European Journal of Operational Research 264 (2018) 66–73

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup times

A. Sioud[*], C. Gagné

Département d'informatique et de mathématique, Université du Québec à Chicoutimi, 555 Boulevard de l'Université, Chicoutimi, Québec G7H 2B1, Canada

ARTICLE INFO

Article history:
Received 23 March 2016
Accepted 8 June 2017
Available online 16 June 2017

Keywords:
Scheduling
Migrating birds optimization
Permutation flowshop
Makespan
Sequence dependent setup times

ABSTRACT

This paper presents an enhanced migrating bird optimization (MBO) algorithm and a new heuristic for solving a scheduling problem. The proposed approaches are applied to a permutation flowshop with sequence dependent setup times and the objective of minimizing the makespan. In order to augment the MBOs intensification capacity, an original problem specific heuristic is introduced. An adapted neighborhood, a tabu list, a restart mechanism and an original process for selecting a new leader also improved the MBO's behavior. Using benchmarks from the literature, the resulting enhanced MBO (EMBO) gives state-of-the-art results when compared with other algorithms reference. A statistical analysis of the numerical experiments confirms the relative efficiency and effectiveness of both EMBO and the new heuristic.

1. Introduction

Since the work of Johnson (1954), the flowshop scheduling area has been a very active research field with a great number of papers. A flowshop scheduling problem can be defined by a set $N = \{1, \ldots, n\}$ of n independent jobs that have to be processed on a set $M = \{M_1, \ldots, M_m\}$ of m machines. Sequentially, all jobs are processed on machine M_1, then on machine M_2 and so on until the last machine M_m. Each job has a known and deterministic fixed processing time denoted as p_{ij}, $i \in M$ and $j \in N$. In flowshop researches, the vast majority of papers deal with the regular flowshop problem with the objective of minimizing the maximum completion time commonly called the makespan and denoted C_{max}. Indeed, setup times in general and sequence dependent setup times (SDST) in particular have attracted much less attention, although there are equipment setup times between two different jobs in many industries, including pharmaceutics, metallurgy, electronics, ceramics and automotive manufacturing. Setup times involve operations that have to be performed on machines but that are not part of the processing times, such as repairing, cleaning, adjusting or releasing machines, etc. These setup times may or may not depend on the job sequence. Dudek, Smith, and Panwalkar (1974) reported that 70% of industrial activities include SDSTs. More recently, Conner (2009) reports that in 250 industrial projects, 50% contain SDST, and when these setup times are taken into account, 92% of the order deadlines are met. Production of good schedules often relies on management of these setup times (Allahverdi, Ng, Cheng, & Kovalyov, 2008; Allahverdi & Soroush, 2008; Zhu & Wilhelm, 2006). In this paper, we consider known, deterministic and non-negative SDST denoted by s_{ijk} on machine i, $i \in M$, when processing the job k, $k \in N$ after processing the job j, $j \in N$. Solving a flowshop scheduling problem consists in finding a sequence for processing the jobs on the machines that optimizes some criterion. This yields $n!$ possible sequences on each machine and a total of $(n!)^m$ sequences. When the sequence on the first machine is the same for all the other machines, i.e. the jobs do not overlap, the flowshop is called the permutation flowshop (PFS). In this paper, we present a new problem specific heuristic and an enhanced migrating birds optimization algorithm to solve the PFS problem with SDST and the objective of minimizing the makespan (PFS/SDST-C_{max}), noted as $F/prmu, s_{ijk}/C_{max}$ in accordance with the notation of Graham, Lawler, Lenstra, and Kan (1979). Gupta and Darrow (1986) have shown that minimizing the FS/SDST-C_{max} is $\mathcal{NP}$-hard even when $m = 2$ and setups are present only on the first or second machine. For $m = 1$, the SDST flowshop is known to be a special case of the Traveling Salesman Problem (TSP) that is also well known to be $\mathcal{NP}$-hard. This paper has five sections. Section 2 reviews the literature on the PFS/SDST-C_{max}.

