European Journal of Operational Research 311 (2023) 845–866

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Exact and heuristic solution approaches for energy-efficient identical parallel machine scheduling with time-of-use costs

Mauro Gaggero[a], Massimo Paolucci[b], Roberto Ronco[b,*]

[a] Institute of Marine Engineering, National Research Council of Italy, Via De Marini 6, Genoa, I-16149, Italy
[b] Department of Informatics, Bioengineering, Robotics and Systems Engineering, University of Genoa, Viale Causa 13, Genoa I-16145, Italy

ARTICLE INFO

Article history:
Received 12 February 2022
Accepted 31 May 2023
Available online 2 June 2023

Keywords:
Scheduling
Time-of-use prices
Fast algorithms
Mixed-integer programming
Dynamic programming

ABSTRACT

Nowadays, energy-efficient scheduling has assumed a key role in ensuring the sustainability of manufacturing processes. In this context, we focus on the bi-objective problem of scheduling a set of jobs on identical parallel machines to simultaneously minimize the maximum completion time and the total energy consumption over a time horizon partitioned into a set of discrete slots. The energy costs are determined by a time-of-use pricing scheme, which plays a crucial role in regulating energy demand and flattening its peaks. First, we uncover a symmetry-breaking property that characterizes the structure of the solution space of the problem. As a consequence, we provide a novel, compact mixed-integer linear programming formulation at the core of an efficient exact solution algorithm. A thorough experimental campaign shows that the use of the novel mathematical programming formulation enables the solution of larger-scale instances and entails a reduction in the computational times as compared to the formulation already available in the literature. Furthermore, we propose a new heuristic that improves the state-of-the-art in terms of required computational effort and quality of solutions. Such a heuristic outperforms the existing heuristics for the problem and is also capable of speeding up the exact solution algorithm when used for its initialization. Finally, we introduce a novel dynamic programming algorithm that is able to compute the optimal timing of the jobs scheduled on each machine to further improve the performance of the new heuristic.

1. Introduction

In the last years, the compelling challenges in environmental sustainability have led to the development of a new paradigm for manufacturing that allows planning the production while restraining the resulting energetic expenditure. Such a paradigm, called energy-efficient scheduling or green scheduling (Gao et al., 2020), enables an energy-conscious approach to job scheduling in production. Among the demand-response strategies to regulate energy generation, provisioning, and consumption, time-of-use (TOU) pricing schemes have proven useful to flatten the peaks in customers' demand to limit the resulting environmental pollution (Wang & Li, 2015). In the literature, one of the most considered energy-aware goals is the minimization of the total energy cost (TEC), that is the sum of the costs associated with the time slots where some job is processed.

In many applications, the need of minimizing energy costs may conflict with other typical goals of scheduling problems, such as the minimization of the makespan (Jia et al., 2017; Jiang & Wang, 2020) or the minimization of the total weighted tardiness (Fang & Lin, 2013; Zhang & Chiong, 2016). In this paper, we consider a bi-objective scheduling problem with TOU costs, where N independent, non-preemptable jobs with no release time have to be scheduled on M identical, parallel, and single-server machines over a time horizon of K time slots to simultaneously minimize the makespan and the TEC. From now on, we refer to such a problem as *Bi-objective identical parallel machine scheduling with Time-of-Use costs problem* (BPMSTP).

The BPMSTP was first investigated by Wang et al. (2018), who proposed a constructive heuristic endowed with local search capabilities and presented the first mixed-integer linear programming (MILP) formulation for the problem. Subsequently, Anghinolfi et al. (2021) proposed a faster and more accurate heuristic approach, by enhancing the constructive heuristic of Wang et al. (2018) and exploiting some intuitions on the combinatorics of the problem. To the best of our knowledge, the work of Anghinolfi et al. (2021) currently constitutes the state-of-the-art heuristic for the BPMSTP (Catanzaro et al., 2023). In this manuscript, we build

* Corresponding author.
E-mail addresses: mauro.gaggero@cnr.it (M. Gaggero), massimo.paolucci@unige.it (M. Paolucci), roberto.ronco@edu.unige.it (R. Ronco).

https://doi.org/10.1016/j.ejor.2023.05.040
0377-2217/

upon the results of Anghinolfi et al. (2021) by proposing a novel MILP formulation and a heuristic that achieves higher computational performances. Specifically, the contribution of this paper is threefold. The first contribution concerns a formal description of fundamental concepts regarding the combinatorics of the BPMSTP. In particular, we establish equivalence relations between different solutions by highlighting structural symmetries inherent to the solution space. The existence of such relations allows us to provide a compact MILP formulation for the BPMSTP. The compactness of the novel formulation also enables the development of a fast, exact solution algorithm. Such results constitute an important step toward the discovery of novel combinatorial properties of multi-objective TOU scheduling problems with multiple machines. These problems have been widely investigated in the last decade (see, e.g., Castro et al., 2013; Li et al., 2016; Mitra et al., 2012; Moon et al., 2013; Zeng et al., 2018, and the related discussion in Section 2). The second contribution is the development of a heuristic approach for the BPMSTP that vastly improves upon the one proposed by Anghinolfi et al. (2021) in terms of both required computational effort and quality of solutions, by performing several enhancements to the involved algorithms. Such a heuristic also exploits a novel exact algorithm based on dynamic programming that efficiently determines the optimal timing of jobs in a single-machine schedule to minimize its energy cost. Lastly, the third contribution regards the combination of the first and the second ones to provide an initial solution to the proposed exact algorithm to further reduce the computational effort required to solve the BPMSTP optimally.

Concerning the complexity of the BPMSTP, we remark that the problem of minimizing the TEC within some given deadline is known to be strongly $\mathcal{NP}$-hard, even on a single machine (Chen & Zhang, 2019). As a consequence, the BPMSTP is strongly $\mathcal{NP}$-hard, as already hinted by Wang et al. (2018). Furthermore, Fang et al. (2016) and Chen & Zhang (2019) both considered the problem of minimizing the TEC on a single machine, showing that it can be solved by using exact algorithms with a polynomial and pseudo-polynomial running time when time slots costs satisfy some specific properties. Determining the existence of similar algorithms for the BPMSTP still constitutes an open problem.

The rest of this paper is organized as follows. In Section 2, we report an overview of the literature on scheduling with energy-efficiency criteria, by specifically focusing on scheduling with TOU costs. In Section 3, we discuss the problem statement and the existing state-of-the-art mathematical formulation. In Section 4, we analyze the combinatorial properties of the BPMSTP, and we present the novel MILP formulation. In Section 5, we present the exact algorithm for the BPMSTP and the novel heuristic approach, and in Section 6 we discuss the numerical results obtained on an extensive experimental campaign. We draw conclusions in Section 7, by also prospecting possible future developments of our work.

2. Literature review

Energy-efficient scheduling has become a relevant topic in production planning due to the growing interest of the manufacturing industry in environmentally-sustainable production over the last years (see, e.g., Gahm et al., 2016; Giret et al., 2015). The shift toward sustainable manufacturing is the result of the worldwide growth of customers' demands as well as more severe standards for environmental pollution, such as CO_2 emissions and extensive land use. Haapala et al. (2013) were among the first authors to stress the importance of energy efficiency as a part of production scheduling in modern manufacturing. In the last decade, several works in scheduling have pursued sustainable production as their key goal. Among the most recent contributions, we mention Karimi et al. (2021) in the context of additive manufacturing, aimed at minimizing energy cost in response to time-varying electricity prices and demand charges, Zhou et al. (2020) for single-machine batch processing with dynamic job arrival times, Barak et al. (2021) for resource-constrained flexible manufacturing systems, and Zeng et al. (2022) for multi-objective flow shop scheduling. In more detail, while Karimi et al. (2021) proposed a mathematical model for the problem at hand, Barak et al. (2021); Zhou et al. (2020) and Zeng et al. (2022) employed multi-objective metaheuristics. In particular, Zeng et al. (2022) presented an implementation of the non-dominated sorting genetic algorithm (Deb et al., 2002), a well-known evolutionary algorithm in the literature of multi-objective optimization (Absalom et al., 2021). According to Gao et al. (2020), evolutionary algorithms are widely used to solve scheduling problems that deal with several objectives and constraints (see, among others, Faria et al., 2019; Lei et al., 2018; Tang et al., 2016). Manufacturing is not the only field that was able to benefit from energy-aware scheduling practices. Among others, we mention scheduling in datacenters (Caviglione et al., 2021), real-time systems (Bambagini et al., 2016), and distributed systems (Agrawal & Rao, 2014).

The literature on energy-efficient scheduling with TOU pricing schemes can be classified according to the number of optimization objectives, the type of processing environment, and the considered solution approaches. We refer the reader to (Catanzaro et al., 2023) for a comprehensive survey of the problems, models, and algorithms in the field. Hereinafter, we review some of the most recent and relevant works characterized by (i) the optimization of a single objective on parallel machines, (ii) the use of multi-objective models and metaheuristics for multiple machines, and (iii) the investigation of multi-objective approaches for some compelling application cases.

First, concerning single-objective parallel machine problems, Ding et al. (2016) presented a time-indexed MILP model for the job scheduling problem of minimizing the TEC on parallel unrelated machines. The authors also proposed a further approach based on a Dantzig–Wolfe decomposition algorithm for the problem. Cheng et al. (2018) expanded the work of Ding et al. (2016) by providing an improved MILP formulation that uses fewer decision variables and constraints. Such a formulation was able to outperform the one proposed by Ding et al. (2016) on a large set of instances. Besides the TEC, another objective function that is often considered in the literature on scheduling with TOU costs is the linear combination of the TEC with the makespan, which enables the simultaneous minimization of both. Usually, the former has a unitary coefficient, while the latter is weighted by a constant that represents a penalty, such as maintenance and overtime costs. This objective function is also compelling in practical applications, as it is able to capture productivity requirements with environmental awareness. Moon et al. (2013) proposed a time-indexed MILP formulation for the problem of minimizing such an objective on unrelated parallel machines. This formulation was later improved by Cheng et al. (2019) through a set of strengthening inequalities. Pei et al. (2021) generalized the problem faced by Moon et al. (2013) and Cheng et al. (2019) by considering the minimization of the linear combination of the makespan and the TEC on unrelated parallel machines, where both objectives are weighted by positive penalty factors. The authors proposed a non-linear mathematical programming formulation for the problem and then developed an approximation algorithm based on a single-objective relaxation.

