G Model
JOCS-241; No. of Pages 7
ARTICLE IN PRESS
Journal of Computational Science xxx (2013) xxx–xxx
Contents lists available at ScienceDirect
Journal of Computational Science
journal homepage: www.elsevier.com/locate/jocs

A parallel competitive colonial algorithm for JIT flowshop scheduling

J. Behnamian*

Department of Industrial Engineering, Faculty of Engineering, Bu-Ali Sina University, Hamedan, Iran

ARTICLE INFO

Article history:
Received 28 February 2013
Received in revised form 25 October 2013
Accepted 9 November 2013
Available online xxx

Keywords:
Bi-objective flowshop
Just-in-time scheduling
Pareto optimum solution
Parallel algorithm
Colonial competitive algorithm
Makespan
Sum of the earliness and tardiness penalties

ABSTRACT

This paper proposes two parallel algorithms which are improved by heuristics for a bi-objective flowshop scheduling problem with sequence-dependent setup times in a just-in-time environment. In the proposed algorithms, the population will be decomposed into the several sub-populations in parallel. Multiple objectives are combined with min–max method then each sub-population evolves separately in order to obtain a good approximation of the Pareto-front. After unifying the obtained results, we propose a variable neighborhood algorithm and a hybrid variable neighborhood search/tabu search algorithm to improve the Pareto-front. The non-dominated sets obtained from our proposed algorithms, a genetic local search and restarted iterated Pareto greedy algorithm are compared. It is found that most of the solutions in the net non-dominated front are yielded by our proposed algorithms.

1. Introduction

In the flowshop (FS) problem, there are m machines located in a fixed order (e.g. 1 through m), and n jobs each of which consists of a sequence of operations on machines (in the order 1 through m). For any job $i \in \{1, \ldots, N\}$ and machine $t \in \{1, \ldots, m\}$, the operation's length of job i on machine t is called its processing time p_i^t. A schedule for the jobs is feasible if (*i*) each machine processes at most one job at any time; (*ii*) for each job, its operations on the machines are processed in the fixed order 1 through m; and (*iii*) each operation (of a job on a machine) is processed without interruption.

The flowshop problem is a special case of acyclic job shop scheduling [1], which in turn is a special case of general job shop scheduling [2]. The FS is an adequate model for the several industrial settings such as semiconductors, electronics manufacturing, airplane engine production, and petrochemical production [3].

Many real-world problems involve simultaneous optimization of several objective functions. In general, these functions often compete and conflict with themselves. Many industries such as aircraft, electronics, semiconductors manufacturing, etc. have tradeoffs in their scheduling problems where multiple objectives need to be considered in order to optimize the overall performance of the system. For reflecting real-world situation adequately, we formulated the scheduling problem as a two-objective optimization problem, i.e., maximum completion time (makespan), and sum of the earliness and tardiness (ET). The use of both objectives is well-justified in practice, as makespan minimization implies the maximization of the throughput and the second objective comes from the make-to-order philosophy in management and production theory: an item should be delivered exactly when it is required by the customer. Therefore, both early and tardy jobs are penalized considering their due dates [4]. Interestingly, the objective of ET problem fits perfectly to the JIT production control policy where an early or a late delivery of an order results in an increase in the production costs.

Moreover, several industries encounter sequence-dependent setup times (SDST) which result in even more difficult scheduling problems. Since machine setup time is a significant factor for production scheduling in many factories; it may easily consume more than 20% of available machine capacity if not well handled [5]. The setup times can be considered either sequence-independent or sequence-dependent. In the sequence-dependent setup times, the length of time required to do the setup depends on both the prior and the current job to be processed [6]. SDST flowshop scheduling can be found in a vast number of industries. Numerous examples are given in the literature, including the plastics manufacturing, rolling slitting in the paper industry [7] and wafer testing in the semiconductor manufacturing [8]. Recently considering sequence-dependent setup times becomes popular among researchers who intend to investigate the scheduling decisions in the real manner [9]. Due to great saving when setup times are explicitly included in the scheduling decisions, we took into account the existence of sequence-dependent setup times in our problem.

* Tel.: +98811 8292505; fax: +98811 8292631.
E-mail address: Behnamian@basu.ac.ir

1877-7503/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jocs.2013.11.002

Scheduling problems with sequence-dependent setup times are among the most difficult classes of scheduling problems. Since the single machine sequence-dependent setup times scheduling problem is equivalent to a traveling salesman problem (TSP) [5] and TSP is NP-hard, our problem is at least that difficult, so it is also NP-hard and the use of metaheuristics is appropriate for it. Our goal in this paper is to develop a parallel competitive colonial algorithm and improved it by a hybrid variable neighborhood search/Tabu search algorithm (namely PCVT) to SDST flowshop with bi-objective that gives a set of compromise (non-dominated) solutions, so that, these solutions should represent a good approximation to the Pareto optimal front.

The paper has the following structure. Section 2 gives literature review. Section 3 introduces the multi-objective optimization. The characteristics of the proposed algorithm approach are described Section 4. Section 5 presents the experimental design which compares the results. Finally, Section 6 devotes to conclusions and future works.

