Computers & Operations Research 77 (2017) 111–126

Contents lists available at ScienceDirect

Computers & Operations Research

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

Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion

M. Fatih Tasgetiren [a,*], Damla Kizilay [a], Quan-Ke Pan [b], P.N. Suganthan [c]

[a] *Industrial Engineering Department, Yasar University, Izmir, Turkey*
[b] *Huazhong University of Science and Technology, Wuhan, PR China*
[c] *School of Electronics, Nanyang Technological University, Singapore*

ARTICLE INFO

Article history:
Received 6 July 2015
Received in revised form
1 April 2016
Accepted 2 July 2016
Available online 12 July 2016

Keywords:
Meta-heuristics
Blocking flowshop
Iterated greedy algorithm
Variable neighborhood search
Constructive heuristics

ABSTRACT

Recently, iterated greedy algorithms have been successfully applied to solve a variety of combinatorial optimization problems. This paper presents iterated greedy algorithms for solving the blocking flowshop scheduling problem (BFSP) with the makespan criterion. Main contributions of this paper can be summed up as follows. We propose a constructive heuristic to generate an initial solution. The constructive heuristic generates better results than those currently in the literature. We employ and adopt well-known speed-up methods from the literature for both insertion and swap neighborhood structures. In addition, an iteration jumping probability is proposed to change the neighborhood structure from insertion neighborhood to swap neighborhood. Generally speaking, the insertion neighborhood is much more effective than the swap neighborhood for the permutation flowshop scheduling problems. Instead of considering the use of these neighborhood structures in a framework of the variable neighborhood search algorithm, two powerful local search algorithms are designed in such a way that the search process is guided by an iteration jumping probability determining which neighborhood structure will be employed. By doing so, it is shown that some additional enhancements can be achieved by employing the swap neighborhood structure with a speed-up method without jeopardizing the effectiveness of the insertion neighborhood. We also show that the performance of the iterated greedy algorithm significantly depends on the speed-up method employed. The parameters of the proposed iterated greedy algorithms are tuned through a design of experiments on randomly generated benchmark instances. Extensive computational results on Taillard's well-known benchmark suite show that the iterated greedy algorithms with speed-up methods are equivalent or superior to the best performing algorithms from the literature. Ultimately, 85 out of 120 problem instances are further improved with substantial margins.

1. Introduction

Permutation flowshop scheduling (PFSP) has been extensively studied in the literature and has important applications in manufacturing and service systems [1–4]. In the traditional permutation flowshop scheduling problem, n jobs are processed on m machines in the same permutation, and work-in-process inventory is allowed since there are infinite buffer capacities amongst consecutive machines. In other words, jobs are allowed to be waiting for their next operations. However, if there does not exist any storage capacity amongst machines, the traditional PFSP become a blocking flowshop scheduling problem (BFSF) [5]. In this case, a job cannot leave the machine unless the next machine is free. The BFSP has important applications in industry which can be found in [5,6]. A comprehensive review on flowshop scheduling with blocking and no-wait constraint can be found in Hall and Sriskandarajah [7].

In this paper, we consider the BFSP with the makespan criterion. With the notation of Graham et al. [8], this problem is denoted as $F_m/blocking/C_{max}$. About the computational complexity of the problem, even though Gilmore and Gomory's algorithm [9] can solve it to optimality when the number of machines is two (m=2), it is proven to be NP-Hard by Hall and Sriskandarajah [7] when m>2. For this reason, heuristic approaches should be considered for scheduling a large number of jobs, which is commonly required in industry.

Due to the NP-Hard nature of the problem, constructive heuristic and meta-heuristic algorithms have been developed for solving the BFSP. Regarding the constructive heuristics, a profile fitting (PF) heuristic is developed by McCormick et al. [10] in order

* Corresponding author.
E-mail address: fatih.tasgetiren@yasar.edu.tr (M.F. Tasgetiren).

http://dx.doi.org/10.1016/j.cor.2016.07.002
0305-0548/© 2016 Elsevier Ltd. All rights reserved.

to solve sequencing problems with blocking to minimize cycle time. The PF heuristic basically tries to sequence the next job having minimum idle and blocking times on machines. Another approach is presented by Leisten [11] in order to deal with flowshop scheduling problems with finite and unlimited buffers to maximize the buffer usages and to minimize the machine blocking times. However, the heuristic developed in [11] was not able to yield better solutions than the NEH heuristic, which is initially proposed in [12] to solve the traditional PFSP. Based on the makespan properties defined by Ronconi and Armentano [13], Ronconi [6] proposed a note on constructive heuristics and developed three constructive heuristics, called minmax (MM), MM combined with NEH (MME), and PF combined with NEH (PFE), respectively for the BFSP with the makespan criterion. In the light of computational analysis, it was demonstrated that the MME and PFE heuristics were superior to the NEH algorithm in problems with up to 500 jobs and 20 machines. Abadi et al. [14] proposed an improvement heuristic for minimizing cycle time in blocking flowshop which can be employed for the BFSP with the makespan criterion. Ronconi and Henriques [15] tried to minimize the total tardiness in a flowshop with blocking and presented some constructive heuristics providing promising results for the problems considered. In addition to above, some effective heuristics based on PF approach were proposed in Pan and Wang [16]. These PF based heuristics are also inspired from LR heuristic proposed by Liu and Revees [17] for the PFSP with total flowtime criterion. They combined PF heuristic with the partial NEH implementation and called the heuristics as PF_NEH(x), WPF_NEH(x) and PW_NEH(x), where x is the number of sequences generated by considering the first x number of jobs in the initial order of jobs. In order to retain good characteristics of the PF heuristic, the NEH heuristic is applied to only the last $\delta = 20$ jobs. Their PW_NEH(5) heuristic was substantially better than NEH, MME, PFE, WPFE and PWE heuristics from the literature.