In the following, we discuss multi-objective models and metaheuristics. Cheng et al. (2017) focused on the simultaneous minimization of the makespan and the TEC for a single-machine batch scheduling problem. In this problem, the involved machine requires an additional amount of power to switch between idle and operational states. Qian et al. (2020) also considered batch

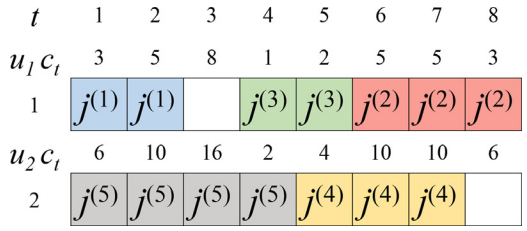

Fig. 1. Example of a schedule of five jobs (named $j^{(1)}$, $j^{(2)}$, $j^{(3)}$, $j^{(4)}$, and $j^{(5)}$) on two machines (denoted by 1 and 2), with energy consumption rates $u_1 = 1$ and $u_2 = 2$. The $K = 8$ time slots are displayed as squares below the corresponding energy cost on the two machines.

scheduling. In particular, the authors investigated the problem of minimizing the makespan and the TEC on uniform batch machines and proposed a multi-objective evolutionary algorithm exploiting adaptive clustering that extracts information on the solution space to ensure diversity in the populations of solutions. Jiang & Wang (2020) addressed a flexible job shop scheduling problem that requires the minimization of both the makespan and the TEC. The authors presented a MILP model and a multi-objective evolutionary algorithm based on decomposition as a solution approach. Sin & Chung (2020) took into account preventive maintenance in a single-machine scheduling problem with the objective of simultaneously minimizing the TEC and machine unavailability. Similarly to Jiang & Wang (2020), the work of Sin & Chung (2020) proposed a MILP model together with a hybrid multi-objective genetic algorithm in order to solve large instances of the problem. Finally, Zeng et al. (2018) focused on a bi-objective scheduling problem on uniform parallel machines, which requires minimizing the TEC and the number of used machines. The latter objective is also significant for applications, where higher machine uptime conflicts with maintenance shifts and increases the overall power consumption. The authors developed an iterative search framework based on an insertion algorithm for the single-objective problem that consists in minimizing the TEC with a fixed number of machines.

Finally, we consider application cases. We observe that TOU pricing schemes can be interpreted as a possible way to implement the general concept of demand side management, which consists of either reducing energy consumption or rescheduling and shifting energy demand to off-peak hours (see, e.g., Golmohamadi, 2022; Panda et al., 2022 and the references therein). More specifically, Mitra et al. (2012) provided a MILP formulation for optimal operational production planning for power-intensive processes in continuous manufacturing, using non-dispatchable demand response programs based on a discrete-time representation. Castro et al. (2013) presented resource-task network MILP formulations of a steel plant, by investigating the impact of fluctuating energy prices on the scheduling of operations that can be obtained through the participation in price- and incentive-based industrial demand side management programs. Subsequently, Castro et al. (2020) presented another MILP formulation for optimal scheduling under TOU electricity pricing to model processing tasks with variable electrode mass depletion and replacement tasks that regenerate the mass. Among other studies on practical applications available in the literature on TOU pricing schemes, Forghani et al. (2021) investigated the interaction among TOU electricity prices, production scheduling, and preventive maintenance of continuous slurry ball mills by proposing a mixed-integer energy-cost-aware hierarchical formulation modeling approach. Furthermore, Sharma et al. (2015) focused on a flexible flow shop scheduling problem with speed-scaling machines that require the minimization of the carbon footprint, which is affected by the time-varying availability of renewables, as well as the optimization of the TEC under TOU costs. Li et al. (2016) proposed heuristic approaches for parallel machine scheduling problems in green manufacturing, with the goal of minimizing the makespan or the total completion time, subject to proper constraints on the value of the cost. Finally, Rocholl et al. (2020) investigated a bi-objective parallel batch machine scheduling problem based on the fabrication of semiconductor wafers. In particular, the authors proposed three different heuristics based on a genetic algorithm that computes the start times of the batches to minimize energy consumption. The proposed algorithms were enhanced with a local search to further improve the solutions computed by the heuristics.

3. Problem statement and previous formulation

In this section, we first formally describe the problem considered in this paper, i.e., the BPMSTP. Subsequently, we report the formulation of the problem provided by Anghinolfi et al. (2021), which currently constitutes the state-of-the-art in the literature.

Let $\mathcal{J} = \{1, \dots, N\}$ be the set of *jobs*, $\mathcal{H} = \{1, \dots, M\}$ the set of identical *machines*, and $\mathcal{T} = \{1, \dots, K\}$ the set of available *time slots*. Jobs are non-preemptable and are characterized by an integer *processing time* $p_j \le K$, $j \in \mathcal{J}$, corresponding to an integer number of distinct time slots. Machines are endowed with an *energy consumption rate*, which is denoted by $u_h > 0$ for the generic machine $h \in \mathcal{H}$. Moreover, a non-negative cost $c_t \ge 0$, $t \in \mathcal{T}$, is associated with each time slot. The processing of a job $j \in \mathcal{J}$ during a subset $\mathcal{T}_j \subseteq \mathcal{T}$ of p_j consecutive time slots on machine $h \in \mathcal{H}$ corresponds to the *assignment* of job j to $\mathcal{T}_j$ on machine h. In this case, job j is said to be *scheduled* in the time slots in $\mathcal{T}_j$ on machine h. If no job is processed by machine $h \in \mathcal{H}$ in the time slot $t \in \mathcal{T}$, we say that t is *free* on h. Then, we define a *schedule*

$$\mathcal{S} = \left\{ (j, h_j, \mathcal{T}_j) : h_j \in \mathcal{H}, \mathcal{T}_j \subseteq \mathcal{T}, \forall j \in \mathcal{J} \right\} \tag{1}$$

as the set of the assignments of the jobs in $\mathcal{J}$ such that each job $j \in \mathcal{J}$ is scheduled on one and only one machine $h_j \in \mathcal{H}$, and at most a single job in $\mathcal{J}$ is assigned to each time slot in $\mathcal{T}$ on each machine in $\mathcal{H}$. If $\mathcal{T}_j$ is a set of p_j consecutive time slots for each $j \in \mathcal{J}$, then schedule $\mathcal{S}$ is *feasible*. Fig. 1 sketches an example of a schedule with five jobs on two machines over eight time slots. The *completion time* $C_j(\mathcal{S})$ of a job $j \in \mathcal{J}$ in schedule $\mathcal{S}$ is the largest time slot in $\mathcal{T}_j$, that is, $C_j(\mathcal{S}) = \max_{t \in \mathcal{T}_j} t$, $j \in \mathcal{J}$. Furthermore, the *makespan* $C^{\max}$ of a schedule $\mathcal{S}$ is the largest among the completion times of the jobs in $\mathcal{J}$, i.e.,

$$C^{\max}(\mathcal{S}) = \max\{C_j(\mathcal{S}) : j \in \mathcal{J}\}. \tag{2}$$

Let $h_j \in \mathcal{H}$ be the machine where job $j \in \mathcal{J}$ is processed. The energy cost associated with the processing of job j in $\mathcal{S}$ is $u_{h_j} \sum_{t \in \mathcal{T}_j} c_t$. As a consequence, the TEC of $\mathcal{S}$ is given by

$$E(\mathcal{S}) = \sum_{j \in \mathcal{J}} u_{h_j} \sum_{t \in \mathcal{T}_j} c_t. \tag{3}$$

Then, the BPMSTP consists in finding a feasible schedule $\mathcal{S}$ that simultaneously minimizes (2) and (3). Hereinafter, since $\mathcal{S}$ is a feasible solution to the BPMSTP, we use the expressions "feasible schedule" and "feasible solution" interchangeably. Moreover, we omit the dependence of C_j, $C^{\max}$, and E on $\mathcal{S}$ to avoid burdening the notation. We also refer to the ordered tuple $\mathcal{I} = (\mathcal{J}, \{p_j, j \in \mathcal{J}\}, \mathcal{H}, \{u_h, h \in \mathcal{H}\}, \mathcal{T}, \{c_t, t \in \mathcal{T}\})$ as an *instance* of the BPMSTP.

In the remainder of the section, we describe the MILP formulation of the BPMSTP provided by Anghinolfi et al. (2021), referred to as "Formulation 1". Toward this end, we denote by

$$X_{j,h,t} \in \{0, 1\}, \quad \forall j \in \mathcal{J}, h \in \mathcal{H}, t \in \mathcal{T},$$

a binary decision variable that is equal to 1 if t is the start time slot of job j on machine h, and 0 otherwise. Moreover, we express the makespan in (2) and the TEC in (3) with the decision variables $C^{\max} \ge 0$ and $E \ge 0$, respectively.

Formulation 1.

$$\min C^{\max}, \tag{4}$$

$$\min E, \tag{5}$$

subject to

$$E = \sum_{h\in\mathcal{H}} u_h \sum_{j\in\mathcal{J}} \sum_{t=1}^{K-p_j+1} X_{j,h,t} \left(\sum_{i=t}^{t+p_j-1} c_i \right), \tag{6}$$

$$\sum_{h\in\mathcal{H}} \sum_{t=1}^{K-p_j+1} X_{j,h,t} = 1, \quad \forall j \in \mathcal{J}, \tag{7}$$

$$\sum_{j\in\mathcal{J}} \sum_{i=\max\{1,t-p_j+1\}}^{t} X_{j,h,i} \le 1, \quad \forall h \in \mathcal{H}, t \in \mathcal{T}, \tag{8}$$

$$\sum_{h\in\mathcal{H}} \sum_{t=1}^{K-p_j+1} (t + p_j - 1) X_{j,h,t} \le C^{\max}, \quad \forall j \in \mathcal{J}, \tag{9}$$

$$C^{\max} \le K, \tag{10}$$

$$C^{\max} \ge 0, \quad E \ge 0, \quad X_{j,h,t} \in \{0,1\}, \quad \forall j \in \mathcal{J}, h \in \mathcal{H}, t \in \mathcal{T}. \tag{11}$$

The objectives (4) and (5) minimize the makespan and the TEC, respectively, consistently with definitions (2) and (6). Constraints (7) impose that each job $j \in \mathcal{J}$ starts in a single slot on a single machine. Constraints (8) avoid more than one job being processed in the same time slot on the same machine. The left-hand side of (9) defines the completion time of each job in $\mathcal{J}$, which must not exceed the makespan $C^{\max}$. In turn, the makespan cannot be greater than the number of time slots K owing to (10). Finally, (11) defines the decision variables. Formulation 1 employs $NMK + 2$ decision variables and $2N + MK + 2$ constraints. The former number is due to the NMK variables $X_{j,h,t}$, $j \in \mathcal{J}, h \in \mathcal{H}, t \in \mathcal{T}$, together with $C^{\max}$ and E, while the latter one is due to constraints (6)–(10).

