Information Sciences 181 (2011) 3459–3475

Contents lists available at ScienceDirect

Information Sciences

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

A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops

M. Fatih Tasgetiren [a], Quan-Ke Pan [b,*], P.N. Suganthan [c], Angela H-L Chen [d]

[a] Department of Industrial Engineering, Yasar University, Bornova, Izmir, Turkey
[b] College of Computer Science, Liaocheng University, Liaocheng, 252059, PR China
[c] School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
[d] The Department of Finance, Nanya Institute of Technology, Taoyuan 320, Taiwan, ROC

ARTICLE INFO

Article history:
Received 16 February 2010
Received in revised form 4 April 2011
Accepted 10 April 2011
Available online 21 April 2011

Keywords:
Permutation flowshop scheduling problem
Iterated greedy algorithm
Discrete differential evolution algorithm
Discrete artificial bee colony algorithm
Estimation of distribution algorithm
Genetic local search

ABSTRACT

Obtaining an optimal solution for a permutation flowshop scheduling problem with the total flowtime criterion in a reasonable computational timeframe using traditional approaches and optimization tools has been a challenge. This paper presents a discrete artificial bee colony algorithm hybridized with a variant of iterated greedy algorithms to find the permutation that gives the smallest total flowtime. Iterated greedy algorithms are comprised of local search procedures based on insertion and swap neighborhood structures. In the same context, we also consider a discrete differential evolution algorithm from our previous work. The performance of the proposed algorithms is tested on the well-known benchmark suite of Taillard. The highly effective performance of the discrete artificial bee colony and hybrid differential evolution algorithms is compared against the best performing algorithms from the existing literature in terms of both solution quality and CPU times. Ultimately, 44 out of the 90 best known solutions provided very recently by the best performing estimation of distribution and genetic local search algorithms are further improved by the proposed algorithms with short-term searches. The solutions known to be the best to date are reported for the benchmark suite of Taillard with long-term searches, as well.

1. Introduction

Flowshop scheduling is among the most prevalent problems in the field of deterministic scheduling in engineering industries [3,14,22,29,47]. The permutation flowshop represents a particular case in which solutions are represented by the permutations of n jobs, i.e., $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$. Each job comprises a set of m operations that must be performed by different machines. Each machine can process only one operation at a time. While all jobs have the same permutations on every machine, these jobs, once initiated, cannot be interrupted (preempted), and the release times of all jobs are zero. Given the processing time p_{jk} for job j on machine k, we consider the total flowtime criterion as the objective function to be minimized.

In the above specified context, $F(\pi_j)$, denoted by the flowtime of job π_j, is equivalent to the completion time $C(\pi_j, m)$ of job π_j on the last machine m because the release times of all jobs are zero. As a result, the total flowtime $TFT(\pi)$ of a permutation π can be computed by summing up flowtimes or completion times of all jobs and defined as $TFT(\pi) = \sum_{j=1}^{n} F(\pi_j) = \sum_{j=1}^{n} C(\pi_j, m)$. Then the optimal permutation $\pi^* = \{\pi_1^*, \pi_2^*, \ldots, \pi_n^*\}$ in the set of all permutations of Π is determined by

* Corresponding author.
E-mail addresses: fatih.tasgetiren@yasar.edu.tr (M.F. Tasgetiren), panquanke@gmail.com (Q.-K. Pan), epnsugan@ntu.edu.sg (P.N. Suganthan), achen@nanya.edu.tw (A.H-L Chen).

0020-0255/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2011.04.018

$TFT(\pi^*) \leqslant TFT(\pi)$ for each permutation π belonging to Π. Under these specifications, the completion time for the n-job and m-machine problem is computed as follows:

$$C(\pi_1, 1) = p_{\pi_1,1} \tag{1}$$

$$C(\pi_j, 1) = C(\pi_{j-1}, 1) + p_{\pi_j,1} \quad j = 2, \ldots, n \tag{2}$$

$$C(\pi_1, k) = C(\pi_1, k-1) + p_{pi_1,k} \quad k = 2, \ldots, m \tag{3}$$

$$C(\pi_j, k) = \max\{C(\pi_{j-1}, k), C(\pi_j, k-1) + p_{\pi_j,k}\} \quad j = 2, \ldots, n; \quad k = 2, \ldots, m \tag{4}$$

