Expert Systems with Applications 42 (2015) 6155–6167

Contents lists available at ScienceDirect

Expert Systems with Applications

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

An efficient Discrete Artificial Bee Colony algorithm for the blocking flow shop problem with total flowtime minimization

Imma Ribas [a,*], Ramon Companys [b], Xavier Tort-Martorell [c]

[a] Departament d'Organització d'Empreses, DOE – ETSEIB – Universitat Politècnica de Catalunya, BarcelonaTech, Avda. Diagonal, 647, 7th Floor, 08028 Barcelona, Spain
[b] CDE – EPSEB – Universitat Politècnica de Catalunya, BarcelonaTech, Gregorio Marañón 44-50, 3rd Floor, 08028 Barcelona, Spain
[c] Departament de Estadística e investigación Operativa – ETSEIB – Universitat Politècnica de Catalunya, BarcelonaTech, Avda. Diagonal, 647, 6th Floor, 08028 Barcelona, Spain

ARTICLE INFO

Article history:
Available online 11 April 2015

Keywords:
Scheduling
Flow shop
Blocking
Total flowtime
Heuristics

ABSTRACT

This paper presents a high performing Discrete Artificial Bee Colony algorithm for the blocking flow shop problem with flow time criterion. To develop the proposed algorithm, we considered four strategies for the food source phase and two strategies for each of the three remaining phases (employed bees, onlookers and scouts). One of the strategies tested in the food source phase and one implemented in the employed bees phase are new. Both have been proved to be very effective for the problem at hand. The initialization scheme named HPF2(λ, μ) in particular, which is used to construct the initial food sources, is shown in the computational evaluation to be one of the main procedures that allow the DABC_RCT to obtain good solutions for this problem. To find the best configuration of the algorithm, we used design of experiments (DOE). This technique has been used extensively in the literature to calibrate the parameters of the algorithms but not to select its configuration. Comparing it with other algorithms proposed for this problem in the literature demonstrates the effectiveness and superiority of the DABC_RCT.

1. Introduction

The blocking flow shop scheduling problem allows many productive systems to be modeled when there are no buffers between consecutive machines. Some industrial examples can be found in the production of concrete blocks, where storage is not allowed in some stages of the manufacturing process (Grabowski & Pempera, 2000); in the iron and steel industry (Gong, Tang, & Duin, 2010); in the treatment of industrial waste and the manufacture of metallic parts (Martinez, Dauzère-Pérès, Guéret, Mati, & Sauer, 2006); or in a robotic cell, where a job may block a machine while waiting for the robot to pick it up and move it to the next stage (Sethi, Sriskandarajah, Sorger, Blazewicz, & Kubiak, 1992). In general, it is useful for those systems that have a production line without a drag system that forces a job to be transferred between two consecutive stations at pre-established times. In this type of production configuration, a machine can be blocked by the job it has processed if the next machine is not available. Hence, accurate scheduling is necessary to minimize machine blocking and idle time, which allows increasing the productivity level.

Although the blocking flow shop scheduling problem has not been as extensively studied as the permutation flow shop problem, several types of metaheuristics have been proposed to solve the former in order to minimize makespan: a genetic algorithm (GA) (Caraffa, Ianes, Bagchi, & Sriskandarajah, 2001); two tabu search (TS) algorithms (Grabowski & Pempera, 2007); a hybrid genetic algorithm (HGA) (Wang, Zhang, & Zheng, 2006); a particle swarm optimization algorithm (HPSO) (Liu, Wang, & Jin, 2008); a differential evolution (DE) algorithm (Qian, Wang, Huang, & Wang, 2009); a hybrid discrete differential evolution algorithm (Wang, Pan, Suganthan, Wang, & Wang, 2010); a hybrid harmony search (Wang, Pan, & Tasgetiren, 2011); an iterated greedy algorithm (Ribas, Companys, & Tort-Martorell, 2011); a simulated annealing algorithm with a local search (Wang, Song, Gupta, & Wu, 2012); a discrete self-organizing migrating algorithm (Davendra & Bialic-Davendra, 2013); a variable neighborhood search (Ribas, Companys, & Tort-Martorell, 2013a); a Memetic algorithm (Pan, Wang, Sang, Li, & Liu, 2013); an artificial immune system (Lin & Ying, 2013); and a Discrete Artificial Bee Colony (Han, Gong, & Sun, 2014).