Observe that Formulation 1 exploits a discrete-time representation, i.e., the time horizon is partitioned into a finite set of time slots consistently with the statement of the problem, and the processing of each job starts at the beginning of a single time slot. To the best of our knowledge, the formulations for identical or unrelated parallel machines available in the literature on scheduling with TOU costs always employ such discrete-time representations, in contrast to the continuous-time and sequence-based formulations that are often used in classical scheduling. In fact, time-based representations enable directly expressing the TEC as a linear combination of the TOU costs.

The main drawback of Formulation 1 lies in the number of decision variables, which may become very large as the size of the BPMSTP instances increases. We overcome this limitation in the following section, by presenting a novel formulation that builds upon a combinatorial property to enable a compact representation of the solution space.

4. New mathematical perspectives

The purpose of this section is to describe the novel mathematical advancements in the BPMSTP proposed in this paper. Specifically, we describe a fundamental combinatorial property of the solution space of the BPMSTP in Section 4.1. As a consequence, we are able to provide a new compact formulation for the problem in Section 4.2.

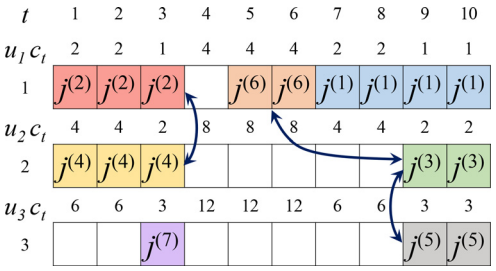

Fig. 2. Example of a schedule of seven jobs (named $j^{(1)}$, $j^{(2)}$, $j^{(3)}$, $j^{(4)}$, $j^{(5)}$, $j^{(6)}$, and $j^{(7)}$) on three machines (denoted by 1, 2, and 3) with energy consumption rates $u_1 = 1$, $u_2 = 2$, and $u_3 = 3$. The arrows showcase possible swaps of jobs resulting in different, equivalent feasible solutions.

4.1. Insights on the combinatorics of the solution space

The solution space of the BPMSTP is characterized by a symmetry-breaking combinatorial property that enables the identification of equivalence classes of solutions for a given BPMSTP instance that are different in structure, but that are identical in terms of makespan and TEC. Such a property is based on the intuition that, given a schedule $\mathcal{S}$, if two jobs scheduled in $\mathcal{S}$ have the same processing time, then exchanging them in $\mathcal{S}$ does not alter the makespan nor the TEC of $\mathcal{S}$.

Hereinafter, we formalize this property. Toward this end, we first define

$$\mathcal{P} := \left\{ d : \exists\, j \in \mathcal{J}, p_j = d \right\}$$

as the set of distinct processing times of the jobs in $\mathcal{J}$. We also define

$$\mathcal{J}_d := \left\{ j : j \in \mathcal{J}, p_j = d \right\}, \quad d \in \mathcal{P}, \tag{12}$$

as the subset of jobs with processing time equal to d. We say that two feasible solutions $\mathcal{S}$ and $\mathcal{S}'$ to the BPMSTP are *equivalent* if they have the same value for $C^{\max}$ and E. Then, the following property holds.

Property 1. *For each feasible solution $\mathcal{S}$ to the BPMSTP, there are at least $\prod_{d\in\mathcal{P}} |\mathcal{J}_d| - 1$ other different, equivalent feasible solutions.*

Proof. Let the schedule $\mathcal{S}$ be given as in (1). Let also $\mathcal{Z} = \{\{h_j, \mathcal{T}_j\}, j \in \mathcal{J}\}$ be the set of all distinct unordered pairs of machines and consecutive time slots such that there is a job $j \in \mathcal{J}$ scheduled in the time slots in $\mathcal{T}_j$ on machine h_j in the schedule $\mathcal{S}$. We observe that $\mathcal{Z}$ can be rewritten as $\bigcup_{d\in\mathcal{P}} \mathcal{Z}_d$, where $\mathcal{Z}_d = \{\{h_j, \mathcal{T}_j\}, j \in \mathcal{J}_d\}$. Since all the jobs in $\mathcal{J}_d$ require the same number d of time slots, all the possible assignments of the jobs in $\mathcal{J}_d$ to the elements of $\mathcal{Z}_d$, for each $d \in \mathcal{P}$, generate schedules that are equivalent to $\mathcal{S}$. As the number of distinct assignments of the jobs in $\mathcal{J}_d$ to $\mathcal{Z}_d$ corresponds to the number $|\mathcal{J}_d|!$ of permutations of the jobs in $\mathcal{J}_d$, the distinct number of assignments of the jobs in $\mathcal{J}$ to $\mathcal{Z}$ is the product of $|\mathcal{J}_d|!$ for each $d \in \mathcal{P}$. The observation that schedule $\mathcal{S}$ is one of such assignments concludes the proof. □

In the following, we illustrate Property 1 through a numerical example.

Example 1. Let us consider Fig. 2, which depicts a possible schedule for jobs $j^{(1)}$, $j^{(2)}$, $j^{(3)}$, $j^{(4)}$, $j^{(5)}$, $j^{(6)}$, and $j^{(7)}$ with processing times $p_{j^{(1)}} = 4$, $p_{j^{(2)}} = p_{j^{(4)}} = 3$, $p_{j^{(3)}} = p_{j^{(5)}} = p_{j^{(6)}} = 2$, and $p_{j^{(7)}} = 1$, on three machines with energy consumption rates $u_1 = 1$, $u_2 = 2$, and $u_3 = 3$, over a time horizon that consists of $K = 10$ time slots. We observe that assigning $j^{(2)}$ in place of $j^{(4)}$, and vice-versa, does not affect the makespan nor the TEC of the schedule, since the two jobs $j^{(2)}$ and $j^{(4)}$ have the same processing time. A similar argument applies to the three jobs $j^{(3)}$, $j^{(5)}$, and $j^{(6)}$.

Indeed, since the number of jobs $|\mathcal{J}_d|$ with the same processing time d is equal to 1, 3, 2, and 1 when d is equal to 1, 2, 3, and 4, respectively, there are at least other different $\prod_{d\in\mathcal{P}}|\mathcal{J}_d|! - 1 = 1!\,2!\,3!\,1! - 1 = 11$ schedules equivalent to the one depicted in the figure, according to Property 1.

4.2. A compact mixed-integer linear programming formulation

We provide a novel formulation that exploits the inherent symmetries of the solution space by building upon the MILP formulation presented by Anghinolfi et al. (2021) and Property 1, described in Section 4.1. First, we denote by

$$b_{d,t} = \sum_{k=t}^{t+d-1} c_k, \quad \forall d \in \mathcal{P}, t = 1, \dots, K-d+1, \tag{13}$$

the cumulative cost associated with the d consecutive time slots $t, t+1, \dots, t+d-1$. As a consequence, any job j with processing time $p_j = d$ assigned to machine h starting at time slot t is characterized by an energy cost equal to $u_h\, b_{d,t}$. Let also

$$Y_{d,h,t} \in \{0,1\}, \quad \forall d \in \mathcal{P}, h \in \mathcal{H}, t \in \mathcal{T},$$

be a binary decision variable that is equal to 1 if t is the first slot of a job with processing time equal to d on machine h, and 0 otherwise.

Formulation 2.

$$\min C^{\max}, \tag{14}$$

$$\min E, \tag{15}$$

subject to

$$E = \sum_{h\in\mathcal{H}} u_h \sum_{d\in\mathcal{P}} \sum_{t=1}^{K-d+1} b_{d,t}\, Y_{d,h,t}, \tag{16}$$

$$\sum_{h\in\mathcal{H}} \sum_{t=1}^{K-d+1} Y_{d,h,t} = |\mathcal{J}_d|, \quad \forall d \in \mathcal{P}, \tag{17}$$

$$\sum_{d\in\mathcal{P}} \sum_{i=\max\{1,t-d+1\}}^{t} Y_{d,h,i} \le 1, \quad \forall h \in \mathcal{H}, t \in \mathcal{T}, \tag{18}$$

$$(t+d-1)\, Y_{d,h,t} \le C^{\max}, \quad \forall d \in \mathcal{P}, h \in \mathcal{H}, t=1, \dots, K-d+1, \tag{19}$$

$$C^{\max} \le K, \tag{20}$$

$$C^{\max} \ge 0, \quad E \ge 0, \quad Y_{d,h,t} \in \{0,1\}, \quad \forall d \in \mathcal{P}, h \in \mathcal{H}, t \in \mathcal{T}. \tag{21}$$

The objectives (14) and (15) minimize the makespan and the TEC, respectively, with the TEC here given by (16). Constraints (17) impose that, for each distinct processing time $d \in \mathcal{J}_d$, exactly $|\mathcal{J}_d|$ jobs with processing time d are assigned to some subsets of slots on the machines. Eq. (18) guarantees that, on each machine, at most a single job is processed in each time slot. The left-hand side of (19) defines the completion time of jobs, which must be less than or equal to the makespan $C^{\max}$. Similarly to Formulation 1, $C^{\max}$ must not exceed the scheduling horizon K, owing to (20). Lastly, (21) defines the decision variables.

Each feasible solution to Formulation 2 defines a class of equivalent schedules. Indeed, Formulation 2 guarantees that, for each $Y_{d,h,t} = 1$, a job with processing time d is non-preemptively scheduled in the slots $t, t+1, \dots, t+d-1$ on machine h, but it does not specify which particular job $j \in \mathcal{J}$, with processing time $p_j = d$, is assigned to such slots on h. Property 1 ensures that, for each solution $\mathcal{S}$ to Formulation 2, there are other $\prod_{d\in\mathcal{P}}|\mathcal{J}_d|! - 1$ different equivalent solutions to $\mathcal{S}$. Since such solutions are all equivalent, each of them has the same representation in terms of the decision variables of Formulation 2.