Many different algorithms have been proposed over time in an attempt to find the exact solution to minimizing the total flowtime (TFT). Several variants of branch and bound algorithms were developed [4,5,11,41]; Ignall and Scharge [11] were the first to apply the branch and bound scheme based on two lower bounds in the two-machine flow shop problem. Then Bansal [4] extended their idea to the m-machine case. Recent publications [5,41] have detailed the development of lower bounding methods either based on Lagrangian relaxation or by introducing slack variables. These exact methods have been successfully implemented in a limited number of small instances due to lengthy execution times. For large instances in the n-job and m- machine total flowtime problem, some efficient heuristics have been developed in [2,7,8,10,23,26,31,45]. The NEH constructive method, proposed by Nawaz et al. [28], was claimed to be the best for makespan minimization in flow shops, but not effective for total flowtime minimization. Therefore, to minimize total flowtime, the heuristics in both Woo and Yim [45] and Framinan and Leisten [7] were founded on different insertion schemes from NEH. Furthermore, a composite heuristic proposed by Allahverdi and Aldowaisan [2] adopted the insertion with pair-wise exchange from Framinan and Leisten [7] to improve their solutions. So far, no heuristic is optimal for total flowtime minimization. In comparison to heuristic methods, metaheuristic methods always obtain better results, as they compose many different kinds of algorithmic components [9,12,21,27,32,42,43,46,48]. Very recently, an estimation of distribution algorithm (EDA) hybridized with variable neighborhood search (VNS) has been introduced in [13]. In addition to EDA, two genetic local search algorithms have been proposed in [24,25]. The first one employs a local search called insertion search with cut and repair, denoted by hGLS, and the second one is the genetic algorithm hybridized with a tabu search, denoted by tsGLS. These three algorithms improved almost all the best known solutions in the existing literature.

Among metaheuristics, modeling the collective behavior of self-organized systems and applying these models to solve real-world problems has been ongoing and has become a class of its own, known as swarm intelligence. Earlier works implementing the ant colony optimization (ACO) and particle swarm optimization (PSO) algorithms were conducted to simulate the swarm behavior of ant colonies and flocks of birds, respectively. Recently, some algorithms were proposed by modeling the specific intelligent behaviors of honeybee swarms [1,15–19,29,40,44]. Karaboga in [15–19] introduced an artificial bee colony (ABC) algorithm to optimize multi-variable and multi-modal continuous functions. Numerical comparisons demonstrated that the performance of the ABC algorithm is as competitive as other population-based algorithms with the advantage of employing fewer control parameters [15–19]. Furthermore, a discrete version of the ABC algorithm has been recently applied to the lot-streaming flowshop scheduling problem in [29]. Tereshko attempted to model the forage behavior of a honeybee colony based on reaction–diffusion equations [40]. Wede and Farooq proposed a routing algorithm, called BeeAdHoc, for energy efficient routing in mobile ad hoc networks [44]. Clearly, swarm intelligence can be understood as an algorithmic framework inspired by the aggregate behavior of the social insects and animals [1]. It has drawn the attention of researchers because of its advantages such as scalability, fault tolerance, adaptation, speed, modularity, autonomy, and parallelism [20].

As there is no detailed work that describes the use of the ABC algorithm to deal with the PFSP under the TFT criterion, we present a novel discrete ABC (DABC) algorithm as well as the hybrid version of our previous discrete differential evolution (hDDE) algorithm in [30] in order to solve the PFSP with the TFT criterion in this paper. The proposed algorithms are hybridized with a local search procedure, denoted by *LocalSearch*() based on swap and insertion neighborhood structures. The main purpose of the hybridization stems from the fact that DABC and DDE carry out the global search by exploration of the search space, whereas local search is responsible for intensifying the search on the local minima. Therefore, the balance in global and local searches has been effectively achieved. Through an experimental analysis, it is shown that the performance of the proposed algorithms is as competitive as the recent three best performing algorithms in terms of solution quality and CPU usage time. Ultimately, 44 out of the 90 best-known solutions recently provided by the EDA, tsGLS, and hGLS algorithms are further improved by the proposed algorithms with short-term search. For long-term search, the new best-known solutions are reported for Taillard's benchmark suite [35].

The remaining paper is organized as follows. Section 2 introduces the DABC; and Section 3 presents the hDDE algorithm. The details of the local search algorithms developed for the PFSP with TFT criterion are provided in Section 4. Section 5 discusses the computational results over benchmark problems. Finally, Section 6 comprises the concluding remarks.

2. Discrete artificial bee colony algorithm

Inspired by the intelligent foraging behaviors of honeybee swarms, Karaboga proposed the artificial bee colony (ABC) algorithm that implemented a new swarm intelligence based optimizer [15–19]. It classifies foraging artificial bees into three groups, namely, *employed bees*, *onlookers*, and *scouts*. An *employed bee* is responsible for flying to and making