However, little research has been done to solve the blocking flow shop scheduling problem in ways that include other interesting criteria for the industry, such as total tardiness or total flowtime. For the former, Armentano and Ronconi (2000) proposed a

* Corresponding author. Fax: +34 93 401 60 54.
E-mail addresses: imma.ribas@upc.edu (I. Ribas), ramon.companys@upc.edu (R. Companys), xavier.tort@upc.edu (X. Tort-Martorell).

http://dx.doi.org/10.1016/j.eswa.2015.03.026
0957-4174/

Tabu Search procedure, Ronconi and Henriques (2009) a new NEH-based method and a GRASP algorithm and Ribas, Companys and Tort-Martorell (2013b) proposed an iterated local search method. For the latter, Wang, Pan, and Fatih Tasgetiren (2010) proposed a hybrid Harmony Search (HS) algorithm, Deng, Xu, and Gu (2012) a Discrete Artificial Bee Colony (DABC) algorithm, and Moslehi and Khorasanian (2013) a branch and bound algorithm that can be used in small instances. The criterion of minimizing total flowtime has been found to be an important real-life objective in industries, since it results in the even utilization of resources, even turnover of finished jobs and reduced in-process inventory. Thus, it is considered to be more relevant and meaningful in today's dynamic production environment (Liu & Reeves, 2001). Therefore, it is interesting to expand on the existing research in order to have efficient scheduling procedures available for sequencing jobs in productive environments that can be modeled as the blocking flow shop problem with total flowtime criterion.

One of the recent swarm metaheuristics that has successfully been applied to several optimization problems is the Artificial Bee Colony (ABC) algorithm proposed by Karaboga (2005). Although the ABC algorithm was described for solving numerical problems, discrete versions have been introduced to solve several combinatorial problems. A complete review of papers published up to 2012 about the ABC algorithm and its applications can be found in (Karaboga, Gorkemli, Ozturk, & Karaboga, 2014). In particular, some Discrete Artificial Bee Colony (DABC) algorithms have been proposed in the field of scheduling to solve several scheduling problems under different constraints and/or objective functions. Nasiri (2015) presents a DABC algorithm for the stage shop problem, which is a special case of the general shop scheduling problem. Pan, Wang, Li, and Duan (2014) present it for the hybrid flow shop scheduling problem to minimize the makespan and Li and Pan (2014) for the hybrid flow shop scheduling problem with limited buffers. Wang, Zhou, Xu, Wang, and Liu (2012) applied a DABC algorithm to the flexible job-shop scheduling problem; Zhang, Song, and Wu (2013) to the job-shop scheduling problem for minimizing the total weighted tardiness; Li, Pan, and Gao (2011) and Wang, Zhou, Xu, Wang, and Liu (2011) proposed a multi-objective DABC algorithm for the flexible job-shop scheduling problem; and Lei (2012) proposed it for the interval job-shop scheduling problem with non-resumable jobs and flexible maintenance. Finally, for the permutation flow shop scheduling problem: Liu and Liu (2013) present a DABC procedure for makespan minimization; and Tasgetiren, Pan, Suganthan, and Chen (2011a) for flowtime minimization. Deng et al. (2012) and Han et al. (2012) considered the blocking constraint and the total flowtime criterion, whereas Tasgetiren, Pan, Suganthan, and Oner (2013) considered the no-idle constraint for total tardiness minimization.