Algorithm 4.1 generates a possible schedule that belongs to the class of equivalent schedules defined by a solution to Formulation 2. In more detail, such an algorithm first initializes the schedule $\mathcal{S}$ to the empty set at line 1, together with the sets $\mathcal{J}'_d$ for each $d \in \mathcal{P}$ needed for subsequent computations (lines 2–4). Then, for each d, h, and t such that $Y_{d,h,t} = 1$, a job in $\mathcal{J}'_d$ is assigned to d consecutive slots on machine h starting from slot t (lines 5–9). Finally, the computed schedule $\mathcal{S}$ is returned (line 10). At the end of the algorithm, $\mathcal{J}'_d = \emptyset$ for each $d \in \mathcal{P}$, all the jobs in $\mathcal{J}$ are assigned, and there are no slots on the same machine assigned to more than one job.

Algorithm 4.1 Generate-schedule.

Input: The assignment variables $Y_{d,h,t}$, $d \in \mathcal{P}, h \in \mathcal{H}, t \in \mathcal{T}$
Output: A schedule $\mathcal{S}$

1: Let $\mathcal{S} \leftarrow \emptyset$
2: **for** $d \in \mathcal{P}$ **do**
3: Let $\mathcal{J}'_d \leftarrow \mathcal{J}_d$
4: **end for**
5: **for** $(\hat{d}, \hat{h}, \hat{t}) \in \{(d,h,t) : Y_{d,h,t} = 1, d \in \mathcal{P}, h \in \mathcal{H}, t \in \mathcal{T}\}$ **do**
6: Let $j \in \mathcal{J}'_{\hat{d}}$
7: $\mathcal{S} \leftarrow \mathcal{S} \cup (j, \hat{h}, \{\hat{t}, \hat{t}+1, \dots, \hat{t}+\hat{d}-1\})$
8: $\mathcal{J}'_{\hat{d}} \leftarrow \mathcal{J}'_{\hat{d}} \setminus \{j\}$
9: **end for**
10: **return** $\mathcal{S}$

Formulation 2 is characterized by $|\mathcal{P}|MK + 2$ decision variables and $|\mathcal{P}| + MK + |\mathcal{P}|M\sum_{d\in\mathcal{P}}(K-d+1) + 2$ constraints. The former number is due to the $|\mathcal{P}|MK$ variables $Y_{d,h,t}$, $j \in \mathcal{J}, h \in \mathcal{H}, t \in \mathcal{T}$, together with $C^{\max}$ and E, while the latter one is due to constraints (16)–(20). Let us now compare the number of variables needed by Formulation 1 and Formulation 2. The worst case for Formulation 2 occurs when $|\mathcal{P}| = N$, i.e., when all the processing times in $\mathcal{J}$ are distinct. In this case, Formulation 2 has the same number $NMK + 2$ of decision variables characterizing Formulation 1. On the contrary, the most convenient situation for Formulation 2 occurs when the processing times of all the jobs in $\mathcal{J}$ are equal, i.e., when $|\mathcal{P}| = 1$. In this case, Formulation 1 is still characterized by $NMK + 2$ decision variables, while Formulation 2 has only $MK + 2$ variables. Thus, Formulation 2 uses fewer decision variables than Formulation 1, except for the case $|\mathcal{P}| = N$ when the two formulations are equivalent in terms of number of decision variables.

Let us now better characterize the worst case for the number of decision variables of Formulation 2. Toward this end, we observe that a necessary condition for an instance of the BPMSTP to admit at least a feasible solution is that the sum of all the time slots required by the jobs in $\mathcal{J}$ does not exceed the overall number MK of slots available for the scheduling, i.e.,

$$N \le \sum_{j\in\mathcal{J}} p_j \le MK, \tag{22}$$

where the equality $\sum_{j\in\mathcal{J}} p_j = N$ holds when $p_j = 1$ for each $j \in \mathcal{J}$. We formulate the following stronger necessary condition for feasibility by building upon (22).

Proposition 1 (Necessary condition for the existence of a solution)**.** *For a BPMSTP instance that admits at least a feasible solution, the following inequality holds:*

$$|\mathcal{P}| \le \left\lfloor \frac{-1 + \sqrt{1 + 8MK}}{2} \right\rfloor. \tag{23}$$

Proof. First,

$$\sum_{j\in\mathcal{J}} p_j = \sum_{d\in\mathcal{P}} |\mathcal{J}_d| d \geq \sum_{i=1}^{|\mathcal{P}|} i = \frac{|\mathcal{P}|(|\mathcal{P}|+1)}{2} \tag{24}$$

since $|\mathcal{J}_d| \geq 1$ and the elements in $\mathcal{P}$ are pairwise distinct positive integers. By combining (22) with (24), we obtain

$$\frac{|\mathcal{P}|(|\mathcal{P}|+1)}{2} \leq MK,$$

which entails $|\mathcal{P}|^2 + |\mathcal{P}| - 2MK \leq 0$, and therefore

$$0 \leq |\mathcal{P}| \leq \frac{-1+\sqrt{1+8MK}}{2}.$$

□

We observe that, since (22) and (23) only depend on the parameters of the BPMSTP, they are valid for both Formulation 1 and Formulation 2. In more detail, Proposition 1 is useful to identify a larger class of unfeasible solutions with respect to (22), and therefore it enables avoiding solving several instances for Formulation 2 by simply checking the validity of (23) beforehand.

In order to illustrate how Proposition 1 provides a better description of the worst case of Formulation 2 as regards the number of decision variables, we consider an instance with $K = 200$ and $M = 10$ as a simple example. The greatest value of N for the existence of at least a feasible solution corresponds to the case $p_j = 1$ for all $j \in \mathcal{J}$, and it is equal to $MK = 2 \times 10^3$, owing to (22). In this case, the number of decision variables of Formulation 1 is $4 \cdot 10^6 + 2$, whereas it is equal to $2 \times 10^3 + 2$ for Formulation 2 since $|\mathcal{P}| = 1$. Observe that, for such an instance, condition (23) also holds. Instead, if $|\mathcal{P}| = N$, the number of decision variables for Formulation 1 and Formulation 2 is the same. In particular, according to Proposition 1, a necessary condition for feasibility is $N \leq \lfloor(-1+\sqrt{16001})/2\rfloor = 62$. Hence, in order for the considered instance to be possibly feasible, the number of the variables has to be no greater than $1.24 \times 10^5 + 2$. The necessary condition (22) would instead provide the higher upper bound $M^2K^2 + 2 = 4 \times 10^6 + 2$.

To complete the comparison of Formulation 1 and Formulation 2, we also have to take into account the number and nature of the sets of constraints. However, as we highlight in Section 5.1, a discussion of such constraints is not relevant in the framework of the developed exact solution algorithm. In Section 6, we also report the significantly lower computational effort required to solve Formulation 2 with respect to Formulation 1 in all the considered experimental tests.

We conclude this section by observing that, since all the optimal solutions of the BPMSTP are equivalent from a theoretical standpoint, practitioners may be interested in evaluating all of them and then selecting the most suitable one according to their specific needs. Thus, instead of focusing on finding a single solution $\mathcal{S}$, in the following sections we develop exact and heuristic approaches to compute all the different optimal solutions.

5. Solution approaches

In this section, we describe the exact algorithm and the novel heuristic for the BPMSTP proposed in this paper. Specifically, the main goals of the section are the following:

(a) present the proposed exact solution algorithm based on the ϵ-constraint method and Formulation 2 discussed in Section 4.2;
(b) summarize the current state-of-the-art heuristic algorithm for the BPMSTP presented in Anghinolfi et al. (2021), i.e., Split-greedy scheduler with exchange search (SGS-ES), which is used as a reference to assess the effectiveness of the novel heuristic proposed in this paper;
(c) present the novel heuristic approach, i.e., Enhanced heuristic scheduler (EHS), that improves SGS-ES. We also describe all the "building blocks" that characterize EHS.

Before entering into the details of the different solution approaches, we provide some fundamental concepts in multi-objective combinatorial optimization (Branke et al., 2008; Deb, 2001). For the purposes of this paper, we restrict our attention to the bi-objective case of the BPMSTP. We refer to the feasibility region of the BPMSTP as $\mathcal{X}$. Moreover, let $\mathcal{S}$ and $\mathcal{S}'$ be two distinct solutions in $\mathcal{X}$. We say that $\mathcal{S}$ *dominates* $\mathcal{S}'$ if either $C_{\max}(\mathcal{S}) \leq C_{\max}(\mathcal{S}')$ and $E(\mathcal{S}) < E(\mathcal{S}')$, or $C_{\max}(\mathcal{S}) < C_{\max}(\mathcal{S}')$ and $E(\mathcal{S}) \leq E(\mathcal{S}')$; in particular, $\mathcal{S}$ *strictly dominates* $\mathcal{S}'$ if both $C_{\max}(\mathcal{S}) < C_{\max}(\mathcal{S}')$ and $E(\mathcal{S}) < E(\mathcal{S}')$ hold, otherwise $\mathcal{S}$ *weakly dominates* $\mathcal{S}'$. Furthermore, given a subset $\mathcal{O} \subseteq \mathcal{X}$, the set of *non-dominated* solutions in $\mathcal{O}$ exactly contains each and every solution in $\mathcal{O}$ that is not dominated by another solution in $\mathcal{O}$ itself. A solution $\mathcal{S}$ is *Pareto-optimal*, or *Pareto-efficient*, if no other solution in $\mathcal{X}$ dominates $\mathcal{S}$. Specifically, $\mathcal{S}$ is *strictly (weakly) Pareto-optimal* if there is no other solution in $\mathcal{X}$ that weakly (strictly) dominates it. Finally, the *Pareto front* is the set of points in the space of the objectives associated with the solutions in the set of Pareto-optimal solutions, also called *Pareto-optimal set*.

The remainder of this section is structured as follows. First, we describe the exact algorithm in Section 5.1 (goal (a)). Then, we recall the heuristic presented by Anghinolfi et al. (2021) in Section 5.2 (goal (b)). Lastly, we introduce the novel heuristic for the BPMSTP in Section 5.3 (goal (c)), which builds upon the algorithmic ideas presented in Section 5.2 to increase the computational efficiency while improving the quality of the computed solutions at the same time.