Regarding the meta-heuristic algorithms, a genetic algorithm (GA) was presented in Caraffa et al. [18]. Ronconi [19] presented a branch and bound algorithm to report the upper bounds for Taillard's benchmark suite [20]. Grabowski and Pempera [21] presented a fast tabu search. Wang et al. [22] proposed a hybrid genetic algorithm. Liu et al [23] developed a hybrid particle swarm optimization algorithm. Qian et al. [24] developed a differential evolution algorithm. Liang et al. [25] proposed a multi-swarm particle swarm optimization algorithm. Harmony search algorithms were developed by Wang et al. [26,27]. Wang et al. [28] proposed a hybrid discrete differential evolution algorithm (HDDE). Ribas et al. [29] proposed an iterated greedy algorithm (IGA) with excellent results, where all best known solutions were further improved. Wang et al. [30] developed a three phase algorithm. effective heuristics with a local search were also presented in [16]. A revised artificial immune algorithm (RAIS) was presented by Lin and Ying [31], where an IG algorithm was also developed. A memetic algorithm (MA) was presented by Pan et al. [32]. A competitive variable neighborhood search (SVNS) algorithms were proposed by Ribas et al. [33] and all the best known solutions reported in [29] were further improved by the SVNS algorithms.

Main contributions of this paper can be summarized as follows. We propose a constructive heuristic with results, which are superior to those in the current literature. For the PFSP and its variants with the makespan criterion, the reduction methods of computational complexity for the insertion neighborhood were proposed by Grabowski and Pempera [5], Taillard [34], Nowicki and Smutnicki [35], Smutnicki [36], and Nowicki [37]. By inspiring from these reduction techniques, Wang et al. [28] developed a speed-up method to evaluate the whole insertion neighborhood for the BFSP with the makespan criterion. We show in this paper that this speed-up method is extremely effective in solving the BFSP with the makespan criterion. Along with it, we also employ a simple speed-up method from Lia et al. [38], for the swap neighborhood, which is adapted for the BFSP with the makespan criterion. In addition, an iteration jumping probability is proposed to jump from one neighborhood structure to another one; hence resulting in a variable neighborhood search algorithm. Instead of employing these two neighborhood structures in a framework of the variable neighborhood descend (VND) algorithms [39], we present two powerful local search algorithms, where the search process is guided by an iteration jumping probability determining which neighborhood structure will be employed. By means of a jump to the swap neighborhood with a small probability, it is shown that some additional enhancements can be gathered while retaining the effectiveness of the insertion neighborhood. We also show that the performance of the iterated greedy algorithm significantly depends on the speed-up method employed. The parameters of the iterated greedy algorithms are tuned through a design of experiments approach on randomly generated benchmark instances. Extensive computational results on Taillard's well-known benchmark suite show that the proposed iterated greedy algorithms with speed-up methods are equivalent or superior to the best performing algorithms from the literature. Especially, we show that iterated greedy algorithms proposed are substantially and significantly better than SVNS algorithms, which are recently proposed by Ribas et al. [33]. Ultimately, 85 out of 120 problem instances are further improved with substantial margins.

The rest of the paper is organized as follows. In Section 2, the blocking flow shop scheduling problem with speed-up methods is formulated. Section 3 presents the IG algorithms with iteration jumping, VND and SVNS variants. Section 4 presents the design of experiment approach for parameter tuning. Extensive computational results and comparisons are provided in Section 5. Finally, Section 6 gives the concluding remarks.

2. Blocking flow shop scheduling problem

In the blocking flow shop scheduling problem, n jobs form the set $J=\{1, 2, \ldots, n\}$ have to be processed on m machines from the set $M=\{1, 2, \ldots, m\}$ with the same permutation on each machine without any intermediate buffer. Each job j has a sequence of m operations. Each operation $o_{j,k}$ has a processing time denoted as $p_{j,k}$. The setup time is assumed to be included in the processing time. Only one job can be processed on each machine. Since the waiting times are not allowed in the flowshop due to the no intermediate buffers, jobs cannot leave machines after finishing their operations until next machines are free. In other words, if the next machine is busy, then the current job on the machine must be blocked since there is no intermediate buffer amongst machines. The goal is to obtain a permutation which will be applied to each machine and the makespan is to be minimum. Given a job permutation, $\pi=\{\pi_1, \pi_2, \ldots, \pi_n\}$, the departure time $e_{j,k}$ of job π_j on machine k can be computed by following Ronconi [6] as follows:

$$e_{1,0}=0 \tag{1}$$

$$e_{1,k} = e_{1,k-1} + p_{\pi_1,k} \quad k = 1, \ldots, m-1 \tag{2}$$

$$e_{j,0} = e_{j-1,1} \quad j = 2, \ldots, n \tag{3}$$