The blocking flow shop problem, denoted as $Fm|block|\sum C_i$, according to the notation proposed by Graham, Lawler, Lenstra, and Rinnooy Kan (1979), can be defined as follows. A set of n jobs have to be processed by m machines in the same order, implying that a job sequence determined for machine 1 is kept throughout the system. Each job i, $i \in \{1,2,\dots,n\}$ requires a fixed positive processing time $p_{j,i}$ on every machine j, $j \in \{1,2,\dots,m\}$. Jobs and machines are available from time zero onwards. Our objective is to find a job processing sequence that minimizes the total flowtime. $Fm|block|\Sigma C_i$ can be modeled with the following equations, where $[k]$ is the index of the job in the kth position in the permutation, $e_{j,k}$ denotes the time at which job $[k]$ begins to be processed by machine j, and $c_{j,k}$ is the departure time of job $[k]$ from machine j. Note that if job $[k]$ can leave machine j when it is completed, which depends on the availability of machine $j+1$, then $c_{j,k}$ is not only the departure time but also the completion time of job $[k]$ on machine j:

$$e_{j,k} + p_{j,[k]} \leqslant c_{j,k} \quad j = 1,2,\dots,m \quad k = 1,2,\dots,n \tag{1}$$

$$e_{j,k} \geqslant c_{j,k-1} \quad j = 1,2,\dots,m \quad k = 1,2,\dots,n \tag{2}$$

$$e_{j,k} \geqslant c_{j-1,k} \quad j = 1,2,\dots,m \quad k = 1,2,\dots,n \tag{3}$$

$$c_{j,k} \geqslant c_{j+1,k-1} \quad j = 1,2,\dots,m \quad k = 1,2,\dots,n \tag{4}$$

$$TF = \sum_{k=1}^{n} c_{m,k} \tag{5}$$

with $c_{j,0} = 0\ \forall j,\ c_{0,k} = 0,\ c_{m+1,k} = 0\ \forall k$ being the initial conditions.

If Eqs. (2) and (3) are summarized as (6) and Eqs. (1) and (4) as (7), the schedule obtained is semi-active, which is interesting because an optimal solution can be found in the subset of the semi-active set of solutions.

$$e_{j,k} = \max\{c_{j,k-1}; c_{j-1,k}\} \quad j = 1,2,\dots,m \quad k = 1,2,\dots,n \tag{6}$$

$$c_{j,k} = \max\{e_{j,k} + p_{j,[k]}, c_{j+1,k-1}\} \quad j = 1,2,\dots,m \quad k = 1,2,\dots,n \tag{7}$$

The aim of this paper is to propose an efficient DABC algorithm, named DABC_RCT, for the blocking flow shop problem with total flowtime criterion. To develop the proposed algorithm, we considered four strategies for the food source phase and two strategies for each of the three remaining phases (employed bees, onlookers and scouts). One of the strategies tested in the food source phase and one implemented in the employed bees phase are new. Both have been proved to be very effective. The initialization scheme named HPF2(λ,μ) in particular was used to construct the initial food sources, which the computational evaluation has shown to be one of the main procedures that allow the DABC_RCT to obtain good solutions for this problem. To find the best configuration of the algorithm, we used design of experiments (DOE). This technique has been extensively used in the literature to calibrate the parameters of the algorithms but not to select its configuration. Comparing it with other algorithms proposed in the literature for this problem demonstrates the effectiveness and superiority of the DABC_RCT.

The rest of the paper is organized as follows. Section 2 describes the different strategies tested in each phase of the algorithm; Section 3 shows the design of experiments done to choose the best combination of strategies; Section 4 shows the computational evaluation of the algorithms; and, finally, Section 5 is devoted to conclusions and future work.

2. Proposed alternatives for a Discrete Artificial Bee Colony algorithm

The ABC algorithm is a swarm intelligence technique inspired by the intelligent foraging behavior of honey bees. This algorithm has three essential components: food sources, which are the set of current solutions; the employed bees that are associated with a particular food source to be exploited; and unemployed bees. The unemployed bees are made up of two types: onlookers, who wait in the nest and establish a food source through the information shared by the employed; and scouts, who search for new food sources in the area surrounding the hive. There are several strategies to implement in each part of the algorithm, and each combination can lead to a different Discrete Artificial Bee Colony algorithm. The point is to know which strategy and which combination among them has to be used in order to enhance the performance of the algorithm for the problem at hand. The final configuration of the algorithm was set by means of a design of experiments, which are explained in Section 4.