5.1. The exact algorithm

The proposed exact algorithm for the BPMSTP relies on the combination of MILP and the ϵ-constraint method for multi-objective optimization, first introduced by Haimes et al. (1971) and further discussed by Chankong & Haimes (2008). Specifically, the exact algorithm iteratively exploits either Formulation 1 or Formulation 2 to compute the set of Pareto-optimal solutions for a given BPMSTP instance $\mathcal{I}$. Without loss of generality, in this subsection, we describe the algorithm by only referring to Formulation 2.

The basic idea of the ϵ-constraint method is to minimize (or maximize) one of the objectives while the other ones are constrained to be lower (or greater) than fixed values. For the considered instance $\mathcal{I}$ of the BPMSTP, the exact algorithm first sets an upper bound on the makespan and then minimizes the TEC. The algorithm iterates over the previous two steps and progressively reduces the upper bound until an unfeasible solution is found. In this way, the algorithm is able to find all the points of the two-dimensional Pareto front of $\mathcal{I}$.

First, we observe that, for a BPMSTP instance $\mathcal{I}$, each Pareto-optimal solution $\mathcal{S}^\star$ of $\mathcal{I}$ corresponds to a non-dominated point $(C^{\max}(\mathcal{S}^\star),\ E(\mathcal{S}^\star))$ in the optimal Pareto front. In particular, there are at most $K - \underline{K}(\mathcal{I}) + 1$ points in the optimal Pareto front, where

$$\underline{K}(\mathcal{I}) = \max\left\{\left\lfloor \sum_{j\in\mathcal{J}} p_j/M \right\rfloor, \max_{j\in\mathcal{J}}\{p_j\}\right\}. \tag{25}$$

Indeed, since the processing times p_j, $j \in \mathcal{J}$, are integer numbers, $C^{\max}(\mathcal{S}^\star)$ is an integer that ranges between the lower bound $\underline{K}(\mathcal{I})$ and the upper bound K. However, we observe that $\underline{K}(\mathcal{I})$ given by (25) is not a tight lower bound for all the instances of the BPMSTP.

Example 2. Let us consider a BPMSTP instance with a set of $N = 4$ jobs, denoted by $j^{(1)}$, $j^{(2)}$, $j^{(3)}$, and $j^{(4)}$, with processing times 2, 9, 9, and 10, respectively, to be scheduled on $M = 3$ machines with $u_1 = u_2 = u_3 = 1$, and a number $K = 11$ of time slots. The lower bound for $C^{\max}$ given by (25) is equal to 10, but there is no feasible solution with such a makespan. In fact, jobs $j^{(2)}$, $j^{(3)}$, and $j^{(4)}$ have to be scheduled on three different machines, without leaving two consecutive free time slots for $j^{(1)}$.

We now define, for a given $\hat{K}$ such that $1 \leq \hat{K} \leq K$, a *downsized instance* of the BPMSTP as

$$\mathcal{D}(\hat{K}) = \big(\mathcal{J}, \{p_j, j \in \mathcal{J}\}, \mathcal{H}, \{u_h, h \in \mathcal{H}\}, \{1, \dots, \hat{K}\}, \{c_t, t \in \{1, \dots, \hat{K}\}\}\big). \tag{26}$$

A downsized instance (26) considers a subset of slots $\{1, 2, \dots, \hat{K}\} \subseteq \mathcal{T}$ instead of the whole set of time slots $\mathcal{T}$. Then, we define the *reduced formulation* of the BPMSTP as the optimization of (15) subject to constraints (16)–(18) and (21). In other words, the reduced formulation only requires the minimization of the TEC without considering constraints (19) and (20) that are related to the makespan.

Algorithm 5.1 reports the pseudo-code of the proposed exact solution algorithm for the BPMSTP based on the aforementioned ideas. It takes a BPMSTP instance $\mathcal{I}$ as input and returns the set of the Pareto-optimal solutions for $\mathcal{I}$ as output. The algorithm first initializes the solution set $\mathcal{O}$ and the parameter $\hat{K}$ at line 1. The latter is used in the downsized instances within the subsequent loop. Then, Algorithm 5.1 repeats lines 2–10 until either $\hat{K}$ is lower than the lower bound $\underline{K}(\mathcal{I})$ or an unfeasible solution is obtained before reaching $\underline{K}(\mathcal{I})$. In more detail, the reduced formulation associated with $\mathcal{D}(\hat{K})$ is solved at line 3. Then, if no feasible solution exists, the loop is stopped (line 5). Otherwise, Algorithm 4.1 is called to obtain an optimal schedule $\mathcal{S}^\star$ for $\mathcal{D}(\hat{K})$ (line 7). Afterward, $\mathcal{S}^\star$ is added to $\mathcal{O}$ (line 8). The number of slots $\hat{K}$ for the next iteration is updated as $C^{\max}(\mathcal{S}^\star) - 1$ at line 9. In fact, we observe that any solution $\mathcal{S}' \neq \mathcal{S}^\star$ to $\mathcal{D}(\hat{K})$ such that $C^{\max}(\mathcal{S}^\star) \leq C^{\max}(\mathcal{S}') \leq \hat{K}$ is either equivalent to or weakly dominated by $\mathcal{S}^\star$. Otherwise, $\mathcal{S}'$ would be the solution computed at line 3, as it would achieve a better TEC than $\mathcal{S}^\star$. Then, Algorithm 5.1 computes the set $\mathcal{F}$ of strictly Pareto-optimal solutions for $\mathcal{I}$ by identifying the non-dominated solutions in $\mathcal{O}$ (line 11) and excluding the weakly Pareto-optimal solutions. Finally, Algorithm 5.1 returns the set of Pareto-optimal solutions $\mathcal{F}$ (line 12).

Algorithm 5.1 Exact algorithm for the BPMSTP.

Input: A BPMSTP instance $\mathcal{I}$
Output: The set $\mathcal{F}$ of Pareto-optimal solutions for $\mathcal{I}$

1: Let $\mathcal{O} \leftarrow \emptyset$ and $\hat{K} \leftarrow K$
2: **while** $\hat{K} \geq \underline{K}(\mathcal{I})$ **do**
3: Solve the reduced formulation of $\mathcal{D}(\hat{K})$ with MILP
4: **if** no feasible solution exists **then**
5: **break**
6: **end if**
7: Let $\mathcal{S}^\star$ be the schedule computed with Algorithm 4.1 from the optimal solution to $\mathcal{D}(\hat{K})$
8: Update $\mathcal{O} \leftarrow \mathcal{O} \cup \{\mathcal{S}^\star\}$
9: $\hat{K} \leftarrow C^{\max}(\mathcal{S}^\star) - 1$
10: **end while**
11: Let $\mathcal{F}$ be the set of non-dominated solutions in $\mathcal{O}$
12: **return** $\mathcal{F}$

The computational efficiency of solving the reduced formulation of $\mathcal{D}(\hat{K})$ with MILP at line 3 in Algorithm 5.1 can be enhanced by providing an initial feasible solution to the MILP solver using a given heuristic. Toward this end, we propose the use of the heuristic schemes described later on in Section 5.3 to perform initialization. The computational advantages of such a choice are investigated in Section 6.

5.2. Split-greedy heuristic and exchange search

In this subsection, we summarize the *Split-greedy heuristic* (SGH) and *Exchange search* (ES) introduced by Anghinolfi et al. (2021) to solve the BPMSTP by describing the concepts at the foundation of the two algorithms. Toward this end, for a given BPMSTP instance $\mathcal{I}$, we first denote a *location* as a pair $l = (h, \mathcal{A})$, where $h \in \mathcal{H}$ and $\mathcal{A}$ is a subset of consecutive slots in $\mathcal{T}$. In addition, l is a *free location* for job j if $|\mathcal{A}| = p_j$, i.e., the number of slots is equal to the processing time of the job, and the slots in $\mathcal{A}$ are free, i.e., no job is assigned. Instead, l is a *split-location* for job j if $\mathcal{A}$ is a set of p_j slots such that there is at least a pair of slots in $\mathcal{A}$ that are not consecutive, and the following condition holds: for each pair of slots $t, t' \in \mathcal{A}$, $t \neq t'$, either t and t' are consecutive, or t and t' are not consecutive and, for each slot $t'' \in \mathcal{T}$ such that $t < t'' < t'$, there is some job in $\mathcal{J} \setminus \{j\}$ assigned to t''. If all the slots in $\mathcal{A}$ are free, then l is a *free split-location*. Finally, a *split-schedule* is a preemptive schedule where at least one job is assigned to a split-location.

The core idea of SGH is to greedily assign the jobs in $\mathcal{J}$ to free locations or free split-locations with the smallest-cost. If the resulting schedule is a split-schedule, then it is converted into an equivalent feasible one. We recall that two schedules are equivalent if they have the same makespan and TEC. The pseudo-code of SGH is reported in Algorithm 5.2. Formally, the algorithm takes a BPMSTP instance $\mathcal{D}(\hat{K})$ as input, with $\hat{K}$ such that $1 \leq \hat{K} \leq K$, and returns a schedule $\mathcal{S}$ as output. If $\mathcal{S}$ is empty, then either no solution exists for $\mathcal{D}(\hat{K})$, or SGH is not able to compute one. In fact, determining whether a feasible schedule exists within a given makespan is already an $\mathcal{NP}$-complete problem (Garey & Johnson, 1978). First, SGH initializes $\mathcal{S}$ as an empty schedule (line 1) and $\mathcal{S}_h$ as an empty set for each $h \in \mathcal{H}$ (line 2). Then, it iterates over each $d \in \mathcal{P}$ in non-increasing order (line 3), according to the well-known longest processing time first (LPT) rule (Pinedo, 2016). For a fixed d, SGH builds a list $L_{d,h}$ of the free locations and free split-locations on machine h for any job with processing time d (line 4). Afterward, the algorithm iterates over each $j \in \mathcal{J}_d$ (line 5). If there

Algorithm 5.2 Split-greedy heuristic (SGH).

