Int. J. Production Economics 141 (2013) 167–178

Contents lists available at SciVerse ScienceDirect

Int. J. Production Economics

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

A hybrid artificial bee colony algorithm for the job shop scheduling problem

Rui Zhang [a,*], Shiji Song [b], Cheng Wu [b]

[a] School of Economics and Management, Nanchang University, Nanchang 330031, PR China
[b] Department of Automation, Tsinghua University, Beijing 100084, PR China

ARTICLE INFO

Article history:
Received 13 October 2011
Accepted 31 March 2012
Available online 12 April 2012

Keywords:
Job shop scheduling problem
Artificial bee colony algorithm
Tree search
Neighborhood property

ABSTRACT

The job shop scheduling problem (JSSP) has attracted much attention in the field of both information sciences and operations research. In terms of the objective function, most existing research has been focused on the makespan criterion (i.e., minimizing the overall completion time). However, for contemporary manufacturing firms, the due date related performance is usually more important because it is crucial for maintaining a high service reputation. Therefore, in this study we aim at minimizing the total weighted tardiness in JSSP. Considering the high complexity, a novel artificial bee colony (ABC) algorithm is proposed for solving the problem. A neighborhood property of the problem is discovered, and then a tree search algorithm is devised to enhance the exploitation capability of ABC. According to extensive computational tests, the proposed approach is efficient in solving the job shop scheduling problem with total weighted tardiness criterion.

1. Introduction

The job shop scheduling problem (JSSP) has been known as a very challenging combinatorial optimization problem since 1950s. Because JSSP is $\mathcal{NP}$-hard in the strong sense (Lenstra et al., 1977), it is by no means easy to guarantee the optimal solution even for small-scale JSSP instances. In the recent years, meta-heuristics have become extremely popular as practical optimization methods for solving JSSPs. Successful meta-heuristics include but are not limited to genetic algorithm (GA) (Essafi et al., 2008; Al-Hinai and ElMekkawy, 2011), tabu search (TS) (Vilcot and Billaut, 2011; Li et al., 2011), particle swarm optimization (PSO) (Sha and Hsu, 2006; Moslehi and Mahnam, 2011), ant colony optimization (ACO) (Huang, 2010; Xing et al., 2010) and memetic algorithm (MA) (Lacomme et al., 2012; Gao et al., 2011).

In the standard JSSP, a set of n jobs $\mathcal{J}=\{J_j\}_{j=1}^n$ are waiting to be processed by a set of m machines $\mathcal{M}=\{M_k\}_{k=1}^m$ under some basic assumptions, the most important of which are listed as follows: (i) Each machine can process at most one job at a time; (ii) Each job can be processed by at most one machine at a time. Each job consists of a group of operations to be performed by each machine in a predetermined order. The duration time of each operation is fixed and known. Besides, a preset due date d_j and a preset weight w_j are given for each job. Due date is the preferred latest finishing time of a job, so completion after this specific time will result in losses such as a worsened reputation among customers. Weights reflect the level of importance of the orders from different customers, larger values suggesting higher strategic importance. If we use C_j to denote the completion time of job j, the objective function of JSSP can be makespan ($C_{\max}=\max_{j=1}^n\{C_j\}$), maximum lateness ($L_{\max}=\max_{j=1}^n\{L_j\}$), total tardiness ($TT=\sum_{j=1}^n T_j$) and total weighted tardiness ($TWT=\sum_{j=1}^n w_jT_j$), where lateness is defined as $L_j=C_j-d_j$ while tardiness is defined as $T_j=\max\{0,C_j-d_j\}$.

Up till now, most research on JSSP has been focused on the makespan criterion. However, due date related performances are becoming more relevant for the firms that adopt the make-to-order (MTO) manufacturing strategy nowadays. Such a firm will first "quote" a due date[1] for each order that it receives, and once the due date has been fixed, the firm must make every effort to deliver the products before or on the due date. Late delivery will result in economic losses in various forms. In this sense, the total weighted tardiness measure better reflects the critical factors that affect the profits of a MTO firm. However, we are by no means claiming that other objective functions are unimportant. In fact, when the due dates are not very tight to form harsh constraints,

[1] Due date quotation is also a significant decision problem for MTO firms. A tight due date setting will increase the possibility of tardiness despite its appeal for the potential customers. A loose due date setting, on the other hand, will hardly be competitive although it relieves pressure on the production scheduling function. Therefore, due date quotation has been studied as an optimization problem in many publications such as Vinod and Sridharan (2011), Zhang and Wu (2012), and Baykasoğlu and Göçken (2009). However, in the traditional scheduling research, due dates are regarded as exogenous variables, i.e., it is assumed that the due dates for each job are already preset and known in advance.

* Corresponding author.
E-mail address: r.zhang@ymail.com (R. Zhang).

0925-5273/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ijpe.2012.03.035

makespan or flowtime can be an important performance measure for production efficiency.

Meanwhile, total weighted tardiness is more difficult to optimize than makespan. This can be explained from two aspects. Intuitively, C_{max} is determined by the completion time of the (probably unique) bottleneck job while *TWT* is determined by the completion times of all the tardy jobs. Especially when the due dates are tight, optimizing *TWT* means the algorithm has to consider a large number of critical paths simultaneously. Making tradeoffs is even more difficult in the case where the weights of jobs are widely different. Therefore, compared with C_{max}, the objective function *TWT* is more sensitive to slight changes in the production schedule, which makes the optimization process more complicated.

