by the National Science Council under Contract Number NSC 96-2918-I-011-002, 97-2221-E-239-022-, 95-2221-E-011-032-MY3.

* Corresponding author. Address: Department of Electrical Engineering, National Taiwan University of Science and Technology, 106 Taipei, Taiwan. Tel.: +886 2 27376700; fax: +886 2 27301081.

E-mail addresses: D8907305@mail.ntust.edu.tw (T.-L. Lin), horngsj@yahoo.com.tw (S.-J. Horng), tkao@ms6.hinet.net (T.-W. Kao), yschen@nuu.edu.tw (Y.-H. Chen), run5116@ms16.hinet.net (R.-S. Run), rjchen@nuu.edu.tw (R.-J. Chen), jllai@nuu.edu.tw (J.-L. Lai), yihonguo@gmail.com (I-Hong Kuo).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.08.015

solution in a continuous space. Because PSO's local search ability is weaker than global searching ability, in order to get better solution, some local search schemes should be integrated with the PSO. In this paper, we embedded a multi-type individual enhancement scheme (MIE) based on simulated annealing technique into particle swarm optimization (PSO). The proposed algorithm enhances the particle's searching ability and is suitable to solve the JSSP. The experimental results show that the proposed PSO with multi-type individual enhancement scheme outperforms the original PSO and is more efficient than those of existing meta-heuristics methods such as discrete particle swarm optimization with simulated annealing model (named HEA (Ge et al., 2007)), discrete particle swarm optimization with artificial immune system (named HIA (Ge, Sun, Liang, & Qian, 2008)) and genetic algorithm (named HGA (Goncalves et al., 2005)) for JSSP scheduling problem, respectively.

The remainder of the paper is organized as follows: an introduction for the job-shop scheduling problem and particle swarm optimization are given in Sections 2 and 3, respectively. Section 4 gives a detailed description of the new proposed job-shop scheduling algorithm. Section 5 discusses the experimental results. Finally, Section 6 summarizes the contribution of this paper.

2. The job-shop scheduling problem

The job-shop scheduling problem (JSSP) consists of n jobs and m machines. Each job must go through m machines to complete its work. We consider one job consists of m operations. Each operation uses one of m machines to complete one job's work for a fixed time interval. Once one operation is processed on a given machine, it cannot be interrupted before it finishes the job's work. The sequence of operations of one job should be predefined and maybe different for any job. In general, one job being processed on one machine is considered as one operation noted as $o_{ji'}$ (means jth job being processed on i'th machine, $1 \leqslant j \leqslant n, 1 \leqslant i' \leqslant m$), then every job has a sequence of m operations. Each machine can process only one operation during the time interval. The objective of JSSP is to find an appropriate operation permutation for all jobs that can minimize the makespan C_{max}, i.e., the maximum completion time of the final operation in the schedule of $n \times m$ operations.

For an $n \times m$ JSSP, the problem can be modeled by a set of m machines, denoted by $M = \{1, 2, \ldots, m\}$, to process a set of $n \times m$ operations, denoted by $o = \{0, 1, 2, \ldots, n \times m + 1\}$. The operations 0 and $n \times m + 1$, which are dummy operations, represent the initial and the last operations, respectively. Dummy operation is used to model the JSSP problem and need not any processing time. A precedence constraint is used to let operation i to be scheduled after all predecessor operations included in P_i are finished. Further, one operation can be scheduled on an appointed machine that is free. For the conceptual model, the notations are defined in the following:

n	number of jobs
m	number of operations for one job
O_i	completed time of operation i $(i = 0, 1, 2, \ldots, n \times m + 1)$
t_i	processing time of operation i on a given machine
ω_{im}	the flag of operation i initiates machine m
P_i	all predecessor operations of operation i
$A(t)$	the set of operations being processed at time t
$o_{ji'}$	i'th operation of job j
C_{max}	makespan

According to the description listed above, the conceptual model of the JSSP can be defined as follows (Goncalves et al., 2005):

$$\text{minimize} \quad O_{n\times m+1} \ (C_{\max}) \tag{1}$$

$$O_q \leqslant O_i - t_i, \quad i = 0, 1, 2, \ldots, n \times m + 1; \ q \in P_i \tag{2}$$

$$\sum_{i \in A(t)} \omega_{im} \leqslant 1, \quad m \in M, \quad t \geqslant 0 \tag{3}$$

$$O_i \geqslant 0, \quad i = 0, 1, 2, \ldots, n \times m + 1 \tag{4}$$

The objective fitness function in Eq. (1) is to minimize makespan that is the completion time of the last operation. The constraint of precedence relationship is defined by Eq. (2). In Eq. (3), it indicates that one machine can process at most one operation at a time. The finish time must be positive by the constraint stated in Eq. (4).

The following example illustrates the JSSP problem.

Suppose there are three jobs and two machines. The processing time and the initiated machine order of each operation are given in Table 1. The operation o_{j1} must be processed before o_{j2} for a job j. In Table 1b, operation o_{21} is processed on machine 2 for 3-unit time interval and operation o_{22} is processed on machine 1 for 1-unit time interval and the operation order of o_{21} should be preceded before that of o_{22}. An operation permutation, $(o_{11}, o_{21}, o_{22}, o_{31}, o_{32}, o_{12})$, is feasible because it satisfied with the operation ordering constraint as stated in Eqs. (1)–(4). O_1 is 2-unit time interval that is the finish time of operation o_{11} on machine 1. Then O_3 is 4-unit time interval as it is the summation of its own operation time and the maximal finished time of its predecessors. According to this permutation, the makespan of $(o_{11}, o_{21}, o_{22}, o_{31}, o_{32}, o_{12})$ is turned out to be 6-unit time interval. The resulting Gantt chart for operation permutation $(o_{11}, o_{21}, o_{22}, o_{31}, o_{32}, o_{12})$ is depicted in Fig. 1.

3. Particle swarm optimization

Particle swarm optimization(PSO) is a novel evolutionary algorithm that was inspired by the motion of a flock of birds searching for foods and was proposed by Kennedy and Eberhart for optimization of continuous non-linear problems (Kennedy & Eberhart, 1995). At the beginning of the evolutionary process, a set of parti-

Table 1
A 3 × 2 JSSP problem.

Job	Operations	
(a) *Operation index*		
Job1	o_{11}	o_{12}
Job2	o_{21}	o_{22}
Job3	o_{31}	o_{32}
(b) *Machine and time*		
Operation	Machine	Time
o_{11}	1	2
o_{12}	2	2
o_{21}	2	3
o_{22}	1	1
o_{31}	2	1
o_{32}	1	1

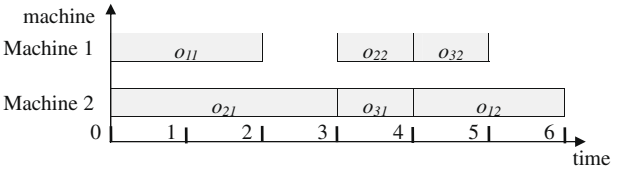

Fig. 1. Gann chart of $(o_{11}, o_{21}, o_{22}, o_{31}, o_{32}, o_{12})$.