Input: A downsized BPMSTP instance $\mathcal{D}(\hat{K})$ as in (26)
Output: A schedule $\mathcal{S}$ for $\mathcal{D}(\hat{K})$

1: Let $\mathcal{S}$ be an empty schedule
2: Let $\mathcal{S}_h \leftarrow \emptyset$, $h \in \mathcal{H}$
3: **for each** $d \in \mathcal{P}$ in non-increasing order **do**
4: Let $L_{d,h}$, $h \in \mathcal{H}$, be the lists of the smallest-cost, free locations and free split-locations on h for $j : p_j = d$
5: **for each** $j \in \mathcal{J}_d$ **do**
6: **if** $L_{d,h} = \emptyset, \forall h \in \mathcal{H}$ **then return** $\mathcal{S}$
7: Select a location $\hat{l} = (\hat{h}, \hat{\mathcal{A}})$ from the smallest-cost locations in $\bigcup_{h \in \mathcal{H}} \{l \in L_{d,h}\}$ randomly
8: Assign job j to $\hat{l}$ by updating $\mathcal{S}_{\hat{h}}$ as $\mathcal{S}_{\hat{h}} \leftarrow \mathcal{S}_{\hat{h}} \cup \{(j, \hat{h}, \hat{\mathcal{A}})\}$
9: Update list $L_{d,\hat{h}}$ by removing the locations affected by the assignment of j
10: **end for**
11: **end for**
12: Let $\mathcal{S} \leftarrow \bigcup_{h \in \mathcal{H}} \mathcal{S}_h$
13: **if** $\mathcal{S}$ is a split-schedule **then** convert it into an equivalent feasible schedule
14: **return** $\mathcal{S}$

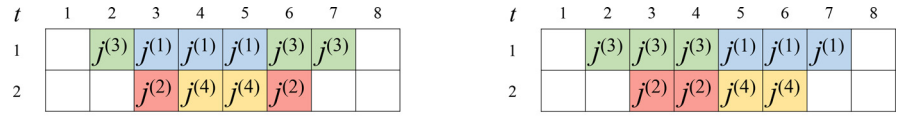

Fig. 3. Example of conversion of a split-schedule (a) into a feasible schedule (b). The two jobs $j^{(3)}$ and $j^{(2)}$ in (a) are assigned to split-locations on machines 1 and 2, respectively. Instead, all the jobs in (b) are feasibly scheduled.

are no free locations for j in $L_{d,h}$, $h \in \mathcal{H}$, then SGH cannot compute a feasible schedule for $\mathcal{D}(\hat{K})$, and it consequently returns an empty schedule (line 6). Otherwise, at each iteration, SGH randomly selects one of the smallest-cost locations $\hat{l} = (\hat{h}, \hat{\mathcal{A}})$ from the locations in the lists $L_{d,h}$, $h \in \mathcal{H}$ (line 8). Then, j is assigned to $\hat{l}$ (line 8), and $L_{d,\hat{h}}$ is updated by removing locations that are not free after the assignment of j, and by adding the new split-locations that may have arisen from the assignment of j (line 9). Subsequently, SGH updates $\mathcal{S}$ as the union of the single-machine schedules $\mathcal{S}_h$, $h \in \mathcal{H}$ (line 12). Finally, if $\mathcal{S}$ is a split-schedule, SGH converts it into an equivalent feasible schedule (line 13). Specifically, the sequence of jobs on each machine is preserved in the converted schedule, and each job starts as soon as possible, but not earlier than its start time in the original split-schedule. Fig. 3 reports an example of such a conversion. Eventually, SGH returns the computed feasible schedule $\mathcal{S}$ at line 14.

ES is a local search algorithm that takes a feasible schedule $\mathcal{S}$ for a BPMSTP instance $\mathcal{D}(\hat{K})$ as input and attempts to improve the TEC without worsening the makespan. The improving moves performed by ES are based on the notion of *exchangeable period sequence* (EPS). An EPS is an ordered pair $(\mathcal{E}, h)$, where $\mathcal{E} \subseteq \mathcal{T}$ is a set of consecutive time slots on a machine $h \in \mathcal{H}$ such that, if a job j is assigned to a time slot in $\mathcal{E}$ on h, then j is scheduled on h, and the time slots where it is processed are in $\mathcal{E}$. In particular, an EPS-J is an EPS that only contains slots assigned to a single job. Instead, an EPS-I is an EPS containing at least an idle slot. We generally refer to an EPS-J and an EPS-I by using the notation $(\mathcal{E}^{\mathrm{J}}, h^{\mathrm{J}})$ and $(\mathcal{E}^{\mathrm{I}}, h^{\mathrm{I}})$, respectively, where $h^{\mathrm{J}} \in \mathcal{H}$ is the machine associated with the EPS-J and $h^{\mathrm{I}} \in \mathcal{H}$ is the one associated with the EPS-I.

For a given schedule $\mathcal{S}$ and an EPS $(\mathcal{E}, h)$, we denote the set of the job assignments in the subset $\mathcal{E} \subseteq \mathcal{T}$ of time slots on machine $h \in \mathcal{H}$ in the schedule $\mathcal{S}$ as $\mathcal{S}_{(\mathcal{E},h)}$. Formally, $\mathcal{S}_{(\mathcal{E},h)}$ is the set of the assignments $(j, h_j, \mathcal{T}_j) \in \mathcal{S}$ such that $h_j = h$ and $\mathcal{T}_j \subseteq \mathcal{E}$. Then, let $(\mathcal{E}', h')$ be another EPS such that $|\mathcal{E}| = |\mathcal{E}'|$. An *EPS swap* is a procedure that reassigns the jobs in $\mathcal{S}_{(\mathcal{E},h)}$ to a subset of slots of $\mathcal{E}'$ on machine h', and the jobs in $\mathcal{S}_{(\mathcal{E}',h')}$ to a subset of slots of $\mathcal{E}$ on h, without changing the relative assignments of the jobs. Specifically, if job j is assigned to the i-th slot of $\mathcal{E}$ on machine h before the EPS swap, then j is assigned to the i-th slot of $\mathcal{E}$' on machine h' after the swap, and vice-versa. The assignments of the jobs to the slots of an EPS $\mathcal{E}$ on a machine h can be changed by using an *EPS rearrangement*, which is a procedure that reschedules the jobs in $\mathcal{S}_{(\mathcal{E},h)}$ in $\mathcal{E}$ on machine h with the goal of reducing the TEC. An *EPS move* combines an EPS swap with an EPS rearrangement. In particular, an EPS move involving two EPSs $(\mathcal{E}, h)$ and $(\mathcal{E}', h')$ such that $|\mathcal{E}| = |\mathcal{E}'|$, respectively, first applies an EPS swap of them, and then an EPS rearrangement of both, separately.

The pseudo-code of ES is reported in Algorithm 5.3. For a given input schedule $\mathcal{S}$, the core idea of ES is to perform all the EPS moves for $\mathcal{S}$ that improve the TEC of $\mathcal{S}$, without worsening its makespan. Specifically, for each $d \in \mathcal{P}$ taken in non-increasing order, and for each EPS-J $(\mathcal{E}^{\mathrm{J}}, h^{\mathrm{J}})$ with a number of time slots $|\mathcal{E}^{\mathrm{J}}| = d$, ES considers every EPS-I $(\mathcal{E}^{\mathrm{I}}, h^{\mathrm{I}})$ with $|\mathcal{E}^{\mathrm{J}}| = |\mathcal{E}^{\mathrm{I}}|$, until it performs an improving EPS move that involves $\mathcal{E}^{\mathrm{J}}$ and $\mathcal{E}^{\mathrm{I}}$. Afterward, ES proceeds with the next EPS-J (lines 6–10). At the end of the iterations (line 15), ES stops if it did not find an improving EPS move for each $d \in \mathcal{J}$. Otherwise, it starts over with the iterations to search for another improving EPS move.

Algorithm 5.3 Exchange search (ES).

Input: A feasible schedule $\mathcal{S}$ for a downsized BPMSTP instance $\mathcal{D}(\hat{K})$ as in (26)

Output: A feasible schedule $\mathcal{S}'$ for $\mathcal{D}(\hat{K})$, with $C^{\max}(\mathcal{S}') \leq C^{\max}(\mathcal{S})$ and $E(\mathcal{S}') \leq E(\mathcal{S})$

1: **repeat**
2: Let $\iota \leftarrow$ false
3: **for** $d \in \mathcal{P}$ in non-increasing order **do**
4: **for** each EPS-J $(\mathcal{E}^{\mathrm{J}}, h^{\mathrm{J}})$ in $\mathcal{S}$ such that $|\mathcal{E}^{\mathrm{J}}| = d$ **do**
5: **for** each EPS-I $(\mathcal{E}^{\mathrm{I}}, h^{\mathrm{I}})$ in $\mathcal{S}$ such that $|\mathcal{E}^{\mathrm{I}}| = d$ **do**
6: Let $\mathcal{S}'$ be the schedule resulting from the EPS move involving $(\mathcal{E}^{\mathrm{J}}, h^{\mathrm{J}})$ and $(\mathcal{E}^{\mathrm{I}}, h^{\mathrm{I}})$ in $\mathcal{S}$
7: **if** $E(\mathcal{S}') < E(\mathcal{S})$ **then**
8: Let $\mathcal{S} \leftarrow \mathcal{S}'$
9: $\iota \leftarrow$ true
10: **break**
11: **end if**
12: **end for**
13: **end for**
14: **end for**
15: **until** ι is false
16: $\mathcal{S}' \leftarrow \mathcal{S}$
17: **return** $\mathcal{S}'$

Example 3. Fig. 4 provides an example of EPS move involving the EPS-J $\mathcal{E}^{\mathrm{J}} = \{1, 2, 3, 4, 5\}$ on machine 1, associated with job $j^{(1)}$, and the EPS-I $\mathcal{E}^{\mathrm{I}} = \{4, 5, 6, 7, 8\}$ on machine 2, including jobs $j^{(4)}$ and $j^{(5)}$ and an idle slot (see Fig. 4(a)). The cost associated with $\mathcal{E}^{\mathrm{J}}$ and $\mathcal{E}^{\mathrm{I}}$ in Fig. 4(a) is 52. Fig. 4(b) shows the result of the EPS swap of $\mathcal{E}^{\mathrm{J}}$ and $\mathcal{E}^{\mathrm{I}}$. After the EPS swap, the cost associated with $\mathcal{E}^{\mathrm{J}}$ and $\mathcal{E}^{\mathrm{I}}$ increases to 56. The EPS rearrangement of $\mathcal{E}^{\mathrm{J}}$ involving jobs $j^{(4)}$ and $j^{(5)}$ in Fig. 4(b) yields the schedule in Fig. 4(c). In such a schedule, the cost associated with $j^{(1)}$, $j^{(4)}$, and $j^{(5)}$ is equal to 49, i.e., it is reduced as compared to the original cost before the move (equal to 52).