From the theoretical aspect, the following relationship exists between different objective functions under the same machine environment (Pinedo, 2008): $\alpha|\beta|C_{max} \propto \alpha|\beta|L_{max} \propto \alpha|\beta|\sum T_j \propto \alpha|\beta|\sum w_j T_j$ (following the three-field notation; Jain and Meeran, 1999). "$P_1 \propto P_2$" indicates that problem P_1 can be reduced to problem P_2, and thus the difficulty of solving P_2 is at least the same as that of solving P_1. The first reduction relation is straightforward by letting $d_j = 0, \forall j$. The last reduction is also obvious by letting $w_j = 1$, $\forall j$. For the proof of the second reduction relation, we should notice that $L_{max} \le z \Leftrightarrow C_j - (d_j + z) \le 0$, $\forall j$. By letting $d'_j = d_j + z$ and $T'_j = \max\{0, C_j - d'_j\}$, we obtain $L_{max} \le z \Leftrightarrow \sum T'_j = 0$.

Therefore, *TWT* includes many other objective functions (e.g. C_{max}, L_{max} and $\sum w_j C_j$) as special cases. The optimization difficulty of *TWT* is thus higher than these objectives. Below we will use the abbreviation "TWT-JSSP" to denote the job shop scheduling problem with total weighted tardiness objective.

The rest of this paper is organized as follows. Section 2 provides a brief review on the existing solution methods for TWT-JSSP and the artificial bee colony algorithm. Section 3 discusses the mathematical model of TWT-JSSP and a neighborhood property. Section 4 describes the design of a new artificial bee colony algorithm for solving TWT-JSSP. Section 5 presents the computational results. Finally, Section 6 concludes the paper.

2. Literature review

2.1. Existing algorithms for TWT-JSSP

The contributions on TWT-JSSP are relatively rare in the literature. The only exact solution methodology is the branch-and-bound algorithm proposed by Singer and Pinedo (1998), while all the following surveyed methods belong to the heuristic category.

Kutanoglu and Sabuncuoglu (1999) present efficient dispatching rules for sequencing the operations in TWT-JSSP, the most powerful one being the ATC (apparent tardiness cost) rule. Mason et al. (2002) and Mönch and Drießel (2005) propose modified shifting bottleneck heuristics, in which the subproblems are solved by dispatching rules (the basic ATC rule or the BATCS rule for complex job shops). The large step random walk (LSRW) algorithm is designed by Kreipl (2000) and aimed at the TWT-JSSP with release times ($Jm|r_j|\sum w_j T_j$). Essafi et al. (2008) and Zhou et al. (2009) present hybrid genetic algorithms for TWT-JSSP. Singer (2001) presents a rolling horizon approach for TWT-JSSP, in which each time window is scheduled with a modified shifting bottleneck heuristic. Bontridder (2005) presents a tabu search algorithm for the generalized TWT-JSSP with release times and precedence constraints. Recently, a new meta-heuristic called electromagnetic algorithm has been hybridized with simulated annealing (Tavakkoli-Moghaddam et al., 2009) and applied to TWT-JSSP.

Although these algorithms are effective, they have not fully utilized the inherent properties of TWT-JSSP. Hence, they may not perform satisfactorily when faced with large-scale instances of the problem. Exploring the structural properties (including neighborhood properties) of TWT-JSSP is substantially important for conquering the problem (Wolpert and Macready, 1997), which also constitutes the motivation of this study.

2.2. The artificial bee colony algorithm

The artificial bee colony (ABC) algorithm is a relatively new swarm intelligence based optimizer. It mimics the cooperative foraging behavior of a swarm of honey bees (Karaboga and Akay, 2009b). ABC was initially proposed by Karaboga (2005) for optimizing multi-variable and multi-modal continuous functions. The latest research has revealed some good properties of ABC (Karaboga and Basturk, 2008, 2007; Karaboga and Akay, 2009a). Especially, the number of control parameters in ABC is fewer than that of other population-based algorithms, which makes it easier to be implemented. Meanwhile, the optimization performance of ABC is comparable and sometimes superior to the state-of-the-art meta-heuristics. Therefore, ABC has aroused much interest and has been successfully applied to different kinds of optimization problems (Kang et al., 2009; Sonmez, 2011; Samanta and Chakraborty, 2011).

The ABC algorithm systematically incorporates exploration and exploitation mechanisms, so it is suitable for solving complex scheduling problems. However, due to its continuous feature, the traditional ABC algorithm cannot be directly applied to scheduling problems with inherent discrete nature. Indeed, in canonical ABC, each solution is represented by a vector of floating-point numbers. But for scheduling problems, each solution is naturally a permutation of integers. To address this issue, two kinds of approaches can be identified in the literature.

(1) A transformation scheme is established to convert permutations into real numbers and vice versa (Shi et al., 2010). In this way, we only need to add a few lines to the encoding and decoding procedures, and it is not necessary to change the implementation of ABC itself.
(2) The search operators in ABC are modified to suit the permutation representation (Pan et al., 2011). The redesign of these operators should be problem-dependent and thus requires a specific analysis of the optimization problem.

In this paper, we use ABC as the basic optimization framework for solving TWT-JSSP, and meanwhile we will combine the above two treatments in the hope of devising a more effective ABC. To our knowledge, this is the first attempt that ABC is applied to TWT-JSSP.

3. Mathematical properties of TWT-JSSP

3.1. The mathematical model and its duality

We utilize the concept of disjunctive graph for formulating TWT-JSSP. In the graph $G(N,A,E)$, $N = O \cup \{0\} = \{0,1,2,\ldots,n \times m\}$ is the set of nodes, where $O = \{1,\ldots,n \times m\}$ corresponds to the operations of the JSSP instance. Node 0 stands for a dummy operation which starts before all the real operations, and the starting time and the processing time of this dummy operation are both zero. A is the set of conjunctive arcs, which indicate the processing order of the operations belonging to the same job. Meanwhile, node 0 is connected with the first operation of each job with a separate conjunctive arc. In other words, if we use $H(O)$