A Matheuristic Approach to the Open Shop Scheduling Problem with Sequence-Dependent Setup Times

Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-10 (2022) 2167–2172

A Matheuristic Approach to the Open Shop Scheduling Problem with Sequence-Dependent Setup Times ⋆

Erica Pastore * **Arianna Alfieri** * **Claudio Castiglione** *
Gaia Nicosia ** **Fabio Salassa** *

* *Dipartimento di Ingegneria Gestionale e della Produzione, Politecnico di Torino, Italy (e-mail: {arianna.alfieri, claudio.castiglione, erica.pastore, fabio.salassa}@polito.it).*
** *Dipartimento di Ingegneria, Università degli Studi "Roma Tre", Italy (e-mail: gaia.nicosia@uniroma3.it)*

Abstract: This paper deals with an open shop scheduling problem in which sequence-dependent setup times are present. In open shops there are no restrictions on the processing route of each job, so the decision regards not only the sequencing of jobs on each machine, but also the sequencing of operations (machines) for each job. These type of problems typically arise in application contexts where the order in which the operations are executed is irrelevant. In this work a novel heuristic approach based on mathematical programming, i.e., a *matheuristic*, is developed and its performance is assessed through a computational study on open shop benchmark instances.

Keywords: Scheduling, Open Shop, Sequence-dependent setup times, Matheuristics, Mixed Integer Linear Programming.

1. INTRODUCTION

Open shop, flow shop, job shop, and mixed shop problems are scheduling problems in multi-machine environments, which are widely used for modeling industrial production processes. The most studied layouts in the literature are flow shops and job shops; here, instead, we deal with the open shop layout, which does not have a pre-determined routing for jobs (Pinedo, 2012). Each job is characterized by a set of operations, each operation must be performed by a given machine, but the processing route can be any. Hence, in the pure open shop scheduling problem, two decisions must be taken: (1) the processing route for each job (i.e., the sequence of operations/machines) and (2) the sequence of jobs on each machine.

Open shop problems typically arise in contexts where the order in which operations must be performed is immaterial, but there is the restriction that only one operation at a time can be performed by each machine and on each job. Examples can be found in testing or maintenance/repair environments, quality control facilities, teacher-class assignments, examination and class scheduling in universities, and medical tests in health care facilities. In all these applications, the operations (repair operations, academic activities, medical tests, quality controls, etc.) that each job (students, patients, aircrafts, or machines to be maintained/repaired/tested, etc.) has to complete, can be performed in any order. The only restriction is to avoid overlapping for the same job (e.g., a student cannot take two exams at the same time), and for the same machine (e.g., two exams cannot be held in the same room at the same time). Open shop problems are generally NP-hard, because the lack of precedence relations among job operations leads to a larger solution space and a looser solution space structure. Furthermore, despite their practical relevance, open shop problems have received far less attention in the literature, with respect to flow shop, job shop and single machine problems. In the literature on open shop problems, the most common used performance measure is the makespan minimization, and various specific versions of the problem have been considered. Just to cite a few examples, widely studied problems are with two or three machines (Dong et al., 2013; Koulamas and Kyparisis, 2015; Liaw, 2003, e.g.), multi-processor (Matta and Elmaghraby, 2010; Abdelmaguid, 2020, e.g.), and with special processing time characteristics (Chen et al., 2020, e.g.). Moreover, for the most complex cases, heuristic algorithms are adopted to address the problem (Abreu et al., 2020; Goldansaz et al., 2013; Mejía and Yuraszeck, 2020, e.g.).

Regardless of the characteristics of the problem, in practical settings, a crucial role is played by setup times, i.e., the time needed to switch from the processing of a job to the processing of another (e.g., time to prepare the machine, preheating time, time to change the tool, etc.). Setup times can be sequence-dependent or independent. In the first case, the time spent to prepare the machine to process a job depends both on the job to be processed and on the job immediately preceding it, while in the second it depends only on the job to be processed. Sequence-dependent setups need to be explicitly considered, as the time used for setups changes with different job sequences.

⋆ This work is partially supported by MIUR PRIN Project AHeAD (Efficient Algorithms for HArnessing Networked Data).

10.1016/j.ifacol.2022.10.029

This heavily impacts on machine capacities and then on all the performance measures. Sequence-dependent setup times are mainly (but not exclusively) found in chemical and food sectors, where cleaning operations might depend on both the job to be processed and on the previous one (the more different the jobs, in terms of flavor or chemical composition, the more accurate must be the cleaning).