Lastly, we describe the *Split-greedy scheduler* (SGS) heuristic, which is used to compute a set of non-dominated solutions for an instance $\mathcal{I}$ of the BPMSTP by using SGH. Similarly to the exact algorithm introduced in Section 5.1, SGS exploits the ϵ-constraint method for multi-objective optimization. Specifically, it first initializes a set $\mathcal{F}$ of heuristic solutions to an empty set. Then, it iterates over $\hat{K}$ from the initial value K to the lower bound $\underline{K}(\mathcal{I})$. At each iteration, SGS solves the BPMSTP instance $\mathcal{D}(\hat{K})$ with SGH instead of using a MILP solver as in line 3 of Algorithm 5.1. If there is no feasible solution for $\mathcal{D}(\hat{K})$, SGS returns an empty set. Otherwise, the set $\mathcal{F}$ is updated by adding the solution to $\mathcal{D}(\hat{K})$, and then, differently from line 9 of Algorithm 5.1, $\hat{K}$ is decreased by 1. At the end of iterations, SGS returns the set of non-dominated solutions in $\mathcal{F}$.

SGS may be combined with ES to improve the computed solutions. *Split-greedy scheduler with exchange search* (SGS-ES) is the algorithm proposed in Anghinolfi et al. (2021) as the result of such

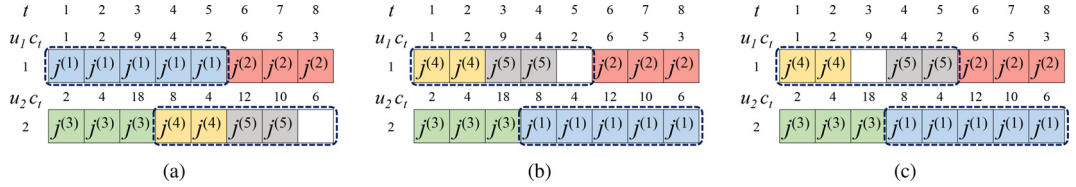

Fig. 4. Example of an EPS move: starting schedule (a); schedule after the EPS swap (b); final schedule after the EPS rearrangements (c). The schedule involves five jobs (named $j^{(1)}$, $j^{(2)}$, $j^{(3)}$, $j^{(4)}$, and $j^{(5)}$) on two machines (denoted by 1 and 2) with energy consumption rate $u_1 = 1$ and $u_2 = 2$.

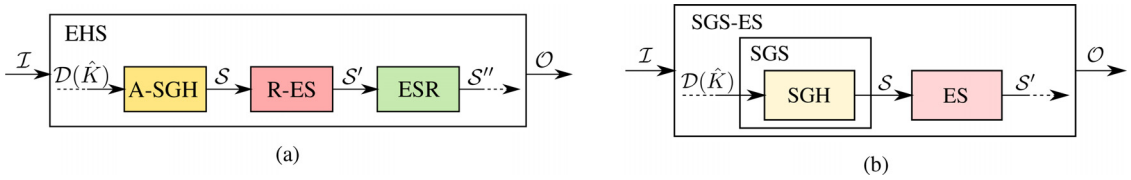

Fig. 5. Sketch of the structure of EHS (a), proposed in this paper as the novel heuristic for the BPMSTP, and SGS-ES (b), introduced by Anghinolfi et al. (2021). Figures (a) and (b) highlight the algorithmic flow that characterizes the different components of EHS (i.e., A-SGH, R-ES, and ESR) and SGS-ES (i.e., SGH, and ES), respectively. Both algorithms take a BPMSTP instance $\mathcal{I}$ as input and return the set of non-dominated solutions in $\mathcal{O}$ as output. The inner part of the bounding boxes depicts the flow for a single iteration of the algorithms.

a combination. In more detail, SGS-ES differs from SGS since it improves the solution computed by SGH by using ES before adding such a solution to $\mathcal{F}$.

5.3. The novel algorithms

In this subsection we introduce the novel heuristic proposed in this paper for the BPMSTP, called *Enhanced heuristic scheduler* (EHS). Such a heuristic builds upon SGH and ES, and also exploits a novel exact algorithm based on dynamic programming that separately optimizes the cost associated with the schedule on each machine. Section 5.3.1 describes *Split-greedy heuristic with assignment history* (A-SGH), which exploits SGH as a subroutine to build a feasible schedule. Section 5.3.2 presents *Exchange search with rescheduling* (R-ES), which constitutes a local search that improves over ES to provide better computational performances. Section 5.3.3 introduces the novel exact algorithm based on dynamic programming called *Exact single-machine rescheduler* (ESR) that, given an input single-machine schedule, efficiently computes the minimum-cost schedule that preserves the original processing sequence. Such an algorithm constitutes an important component of EHS, as it enables to further reduce the cost of the schedule on each machine. Section 5.3.4 finally describes EHS. As showcased in Section 6, EHS is able to obtain high-quality solutions with a low computational burden, thus outperforming the state-of-the-art heuristics for the BPSMTP. Fig. 5 displays a graphical comparison of the structure of EHS (Fig. 5(a)) and SGS-ES (Fig. 5(b)), along with the algorithmic flow characterizing their components.

5.3.1. Split-greedy heuristic with assignment history

We first observe that, as pointed out in Section 5.2, both the exact algorithm and SGS-ES solve a sequence of downsized instances for distinct numbers $\hat{K}$ of time slots. As a result, for a given BPMSTP instance $\mathcal{I}$ and a positive number of available time slots $\hat{K} < K$, the optimal solutions of the two instances $\mathcal{D}(\hat{K})$ and $\mathcal{D}(\hat{K}+1)$ are generally unrelated. However, the heuristic solutions to $\mathcal{D}(\hat{K})$ and $\mathcal{D}(\hat{K}+1)$ generated by SGS-ES share a similar structure. Indeed, many jobs in a solution to $\mathcal{D}(\hat{K}+1)$ are intuitively expected to have the same assignment in a solution to $\mathcal{D}(\hat{K})$, as the two instances only differ for the last time slot $\hat{K}+1$, which is not available in $\mathcal{D}(\hat{K})$. Formally, let $\mathcal{S}'$ be a solution to $\mathcal{D}(\hat{K}+1)$. Then, the assignments in $\mathcal{S}'$ involving jobs whose last slot is no greater than $\hat{K}$ can be exploited to compute a solution $\mathcal{S}$ to $\mathcal{D}(\hat{K})$. Assignments involving slot $\hat{K}+1$ are instead unfeasible for $\mathcal{S}$, as its makespan has to be less than or equal to $\hat{K}$. Hence, the jobs involved in such assignments have to be rescheduled in $\mathcal{S}$.

These observations lie at the core of Split-greedy Heuristic with Assignment History (A-SGH), which solves $\mathcal{D}(\hat{K})$ by exploiting a subset of the job assignments in $\mathcal{S}'$ and employs SGH as a subroutine to schedule the jobs whose assignment in $\mathcal{S}'$ is unfeasible for $\mathcal{D}(\hat{K})$. Specifically, the fundamental idea underlying A-SGH is to start from an initially empty schedule $\mathcal{S}$, and perform the following steps for each $d \in \mathcal{P}$ considered in non-increasing order, where $s_j(\mathcal{S}')$ is the first slot of j in the schedule $\mathcal{S}'$:

(i) update $\mathcal{S}$ with the set of job assignments $(j, h_j, \mathcal{T}_j) \in \mathcal{S}'$, $j \in \mathcal{J}_d$, such that $s_j(\mathcal{S}') + p_j - 1 \leq \hat{K}$ and $\mathcal{S} \cup (j, h_j, \mathcal{T}_j)$ is feasible;
(ii) schedule all the jobs that could not be scheduled in $\mathcal{S}$ during step (i) through SGH.

Step (i) updates $\mathcal{S}$ with all the assignments in $\mathcal{S}'$ for the jobs in $\mathcal{J}_d$ that are feasible in $\mathcal{S}$. Then, step (ii) relies on SGH to schedule in $\mathcal{S}$ the jobs that are disregarded in step (i).

The pseudo-code of A-SGH is reported at Algorithm 5.4. A-SGH takes the BPMSTP instance $\mathcal{D}(\hat{K})$ and the solution $\mathcal{S}'$ to the BPMSTP instance $\mathcal{D}(\hat{K}+1)$ as inputs and returns a schedule $\mathcal{S}$ for $\mathcal{D}(\hat{K})$. Hereinafter, with a slight abuse of notation, we refer to $\mathcal{S}_h$ as the single-machine schedule obtained by considering the jobs scheduled on machine h, i.e., $\{(j, h_j, \mathcal{T}_j) \in \mathcal{S}, h_j = h\}$. As a practical note, we observe that $\mathcal{S}_h$ can be easily accessed by implementing $\mathcal{S}$ as a collection $\{\mathcal{S}_h, h \in \mathcal{H}\}$ of single-machine schedules, one for each $h \in \mathcal{H}$. A-SGH first initializes $\mathcal{S}$ (line 1) as an empty schedule and declares $\mathcal{J}_d$ (line 2) according to (12). Then, A-SGH starts iterating over each $d \in \mathcal{P}$ according to the LPT rule (line 3). As the first step in the loop, the set $\mathcal{Q}$ is initialized as an empty set (line 4). Such a set is used in the subsequent lines to keep track of the jobs involved in the assignments of $\mathcal{S}'$ that are unfeasible for $\mathcal{S}$. Then, A-SGH iterates over each $j \in \mathcal{J}_d$ to schedule all the jobs in $\mathcal{J}_d$ (lines 5–21). At each iteration, it verifies if the assignment of job j in $\mathcal{S}'$ is also feasible for $\mathcal{S}$. In this case, the assignment of j in $\mathcal{S}'$ is repeated in $\mathcal{S}$; otherwise, j is added to $\mathcal{Q}$. Then, A-SGH computes the sets $\mathcal{L}$ and $\mathcal{R}$ of the possible predecessors and successors of job j on h'_j, respectively, if the assignment of j in $\mathcal{S}'$ was the same as in $\mathcal{S}$ (lines 7-8), where $\mathcal{J}(\mathcal{S})$ denotes the set of jobs scheduled in $\mathcal{S}$. If $\mathcal{L}$ is non-empty, A-SGH sets $\hat{l}$ as the predecessor of j with the greatest start time in $\mathcal{L}$ (line 10). Similarly, if $\mathcal{R}$ is non-empty, A-SGH sets $\hat{u}$ as the successor of j with the lowest start time in $\mathcal{R}$ (line 13). If the processing of j ends after $\hat{K}$ in $\mathcal{S}'$, or there would exist an overlapping predecessor $\hat{l}$ or successor $\hat{r}$ of

Algorithm 5.4 Split-greedy heuristic with assignment history (A-SGH).