In Section 3, we introduce the new heuristic based on setup times and the new enhanced migrating birds optimization algorithm, noted as STH and EMBO, respectively. A thorough experimental comparison of algorithms is described in Section 4.

[*] Corresponding author.
E-mail addresses: aymen.sioud@uqac.ca (A. Sioud), caroline.gagne@uqac.ca (C. Gagné).

http://dx.doi.org/10.1016/j.ejor.2017.06.027
0377-2217/© 2017 Elsevier B.V. All rights reserved.

Finally, Section 5 concludes this paper with some remarks for future research.

2. Literature review of the PFS with SDST

There is less research on flowshop with sequence dependent setup times (SDST) than on regular flowshops. Ruiz and Maroto (2005) made an extensive survey of the literature on permutation flowshop (PFS) problem. Later, Pan and Ruiz (2013) surveyed the literature on the PFS minimizing the total flowtime. Neither survey mentioned any research dealing with setup times. Recently, Allahverdi (2015) surveyed scheduling problems with setup/cost times. Only a few of the more 150 papers on flowshop problems dealt with PFS/SDST-C_{max}, although PFS is extensively considered in the literature. Also, many examples in real-life factories can be formulated as a PFS/SDST-C_{max} based problem (Allahverdi, 2015). Gupta and Darrow (1986) solved the two-machine flowshop scheduling problem with setup times that minimizes the makespan, proposing four approximate algorithms. They also used these heuristics in a branch-and-bound algorithm to decrease the computation time. Rios-Mercado and Bard (1999) presented lower and upper bounds in a branch-and-bound algorithm to minimize the makespan in a PFS with SDST. This algorithm has limitations for large instances. Ruiz and Maroto (2006) proposed genetic and memetic algorithms for solving PFS/SDST-C_{max}. They carried out an experimental study comparing their proposals with several methods adapted from the general flowshop problem. The proposed algorithms outperformed the adapted ones. Kaweegitbundit (2011)introduced and compared a genetic and memetic algorithms to solve the PFS/SDST-C_{max}. The memetic algorithm, which embedded the NEH heuristic, shows good results for the new benchmark described in the paper. Ladhari, Msakni, and Allahverdi (2011) proposed a priority rule, several constructive heuristics, local search procedures and a multiple crossover genetic algorithm to minimize the sum of completion times in a two-machine PFS subject to setup times. Schaller (2012) solved a PFS with family setup times, using greedy algorithms and neighborhood searches. The various algorithms presented minimize the total tardiness. Li and Zhang (2012) proposed several adaptive hybrid genetic algorithms to separately minimize makespan and total weighted tardiness. They also integrated several local searches in their adaptive genetic algorithms, but their AHA_3 outperform all the other algorithms including the memetic algorithm of Ruiz and Maroto (2006). Gharbi, Ladhari, Msakni, and Serairi (2013) minimized the makespan, introducing new lower bounds in a branch-and-bound algorithm to solve the two-machine flowshop scheduling problem with sequence independent setup times. Ciavotta, Minella, and Ruiz (2013) introduced a restarted iterated greedy heuristic to solve a bi-objective PFS with SDST minimizing both the makespan and the total weighted tardiness. Shen, Gupta, and Buscher (2014) solved a flowshop batching problem where sequence dependent family setup times are present and the objective is to minimize makespan using a tabu search metaheuristic. Mirabi, Ghomi, and Jolai (2014) proposed a hybrid genetic algorithm to minimize the makespan for the PFS with SDST. The proposed algorithm introduced new genetic operators and a swap based improvement heuristic. The experiments confirmed its efficiency, outperforming other heuristics from literature (Laha & Chakraborty, 2007; Sheibani, 2010; Xiao-Ping, Yue-Xuan, & Cheng, 2004). Vanchipura, Sridharan, and Babu (2014) proposed a variable neighborhood descent (VND) using two different heuristics to construct the initial solutions, in order to solve a regular flowshop with SDST. There are some other related problems that have been dealt with in the literature. Ruiz and Stützle (2007) introduced an iterated greedy heuristic (IGH) to minimize the makespan in a PFS scheduling problem. They used NEH based construction and destruction phase. Pan, Tasgetiren, and Liang (2008) introduced a discrete differential evolution algorithm and an IGH based on the one introduced by Ruiz and Stützle (2007). They showed that their methods perform better for both makespan and total flowtime. Naderi, Zandieh, and Roshanaei (2009) and Naderi, Ruiz, and Zandieh (2010) introduced a simulated annealing and an iterated greedy heuristic, respectively, to solve a variant of the hybrid flexible flowshop with SDST. Pan and Ruiz (2012a) introduced an estimation of distribution algorithm metaheuristic (EDA) for a batch streaming flowshop with setup times under the idling and no-idling cases. The EDA embedded a local search technique based on the NEH heuristic and a speed-up process. Pan and Ruiz (2012b) also proposed an IGH and adapted a genetic algorithm to minimize the total flowtime in a PFS. They showed that the IGH, based on the one introduced by Ruiz and Stützle (2007) outperforms 15 other algorithms, such as the discrete differential evolution algorithm and the IGH of Pan et al. (2008). Dhouib, Teghem, and Loukil (2013) used a mathematical programming formulation and simulated annealing to solve the PFS problem with SDST and time-lags constraints minimizing the number of tardy jobs. Hecker, Stanke, Becker, and Hitzmann (2014) minimized the makespan and the total idle time of machines in a bakery with 26 stages and 40 products, using a genetic algorithm, an ant colony optimization and a random search procedure, separately. This problem is modeled by a hybrid flexible flowshop with SDST. Ziaee (2013) introduced an NEH based heuristic to solve a general flowshop with SDST minimizing the total weighted tardiness. Pan and Dong (2014) proposed a migrating birds optimization algorithm to solve the PFS/SDST problem minimizing the total flowtime. They used a mixed neighborhood and NEH-based initial population and an intensification mechanism. In the same vein, we also found more complex problems. Sioud, Gagné, and Gravel (2014) proposed a genetic algorithm, an ant colony optimization algorithm and a hybrid genetic algorithm. Their hybrid genetic algorithm outperformed all the other algorithms compared, including the IGH of Naderi et al. (2010).