In this paper, an open shop problem with an arbitrary number of machines and jobs is considered. Sequence-dependent setup times are present, and the total completion time has to be minimized. Although, from a practical point of view, the total completion time is a very relevant performance measure, most of the literature focuses on makespan minimization (Abreu and Nagano, 2022; Bai et al., 2016; Goldansaz et al., 2013; Koulamas and Kyparisis, 2015; Naderi et al., 2010; Sedeño-Noda et al., 2009, e.g.). Usually, the total completion time minimization is addressed through weighted solution approaches (Bai et al., 2017, e.g.) or special cases in terms of processing time (Brucker et al., 1993, e.g.), or focus on basic cases to show asymptotic behavior of some standard rules such as the Shortest Processing Time rule (SPT) (Bräsel et al., 2008; Liaw et al., 2002; Tang and Bai, 2010, e.g.). Although sequence-dependent setup times have been largely addressed in the literature in the context of different scheduling problems (Agnetis et al., 2004; Alfieri and Nicosia, 2014; Engehausen and Lödding, 2022; Shen et al., 2018, e.g.), very few papers explicitly consider them for open shops (Abreu and Nagano, 2022; Naderi et al., 2011; Roshanaei et al., 2010, e.g.). The problem addressed in this paper is the same as in Naderi et al. (2011), for which an electromagnetism-like meta-heuristic, improved by a simulated annealing algorithm, is proposed. The complexity of the problem calls for a heuristic algorithm, as only very small instances can be solved to optimality; however, differently from Naderi et al. (2011), and from most of the heuristic approaches found in the literature, in this paper, a matheuristic rather than a meta-heuristic algorithm is proposed. Matheuristics are optimization algorithms that alternate exact solution phases, mainly dealt with mathematical programming-based techniques, with heuristic phases. In many applications, the heuristic phase is the relaxation of some model constraints, or some integer variable relaxation and/or fixing. In these cases, the mathematical model of the problem is very important and, hence, matheuristics are also called model-based heuristics.

1.1 Our Contribution

In this paper, as already mentioned, a matheuristic algorithm based on a Mixed Integer Linear Programming (MILP) model of the problem is used. Specifically, in the proposed solution approach, the continuous relaxation of a set of variables (the so-called *sequence variables*) and their further rounding is alternated to computing the exact solution of the relaxed/fixed model. Computational tests on a set of benchmark instances from Taillard (1993) show that the proposed matheuristic becomes more and more effective when the number of jobs and machines increases if compared to a pure mathematical programming approach.

The reminder of the paper is organized as follows. Section 2 formally describes the problem and presents the mathematical programming model. Section 3 illustrates in details the proposed matheuristic approach. Numerical results are presented and discussed in Section 4, while some conclusions are drawn in Section 5.

2. PROBLEM DESCRIPTION

The classical open shop scheduling problem can be defined as follows. A set of n jobs must be processed by a set of m machines such that each job visits all the machines exactly once. Differently from other shop scheduling problems, such as flow shop or job shop problems, there are no predetermined routes for the jobs on the machines; so, the processing path (route) of each job must be determined during the scheduling. The open shop problem can then be considered as a generalization of both the flow shop and the job shop. Therefore, in all the open shop problems, two levels of decision must be undertaken, namely the processing route of each job and the sequence of jobs on each machine. Other assumptions, common to most shop scheduling problems, also apply to the open shop scheduling.

Rigorously, the addressed open shop problem with sequence-dependent setup times and total completion time minimization can be defined as follows. [1]

We are given:

- a set $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$ of m machines;
- a set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ of n jobs;
- for each job $J_j \in \mathcal{J}$ a set of m operations $\mathcal{O}(j) = \{O_{j,1}, O_{j,2}, \ldots, O_{j,m}\}$, one for each machine, where operation $O_{j,i}$ has to be processed by machine M_i for $p_{j,i}$ time units;
- for each machine $M_i \in \mathcal{M}$ and for each pair of jobs J_j and J_k a setup time $s_{jk,i}$, which is the time needed for set-up operations that have to be spent on machine M_i when switching from the execution of job J_j to that of job J_k.