2. Literature review

Various genetic algorithms have ever been derived for bi-objective or multi-objective optimization problems. Schaffer [10] proposed Vector Evaluated Genetic Algorithm (VEGA) to solve the Pareto-optimal solution of multi-objective problem. The VEGA is the first method modifying the genetic algorithm (GA) to solve multi-objective problems. Murata and Ishibuchi [11] proposed a multi-objective genetic algorithm (MOGA). One characteristic of MOGA is using the dynamic weighting to transform the multiple objectives into a single objective, which it randomly assigns different weight values to the different objectives. Zitzler et al. [12] modified Strength Pareto Evolutionary Algorithm (SPEA) as SPEA II for multi-objective optimization. Non-dominated Sorting Genetic Algorithm-II (NSGA2) was proposed by Deb et al. [13], where the elitism strategy was modified. Besides, in order to keep the solutions' diversity, the algorithm also provided a crowding distance to measure the density of individuals in the solution space. In addition, some sub-population like approaches also can be found in the related literatures, such as a segregative genetic algorithms [14], multi-population genetic algorithm [15], hierarchical fair competition model [16], multi-objective particle swam optimization [17] and two phases sub-population GA for parallel machines scheduling problems [18]. Chang et al. [19] proposed a modified SPGA and an adaptive SPGA to solve the parallel machines scheduling problem with total tardiness time and makespan objective functions. Arroyo and Armentano [20] proposed a genetic local search algorithm for multi-objective flowshop scheduling problems.

Mansouri et al. [21] considered a two-machine flowshop scheduling problem with sequence-dependent setup times. To minimize setups and makespan, they designed two multi-objective meta-heuristics based on genetic algorithm and simulated annealing. In this paper, the performances of these approaches are compared with lower bounds for the small-sized problems and in the larger problems, performance of the proposed algorithms are compared with each other. Eren [22] proposed three heuristic approaches for a bi-objective m-machine flowshop scheduling with sequence-dependent setup times. The objective function of problem is minimization of the weighted sum of total completion time and makespan. Minella et al. [23] proposed an Iterated Greedy technique for solving the multi-objective permutation flowshop problem. Their proposed algorithm was characterized by an initialization of the population, management of the Pareto front, and a specially tailored local search, among other things. Furthermore, in this research, the authors used a graphical tool to compare the performances of stochastic Pareto fronts obtained by the proposed multi-objective Iterated Greedy method with empirical attainment functions. Dubois-Lacoste et al. [24] proposed a hybrid two-phase local search and a Pareto local search algorithm for several bi-objective permutation flowshop scheduling problems, i.e., pairwise combinations of the objectives included (*i*) makespan, (*ii*) sum of the completion times of the jobs, and (*iii*) both, the weighted and non-weighted total tardiness of all jobs.

Khalili and Tavakkoli-Moghaddam [25] dealt with a bi-objective flowshop scheduling problem in which all jobs may not be processed by all machines. Furthermore, they considered transportation times among machines for minimizing the makespan and total weighted tardiness and proposed an electromagnetism algorithm for it. Chung and Tong [26] considered a bi-criteria scheduling problem in a permutation flowshop environment with varied learning effects on different machines. To minimize the weighted sum of total completion time and makespan, a dominance criterion and a lower bound are proposed to accelerate the branch-and-bound algorithm for deriving the optimal solution. Also, they proposed two heuristic algorithms for this problem. Ciavotta et al. [27] considered a permutation flowshop scheduling with multi-objective functions and sequence-dependent setup times. In that study, they proposed a Restarted Iterated Pareto Greedy algorithm (RIPG) and compared it against the best performing approaches from the relevant literature.

3. Pareto-optimality-based algorithm

Many decision problems contain a large, possibly infinite number of decision alternatives. In such cases, it is impossible to explicitly compare all alternatives, and therefore the choice problem is accompanied by a search problem to filter promising (optimal) from unpromising (non-optimal) alternatives. Problems of this type are treated in the area called multi-objective optimization. Multi-objective optimization was originally conceived with a goal of finding Pareto optimal solutions, also called efficient solutions. Such solutions are non-dominated, i.e., no other solutions are superior to them when all objectives are taken into account. A typical classification of methods for a bi-objective decision making is given by Hwang and Masud [30], who distinguish four classes according to when the decision maker's preferences enters into the formal decision making process. These different possibilities are:

- No articulation of preference information (only search).
- Priori aggregation of preference information (choice before search).
- Progressive articulation of preference information (integration of search and choice).
- Posteriori articulation of preference information (search before choice).

In this paper, we combined "no articulation of preference information" and "posteriori approach" methods. Now, let us consider a problem where we want to minimize two conflicting objective functions $f_1(x)$ and $f_2(x)$, simultaneously, subject to a general constraint $x \in S$. The vector of objective functions, called *objective vector*, is denoted by $F(x) = (f_1(x), f_2(x))^T$, and the vector $x = (x_1, x_2, \ldots, x_n)^T$ is called a *decision vector*.

4. Proposed algorithm

4.1. Parallel competitive colonial algorithm

The basic idea of proposed algorithms is to decompose the population into several sub-populations and assign the different weights to each of them [19]. Each sub-population is just like a squad team