3. Proposed algorithms for the PFS with SDST

We present in this section a new problem specific heuristic noted STH and the enhanced migrating birds algorithm noted EMBO.

3.1. STH: a new heuristic based on setup times for the PFS with SDST

Most heuristics and priority rules found in the literature do not take sequence dependent setup times (SDST) into consideration (Allahverdi et al., 2008; Zhu & Wilhelm, 2006). We introduce here a new heuristic (*STH*) for the PFS with SDST. The *STH* algorithm is illustrated in Algorithm 1. From a complete solution and with a b parameter, the STH works in four steps: (i) choose a block of jobs, (ii) select b insertion points, (iii) insert the chosen block and evaluate the generated solution, and (iv) iterate step (i)–(iii) b times and return the best generated solution. From a complete permutation S we randomly choose a jobs block $\mathcal{B}$ with size in [1,3]. Classical block insertion heuristics (Allahverdi et al., 2008) try to improve the S permutation by inserting the $\mathcal{B}$ jobs block at all the possible positions. In the STH heuristic, block $\mathcal{B}$ is inserted only in the b best positions, where b is a STH' parameter. These positions are selected by minimizing the setup times s_{lm} in Fig. 1, where m represents the first job in the block ($\mathcal{B}[0]$) and l its position. We insert the $\mathcal{B}$ at all the selected positions, thereby generating b solutions, among which the STH returns the best one, i.e., with the best C_{max}. In the special case of a first position insertion, we consider the s_{ml} setup times where l represents the last job of the $\mathcal{B}$ block. This process (choosing a block, selecting insertion points, generating and evaluating new solutions by