We assume, as in most job shop problems, that a feasible schedule is such that (*i*) at any time each machine can process at most one operation; (*ii*) the operations of the same job are totally ordered since each job can be processed by at most one machine at a time; (*iii*) no preemption is allowed. Assuming that all jobs are available at time 0, we want to find *job orders* (orders of operations belonging to the same job) and *machine orders* (orders of operations to be processed on the same machine) so that the total completion time is minimized.

According to Graham's three field notation (Graham et al., 1979), the open shop scheduling problem with sequence-dependent setup times in which the objective is the minimization of the total completion time can be denoted as $O|ST_{sd}|\sum_j C_j$.

In the following, a mixed integer linear programming formulation (which is an adaptation of the one in Naderi et al. (2011)) is presented. The decision variables are the following:

[1] Note that, to help readability, job and machine indices used in variables and parameters subscripts are separated by a comma, e.g., subscript jk,i refers to jobs J_j and J_k and machine M_i.

- $C_{j,i}, C_j \in \mathbb{R}_+$, representing the completion times of operation $O_{j,i}$ and of job J_j, respectively, with $i = 1, 2, \dots, m$ and $j = 1, 2, \dots n$;
- $x_{j,il} \in \{0, 1\}$, encoding whether operation $O_{j,i}$ precedes $O_{j,l}$ for each job $j = 1, 2, \dots n$ and for each pair of operations on machines M_i and M_l with $i, l = 1, 2, \dots m$ with $i \neq l$;
- $y_{i,jk} \in \{0, 1\}$, indicating whether operation $O_{j,i}$ *immediately* precedes $O_{k,i}$ on machine M_i, $i = 1, 2, \dots m$ for each pair of jobs J_j and J_k in $\mathcal{J}$.

Note that in the following MILP [2], in order to better model the precedences between jobs, we consider a dummy job J_0, which has null processing times and that will be scheduled before all other jobs and we let $\bar{\mathcal{J}} = \mathcal{J} \cup \{J_0\}$.

$$\min \sum_{j \in \mathcal{J}} C_j \tag{1}$$

$$s.t. \sum_{\substack{k \in \bar{\mathcal{J}} \\ k \neq j}} y_{i,kj} = 1 \qquad \forall i \in \mathcal{M}, \forall j \in \mathcal{J} \tag{2}$$

$$\sum_{\substack{k \in \mathcal{J} \\ k \neq j}} y_{i,jk} \leq 1 \qquad \forall i \in \mathcal{M}, \forall j \in \mathcal{J} \tag{3}$$

$$\sum_{j \in \mathcal{J}} y_{i,0j} = 1 \qquad \forall i \in \mathcal{M} \tag{4}$$

$$x_{j,il} + x_{j,li} = 1 \qquad \forall i, l \in \mathcal{M}, j \in \mathcal{J} \tag{5}$$

$$y_{i,jk} + y_{i,kj} \leq 1 \qquad \forall i \in \mathcal{M}, \forall j, k \in \mathcal{J} \tag{6}$$

$$C_{k,i} \geq C_{j,i} + p_{k,i} + s_{jk,i} - M(1 - y_{i,jk}) \qquad \forall i \in \mathcal{M}, j, k \in \mathcal{J} \tag{7}$$

$$C_{j,l} \geq C_{j,i} + p_{j,l} + \sum_{k=0, k \neq j}^{n} s_{kj,l} y_{l,kj} - M(1 - x_{j,il}) \qquad \forall j \in \mathcal{J}, \forall i, l \in \mathcal{M} \tag{8}$$

$$C_{j,i} \geq C_{j,l} + p_{j,i} + \sum_{k=0, k \neq j}^{n} s_{kj,i} y_{i,kj} - M \cdot x_{j,il} \qquad \forall j \in \mathcal{J}, \forall i, l \in \mathcal{M} \tag{9}$$

$$C_j \geq C_{j,i} \qquad \forall j \in \mathcal{J}, i \in \mathcal{M} \tag{10}$$

$$C_j \geq 0 \qquad \forall j \in \mathcal{J} \tag{11}$$

$$C_{j,i} \geq 0 \qquad \forall j \in \mathcal{J}, \forall i \in \mathcal{M} \tag{12}$$

$$x_{j,il} \in \{0, 1\} \qquad \forall i, l \in \mathcal{M}, \forall j \in \mathcal{J} \tag{13}$$

$$y_{i,jk} \in \{0, 1\} \qquad \forall i \in \mathcal{M}, \forall j, k \in \mathcal{J} \tag{14}$$

The first constraint set (2) ensures that every job J_k has to be scheduled exactly once on each machine M_i by imposing that each operation $O_{k,i}$ has exactly one operation $O_{j,i}$ preceding it for some $j = 0, 1, \dots n$. Constraints (3) guarantee that each job has at most one succeeding job on each machine. Constraints (4) specify that the dummy job J_0 must be the predecessor of exactly one job on each machine and thus defining the first job to be processed on each machine. Constraints (5) imply that each job J_i is either processed on machine M_i first and then on M_l or vice versa, for any pair of machines. Analogously, constraints (6) avoid cross precedence among jobs on each machine M_i, i.e., if job J_j precedes job J_k, then job J_k cannot precede job J_j on the same machine. Note that, in order to avoid repetitions, in constraints (5) and (6) we consider only ordered pairs of machines and of jobs. More specifically, constraints (5) can be limited to all pairs of machines M_i, M_l such that $i < l$, while constraints (6) can be restricted to pairs of jobs J_j, J_k with $j < k$. Constraints (7) impose that operation $O_{k,i}$ cannot start before the completion of operation $O_{j,i}$ and of the necessary setup if $O_{k,i}$ is processed immediately after $O_{j,i}$ (that is, if $y_{i,jk} = 1$). Constraint sets (8) and (9) are the disjunctive constraints relating each pair of operations of each job J_j (i.e., they state that given a job J_j, either operation $O_{j,i}$ on machine M_i precedes operation $O_{j,l}$ on machine M_l, or vice versa). Constraints (10) compute the completion time of each job J_j. Finally, constraints (11)–(14) define the domain of the decision variables.

[2] With a slight abuse of notation the set $\mathcal{M}$ (resp. the set $\mathcal{J}$) will denote both the set of machines (resp. jobs) and the set of associated indices from 1 to m (resp. from 1 to n).

3. A HEURISTIC APPROACH

Due to the large number of integer variables and to the big-M constraints (7)–(9), the MILP model proposed in the previous section requires too much time to be solved on large instances. For this reason, we propose a heuristic approach, namely a *matheuristic*. It relies on conveniently rounding some of the y variables values obtained in the optimal solution of the continuous relaxation of MILP model (1)-(14) using an iterative scheme. The solution approach is sketched in Algorithm 1, in which we call a commercial solver for the solution of different relaxations and/or special cases of MILP (1)-(14) and we denote as:

- $\mathcal{I}$ a subset of indices for the y variables, i.e. $\mathcal{I} \subseteq \mathcal{M} \times \mathcal{J} \times \mathcal{J}$;
- LP the continuous relaxation of the above MILP;
- $LP(\mathcal{I})$ the continuous relaxation of the above MILP in which all $y_{i,jk}$ variables with indices $i, jk \in \mathcal{I}$ are fixed to value 1;
- $MILP(\mathcal{I}, C)$ the above MILP in which some additional constrains contained in the set C are added, the y variables with indeces in $\mathcal{I}$ are fixed to 1, the remaining y variables are constrained to assume binary values, while the x variables are continuous in interval $[0, 1]$.

In Algorithm 1, t_1 is the maximum iteration time for the main loop, t_2 the time limit to solve $MILP(\mathcal{I}, C)$ at each iteration in the main loop, and t_3 the time limit to compute the final solution.

The proposed solution procedure starts from an initial solution $\hat{y}$, found as the optimal solution of the continuous relaxation of the model. At each iteration, a random percentage of the y variables of the initial solution is fixed to 1 and the new (partially fixed) mathematical programming model is solved to optimality. This way, several distinct solutions may be explored (the algorithm mimics a multi-start solution approach). At the end of each iteration of the main loop, the y variables are set again to assume binary values, while the domain of the x variables is always kept continuous between 0 and 1. At the end of all the iterations (i.e., when the time limit t_1 is reached), to compute the final solution, the y values corresponding to the best solution found are fixed as constraints in the original MILP model, which is then solved with a time limit t_3. The resulting solution is a heuristic solution of the original problem. Note that, during the execution of the algorithm, to ensure that the rounding of the y variables

will lead to a feasible solution, a feasibility check and a restoration procedure, called REPAIR, are performed.

Algorithm 1: Matheuristic for $O|ST_{sd}|\sum_j C_j$

```
/* Initialization                                          */
LB* ← +∞, best solution y* ← ∅, and constraint set C ← ∅;
Solve LP and let ŷ be the solution values for the y variables;
Let I be the set of indices of ŷ variables having value larger than threshold s;
/* Main step: Exploration of candidate solutions          */
while time limit t1 is not reached do
    Randomly pick a probability value p;
    Each variable index in I is assigned to index set I' ⊆ I with probability p;
    Solve LP(I') and let y̌ be the solution values obtained for the y variables;
    Round up to 1 all y̌ variables having value larger than threshold s and add the corresponding indices to I';
    if y̌ is not feasible for job routing on machines then
        Apply REPAIR procedure updating index set I' and constraint set C;
    end
    Solve MILP(I', C) with a time limit of t2 and let LB be the lower bound obtained after time limit is reached corresponding to the best solution found;
    if LB < LB* then
        Update best solution y* and best lower bound LB*;
    end
end
/* Final solution computation                             */
Solve the original MILP model with a time limit of t3 and with the constraints that all y variables having integer values in y* are fixed;
```

A more detailed description of the matheuristic approach sketched in Algorithm 1 is given hereafter.

Initialization. To get an initial solution in terms of y values, the continuous relaxation LP of the original problem (obtained by letting the binary variables y and x assume any value in the interval $[0, 1]$) is solved. The y values found as solution ($\hat{y}$) of the continuous relaxation are used as initial values for the algorithm. As $\hat{y}$ values are continuous, some of them are rounded up to an integer value according to the following: if $\hat{y}_{i,jk} > s$ then $\hat{y}_{i,jk} = 1$. We denote as $\mathcal{I}$ the set of indices of $\hat{y}$ variables having value larger than the threshold s (in our experiments, s has been set to 0.98).

Exploration of candidate solutions. A time limit t_1 is set, and until it is not reached, new solutions in terms of y values are explored iteratively. At each iteration, first a random probability value p is selected and then, the continuous relaxation is solved again, by adding some constraints to fix several y values. Specifically, if in the initial solution $\hat{y}_{i,jk} = 1$ (i.e. if the index of the y variable is in $\mathcal{I}$), then with probability p the constraint $y_{i,jk} = 1$ is added to the model. The new LP, with these constraints, is called $LP(\mathcal{I}')$ (where $\mathcal{I}' \subseteq \mathcal{I}$ is the subset of randomly selected indices). The y variables in the solution of $LP(\mathcal{I}')$ are rounded up to 1 according to the threshold s, and the corresponding solution is called $\check{y}$. This solution has no guarantee to be feasible. Indeed, since some continuous variables in problems LP and $LP(\mathcal{I}')$ are rounded up to 1, sometimes loops can appear when creating the job sequences for each machine. As an example, consider machine M_1 and three jobs. It may happen that the following variables are set to 1 after the rounding: $\check{y}_{1,13} = 1$, $\check{y}_{1,32} = 1$ and $\check{y}_{1,21} = 1$. This means that job J_1 precedes J_3, job J_3 precedes J_2 and job J_1 follows job J_2, thus the sequence of jobs on machine M_1 would be $J_1 \rightarrow J_3 \rightarrow J_2 \rightarrow J_1$. Such sequence is clearly not feasible for the original MILP model. Thus, before using $\check{y}$ values in the integer model, its feasibility is restored through a REPAIR procedure, which eliminates all the loops in each machine sequence. In the illustrative case, the REPAIR procedure will assure that the loop is broken by removing the constraints that impose that $\check{y}_{1,13}$ and $\check{y}_{1,21}$ are equal to 1 (i.e., by removing the indices from $\mathcal{I}'$), and by adding to the current mathematical model (i.e., to the set C) the constraint $y_{1,13} + y_{1,21} \leq 1$. In general, the feasibility check is performed by creating all the job sequences for each machine starting from the solution $\check{y}$. For all such sequences, it checks if any loop is present. If this is the case, the REPAIR procedure removes all the loops as follows. For each detected loop, a job is selected (J_1 in the above example) and the corresponding $\check{y}$ values containing such job as index (in the example, $\check{y}_{1,13}$ and $\check{y}_{1,21}$) are removed from the list of variables that have value fixed to 1, so the index set $\mathcal{I}'$ is updated by removing two indices. A constraint is created and added to the set C to force the sum of these variables to be less than or equal to 1 to avoid that the loop is created again (and thus reducing the solution space).

Lastly, the model $MILP(\mathcal{I}', C)$ is solved within a very short time limit t_2. In $MILP(\mathcal{I}', C)$ we have all the original constraints of the problem, i.e., constraints (2)–(12), plus: (i) all the variables y that correspond to $\check{y}$ variables equal to 1, i.e., those in the set $\mathcal{I}'$, have their value fixed to 1; (ii) the constraints in C, which are the output of the REPAIR procedure to avoid loops; (iii) the remaining y variables constrained to assume binary values; (iv) the x variables constrained to assume continuous values in interval $[0, 1]$. The solution of the $MILP(\mathcal{I}', C)$ model allows to get the best lower bound found by the MILP solver (namely, LB). LB is used as a proxy of the quality of the y solution; indeed, it is the best overall solution that could be achieved with y variables with index in $\mathcal{I}'$ fixed to 1. In case LB is the best value found so far, the corresponding solution, namely y^*, is stored.

Final solution. After reaching the time limit t_1, the final solution is computed by solving the original MILP model in which y values are fixed to the best solution found in the iterations (i.e., the solution with the best LB). A time limit t_3 is set, variables y and x are set to binary, and variables y are fixed to y^*. Obviously, as some y values have been fixed during the solution process, there is no guarantee that the final solution found is optimal for the original MILP problem.

4. EXPERIMENTS

The solution procedure has been tested on various instances to evaluate its efficiency and effectiveness when some characteristics of the input are varied.
The tested instances are based on Taillard open shop benchmarks (Taillard, 1993). Thus, the following combinations of number of jobs and machines values are tested: $(n, m) \in \{(4,4),(5,5),(7,7),(10,10),(15,15),(20,20)\}$. For each combination, 10 instances are available. Since Taillard instances do not include setup times, they are generated as follows. For each (n, m), setup times are generated from $U(1, 499)$ for the first five instances (identified as *low setup time*), and from $U(500, 999)$ for the last five instances (*high setup time*). For each instance, the solution procedure is run 5 times, and the average results are considered. All in all, 300 instances are generated.

The proposed procedure has been implemented in C++, and the mathematical programming models are solved by calling CPLEX solver, version 12.9. Tests were run on a computer having 3.70 GHz Intel (R) *i*7 processor with 32 GB RAM. A one-hour total time limit has been set for the solution procedure, where 10 minutes are devoted to the iteration phase (i.e., $t_1 = 600$ seconds), while 50 minutes to the final solution procedure (i.e., $t_3 = 3000$ seconds). In the iteration phase, the time limit t_2 for computing an integer solution is set to 5 seconds, as it is a sufficient time for any solver to detect a lower bound.

The preliminary computational results are reported in Table 1. The proposed matheuristic procedure (column MATHEUR) is compared to the solution of the original problem through CPLEX solver (column MILP), with a one-hour time limit. In Table 1, each row represents the average solutions calculated over the five instances and over the 5 runs for each instance of the algorithms, for the matheuristic and MILP procedures. Also, the average percentage gap is shown (column %Gap), which is the average over runs and replicates of the percentage gap calculated as: $\%Gap = \frac{Matheur-MILP}{MILP}$. Regarding MILP results, the solutions followed by $*$ are optimal (i.e., CPLEX was able to find the optimal solution in less than one hour).

Table 1. Main results.

(n, m)	Setup time	MATHEUR	MILP	% Gap
(4, 4)	low	4684.6	3759.2*	24.2%
(4, 4)	high	14093	12947.4*	15.5%
(5, 5)	low	8007.2	5611.4*	42.4%
(5, 5)	high	22833.6	19070.2	19.8%
(7, 7)	low	16198.6	10552	54.4%
(7, 7)	high	52365.4	41526.4	26.1%
(10, 10)	low	39159.2	37486.2	5.8%
(10, 10)	high	111248.2	124228.6	-10.4%
(15, 15)	low	91479	122567.4	-25.3%
(15, 15)	high	254821.6	288325.0	-11.6%
(20, 20)	low	173571.6	229103.0	-24.2%
(20, 20)	high	465308.8	519423.0	-10.4%

The results in Table 1 show that the instances with up to 5 jobs and 5 machines can be solved to optimality by CPLEX in less than one hour. More specifically, CPLEX needs less than one second to solve the $(4, 4)$ instances with low setup times, and up to five seconds with high setup times. For the $(5, 5)$ instances, high setup times make CPLEX unable to find the optimal solution in one hour, whereas for low setup times the optimum is found on average in 270 seconds. This shows the impact of the magnitude of setup times on the complexity of the solution procedure. Instead, larger instances cannot be solved to optimality within the time limit. The proposed matheuristic finds worse solutions than MILP for small instances. Indeed, the percentage gap is positive and large for all the instances up to 7 jobs and 7 machines. For these instances, the proposed heuristic procedure never reaches the time limit t_3 for the final solution, as the integer problem with variable rounding takes less than five minutes to find its final solution (which, however, has no guarantee to be optimal). Instead, for larger instances, the solution procedure is able to guarantee negative gaps. Although the proposed procedure is not always more efficient than solving the mathematical model with CPLEX, significant improvements may be noticed when the instances become larger and larger, and hence also more realistic. For the instances equal or larger than (10, 10), the final solution phase of the algorithm reaches the time limit t_3, thus the proposed approach takes in total one hour, as the MILP, to find the solution.

5. CONCLUSIONS

This paper addressed an open shop scheduling problem with sequence-dependent setup times, a problem that has not been widely studied in the literature so far. The objective is to find the schedule of jobs on machines that minimizes the total completion time. A matheuristic approach exploiting a mathematical formulation was developed to solve such problem. The computational results show that for very small numbers of jobs and machines the problem can be solved to optimality with a commercial solver in less than five minutes. When there are more than 7 jobs and 7 machines, instead, the commercial solver is not able to find the optimal solution in one hour computation time. The experiments show that the larger is the problem size (i.e., the larger the number of jobs and machines), the better are the solutions found by the proposed heuristic approach with respect to those found by the solver with the same computational budget.

Although the results are promising, the proposed approach still could be improved. At the current state, the matheuristic randomly fixes part of the initial job sequences on each machine. More effective rules can be used to either generate the initial solution or to fix a part of it during iterations. Also, no control has been taken on the variables related to the machine sequences for each job. Moreover, next research threads could deal with the exploitation of the disjuctive graph representation of the problem. In fact, it is possible to represent the problem solved at each iteration of the proposed algorithm (i.e., the one with fixed sequences of jobs on machines) by a disjunctive graph. Differently from what happens with job shop problems, instead of determining the orientation of the disjunctive arcs on the machines, the decision deals with establishing the sequence of machines for each job. Also, a generalization of the classical disjunctive graph as in Agnetis et al. (2011) could be used to address the problem by means of a combinatorial branch and bound

procedure. In addition, at the current state, the solution approach is driven by the exploration of various solutions in terms of sequence of jobs on machines. Future research could be devoted also to address the opposite direction: instead of fixing the sequences of jobs in each machine, a new perspective could lead to fixing the sequences of machines for each job. By doing so, the problem with fixed job sequences would be a job shop scheduling problem, with sequence-dependent setup times.

REFERENCES

Abdelmaguid, T.F. (2020). Scatter search with path relinking for multiprocessor open shop scheduling. *Computers & Industrial Engineering*, 141, 106292.

Abreu, L.R., Cunha, J.O., Prata, B.A., and Framinan, J.M. (2020). A genetic algorithm for scheduling open shops with sequence-dependent setup times. *Computers & Operations Research*, 113, 104793.

Abreu, L.R. and Nagano, M.S. (2022). A new hybridization of adaptive large neighborhood search with constraint programming for open shop scheduling with sequence-dependent setup times. *Computers & Industrial Engineering*, 168, 108128.

Agnetis, A., Flamini, M., Nicosia, G., and Pacifici, A. (2011). A job-shop problem with one additional resource type. *Journal of Scheduling*, 14(3), 225–237.

Agnetis, A., Alfieri, A., and Nicosia, G. (2004). A heuristic approach to batching and scheduling a single machine to minimize setup costs. *Computers & Industrial Engineering*, 46(4), 793–802.

Alfieri, A. and Nicosia, G. (2014). Sequencing a batching flexible cell to minimise set-up costs. *International Journal of Production Research*, 52(8), 2461–2476.

Bai, D., Zhang, Z.H., and Zhang, Q. (2016). Flexible open shop scheduling problem to minimize makespan. *Computers & Operations Research*, 67, 207–215.

Bai, D., Zhang, Z., Zhang, Q., and Tang, M. (2017). Open shop scheduling problem to minimize total weighted completion time. *Engineering Optimization*, 49(1), 98–112.

Bräsel, H., Herms, A., Mörig, M., Tautenhahn, T., Tusch, J., and Werner, F. (2008). Heuristic constructive algorithms for open shop scheduling to minimize mean flow time. *European Journal of Operational Research*, 189(3), 856–870.

Brucker, P., Jurisch, B., and Jurisch, M. (1993). Open shop problems with unit time operations. *Zeitschrift für Operations Research*, 37(1), 59–73.

Chen, Y., Goebel, R., Lin, G., Su, B., and Zhang, A. (2020). Open-shop scheduling for unit jobs under precedence constraints. *Theoretical Computer Science*, 803, 144–151.

Dong, J., Zhang, A., Chen, Y., and Yang, Q. (2013). Approximation algorithms for two-machine open shop scheduling with batch and delivery coordination. *Theoretical Computer Science*, 491, 94–102.

Engehausen, F. and Lödding, H. (2022). Managing sequence-dependent setup times - the target conflict between output rate, WIP and fluctuating throughput times for setup cycles. *Production Planning & Control*, 33(1), 84–100.

Goldansaz, S.M., Jolai, F., and Anaraki, A.H.Z. (2013). A hybrid imperialist competitive algorithm for minimizing makespan in a multi-processor open shop. *Applied Mathematical Modelling*, 37(23), 9603–9616.

Graham, R.L., Lawler, E.L., Lenstra, J.K., and Kan, A.R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics*, volume 5, 287–326. Elsevier.

Koulamas, C. and Kyparisis, G.J. (2015). The three-machine proportionate open shop and mixed shop minimum makespan problems. *European Journal of Operational Research*, 243(1), 70–74.

Liaw, C.F. (2003). An efficient tabu search approach for the two-machine preemptive open shop scheduling problem. *Computers & Operations Research*, 30(14), 2081–2095.

Liaw, C.F., Cheng, C.Y., and Chen, M. (2002). The total completion time open shop scheduling problem with a given sequence of jobs on one machine. *Computers & Operations Research*, 29(9), 1251–1266.

Matta, M.E. and Elmaghraby, S.E. (2010). Polynomial time algorithms for two special classes of the proportionate multiprocessor open shop. *European Journal of Operational Research*, 201(3), 720–728.

Mejía, G. and Yuraszeck, F. (2020). A self-tuning variable neighborhood search algorithm and an effective decoding scheme for open shop scheduling problems with travel/setup times. *European Journal of Operational Research*, 285(2), 484–496.

Naderi, B., Ghomi, S.F., Aminnayeri, M., and Zandieh, M. (2010). A contribution and new heuristics for open shop scheduling. *Computers & Operations Research*, 37(1), 213–221.

Naderi, B., Ghomi, S.F., Aminnayeri, M., and Zandieh, M. (2011). Modeling and scheduling open shops with sequence-dependent setup times to minimize total completion time. *The International Journal of Advanced Manufacturing Technology*, 53(5-8), 751–760.

Pinedo, M.L. (2012). Open shops (deterministic). In *Scheduling: Theory, Algorithms, and Systems*, 221–243. Springer US, Boston, MA.

Roshanaei, V., Seyyed Esfehani, M., and Zandieh, M. (2010). Integrating non-preemptive open shops scheduling with sequence-dependent setup times using advanced metaheuristics. *Expert Systems with Applications*, 37(1), 259–266.

Sedeño-Noda, A., de Pablo, D.A.L., and González-Martín, C. (2009). A network flow-based method to solve performance cost and makespan open-shop scheduling problems with time-windows. *European Journal of Operational Research*, 196(1), 140–154.

Shen, L., Dauzère-Pérès, S., and Neufeld, J.S. (2018). Solving the flexible job shop scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, 265(2), 503–516.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2), 278–285.

Tang, L. and Bai, D. (2010). A new heuristic for open shop total completion time problem. *Applied Mathematical Modelling*, 34(3), 735–743.