Input: A downsized BPMSTP instance $\mathcal{D}(\hat{K})$
A schedule $\mathcal{S}' = \{(j, h'_j, \mathcal{T}'_j) : h'_j \in \mathcal{H}, \mathcal{T}'_j \subseteq \mathcal{T}, \forall j \in \mathcal{J}\}$ for the BPMSTP instance $\mathcal{D}(\hat{K}+1)$
Output: A schedule $\mathcal{S}$ for $\mathcal{D}(\hat{K})$

1: Let $\mathcal{S}$ be an empty schedule
2: Let $\mathcal{J}_d \leftarrow \{j \in \mathcal{J} : p_j = d\}$, $d \in \mathcal{P}$
3: **for** $d \in \mathcal{P}$ in non-increasing order **do**
4: Let $\mathcal{Q} \leftarrow \emptyset$
5: **for** $j \in \mathcal{J}_d$ **do**
6: // Compute the predecessors and the successors of job j in schedule $\mathcal{S}$
7: Let $\mathcal{L} \leftarrow \{l : l \in \mathcal{J}, s_l(\mathcal{S}) \le s_j(\mathcal{S}'), h_l = h'_j\}$
8: Let $\mathcal{R} \leftarrow \{r : r \in \mathcal{J}, s_r(\mathcal{S}) > s_j(\mathcal{S}'), h_r = h'_j\}$
9: **if** $\mathcal{L} \neq \emptyset$ **then**
10: Let $\hat{l} \leftarrow \underset{l \in \mathcal{L}}{arg\,max}\{s_l(\mathcal{S})\}$
11: **end if**
12: **if** $\mathcal{R} \neq \emptyset$ **then**
13: Let $\hat{u} \leftarrow \underset{r \in \mathcal{R}}{arg\,min}\{s_r(\mathcal{S})\}$
14: **end if**
15: // Check whether the assignment of j in $\mathcal{S}'$ is also feasible in $\mathcal{S}$
16: **if** $s_j(\mathcal{S}') + p_j - 1 > \hat{K}$ **or** ($\mathcal{L} \neq \emptyset$ **and** $s_{\hat{l}}(\mathcal{S}) + p_{\hat{l}} - 1 \ge s_j(\mathcal{S}')$) **or** ($\mathcal{R} \neq \emptyset$ **and** $s_{\hat{u}}(\mathcal{S}) \le s_j(\mathcal{S}') + p_j - 1$) **then**
17: $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{j\}$ // If not, add j to the separate set of jobs $\mathcal{Q}$ to be scheduled after the end of the loop
18: **else**
19: $\mathcal{S}_{h'_j} \leftarrow \mathcal{S}_{h'_j} \cup (j, h'_j, \mathcal{T}'_j)$ // Otherwise, perform the same assignment in $\mathcal{S}$
20: **end if**
21: **end for**
22: $\hat{\mathcal{S}} \leftarrow$ SGH($(\mathcal{Q}, \{p_j, j \in \mathcal{Q}\}, \mathcal{H}, \{u_h, h \in \mathcal{H}\}, \{1, \ldots, \hat{K}\}, \{c_t, t \in \{1, \ldots, \hat{K}\}\})$)
23: **if** $\hat{\mathcal{S}} = \emptyset$ **then return** $\emptyset$
24: $\mathcal{S} \leftarrow \mathcal{S} \cup \hat{\mathcal{S}}$
25: **end for**
26: **return** $\mathcal{S}$

j on machine h'_j if job j was assigned in $\mathcal{S}$ as in $\mathcal{S}'$ (line 16), then job j is added to the set $\mathcal{Q}$ (line 17). Otherwise, the assignment of j in $\mathcal{S}'$ is feasible in $\mathcal{S}$ as well, and therefore $\mathcal{S}$ is updated accordingly (line 19). Finally, A-SGH schedules the jobs belonging to the set $\mathcal{Q}$ by using SGH (line 22). If SGH cannot compute a feasible schedule for $\mathcal{D}(\hat{K})$ for the jobs in $\mathcal{Q}$, i.e., $\hat{\mathcal{S}}$ is empty, then A-SGH returns an empty schedule as well and stops (line 23). Otherwise, $\mathcal{S}$ is properly updated at line 24. Finally, A-SGH returns the computed feasible schedule at line 26.

5.3.2. Exchange search with rescheduling

In this subsection, we present *Exchange search with rescheduling* (R-ES), which builds upon ES (Algorithm 5.3) and circumvents its computational drawbacks. In particular, among all possible EPS moves, ES also considers the moves involving EPS-Is containing only free slots, hereinafter referred to as *empty* (see line 5 of Algorithm 5.3). However, given two machines $h^{\mathrm{J}}, h^{\mathrm{I}} \in \mathcal{H}$, if an EPS move involving an EPS-J $(\mathcal{E}^{\mathrm{J}}, h^{\mathrm{J}})$ and an empty EPS-I $(\mathcal{E}^{\mathrm{I}}, h^{\mathrm{I}})$ is found to be not improving, then the other subsequent EPS moves involving $(\mathcal{E}^{\mathrm{J}}, h^{\mathrm{J}})$ and other empty EPS-Is whose cost is higher than the cost of $(\mathcal{E}^{\mathrm{I}}, h^{\mathrm{I}})$ cannot be improving as well within the same iteration over $(\mathcal{E}^{\mathrm{J}}, h^{\mathrm{J}})$. In fact, ES iterates over the EPS-Is by disregarding their energy cost. In this way, it considers the empty EPS-Is in no specific order. These observations may prospect a high computational burden, especially in the last iterations performed by ES, when the number of possible improving EPS moves is smaller as compared to the first iterations. The core idea of R-ES is to apply a local search strategy that improves a feasible input schedule $\mathcal{S}$, by first applying ES while disregarding the empty EPS-Is and then performing the rescheduling of each job (ordered according to the LPT rule) to take into account all such empty EPS-Is. In particular, R-ES removes a job at a time from the current schedule $\mathcal{S}$, and greedily reinserts the same job in $\mathcal{S}$ by assigning it to the free location or free split-location with the smallest-cost in $\mathcal{S}$.

The pseudo-code of R-ES is reported in Algorithm 5.5. R-ES

Algorithm 5.5 Exchange search with rescheduling (R-ES).

Input: A feasible schedule $\mathcal{S}$ for the downsized BPMSTP instance $\mathcal{D}(\hat{K})$ as in (26)
Output: A feasible schedule $\mathcal{S}'$ for $\mathcal{D}(\hat{K})$, with $C^{\max}(\mathcal{S}') \le C^{\max}(\mathcal{S})$ and $E(\mathcal{S}') \le E(\mathcal{S})$

1: **repeat**
2: Let $\mathcal{S}' \leftarrow \mathcal{S}$
3: Update $\mathcal{S}$ with ES by disregarding empty EPS-Is (at line 5 of Algorithm 5.3)
4: **for** $d \in \mathcal{P}$ in non-increasing order **do**
5: Let $L_{d,h}$, $h \in \mathcal{H}$, be the lists of the smallest-cost free locations and split-locations on h for any $j : p_j = d$
6: **for** $j \in \mathcal{J}_d$ **do**
7: $\mathcal{S} \leftarrow \mathcal{S} \setminus (j, h_j, \mathcal{T}_j)$
8: Update $L_{d,h}$ by adding location $(h_j, \mathcal{T}_j)$
9: Select a location $\hat{l} = (\hat{h}, \hat{\mathcal{A}})$ from the smallest-cost locations in $\bigcup_{h \in \mathcal{H}}\{l \in L_{d,h}\}$ randomly
10: Assign job j to $\hat{l}$ by adding $(j, \hat{h}, \hat{\mathcal{A}})$ to $\mathcal{S}_{\hat{h}}$
11: Update list $L_{d,\hat{h}}$ by removing the locations affected by the assignment of j
12: **end for**
13: **end for**
14: **if** $\mathcal{S}$ is a split-schedule **then** convert it into an equivalent feasible schedule
15: **until** $E(\mathcal{S}) < E(\mathcal{S}')$
16: **return** $\mathcal{S}'$

takes a feasible schedule $\mathcal{S}$ for the downsized BPMSTP instance $\mathcal{D}(\hat{K})$ as input and returns a feasible schedule $\mathcal{S}'$ for $\mathcal{D}(\hat{K})$ with $C^{\max}(\mathcal{S}') \le C^{\max}(\mathcal{S})$ and $E(\mathcal{S}') \le E(\mathcal{S})$. In fact, the purpose of R-ES is to improve the TEC of $\mathcal{S}$, while possibly improving also the makespan as a byproduct of the performed EPS moves. However, the makespan cannot be worsened by construction, as in ES. Specifically, R-ES iteratively improves the TEC of $\mathcal{S}$ until at least an improving move is performed (see the loop at lines 1–15). At the beginning of the loop, R-ES stores the current schedule in $\mathcal{S}'$ (line 3). This allows checking the termination condition at the end of the loop (line 15). Then, it tries to improve $\mathcal{S}$ by applying ES without considering the empty EPS-Is in $\mathcal{S}$ (line 3). Next, similarly to SGH (Algorithm 5.2), R-ES iterates over the processing times of the jobs in $\mathcal{J}$ in non-increasing order (lines 4–13), by first building the list of free locations and free split-locations (line 5) as in line 4 of Algorithm 5.2. Subsequently, it iterates over each $j \in \mathcal{J}_d$ (lines 6–12). Inside this loop, R-ES first removes the assignment $(j, h_j, \mathcal{T}_j)$ from $\mathcal{S}$ (line 7) and then adds the new free location $(h_j, \mathcal{T}_j)$ to $L_{d,h}$ (line 8). Afterward, R-ES reassigns j in $\mathcal{S}$ through lines 9–11, which are identical to lines 7–9 of SGH (Algorithm 5.2). If the resulting schedule is a split-schedule, then it is converted into a feasible one (line 14). R-ES stops the loop if it is unable to further improve the TEC of $\mathcal{S}$ (line 15), i.e., if $E(\mathcal{S}) = E(\mathcal{S}')$. Otherwise, it starts another

iteration to search for further improvements. Eventually, R-ES returns the resulting schedule (line 16).

5.3.3. Exact single-machine rescheduler

In this subsection, we introduce a novel algorithm based on dynamic programming that is able to further improve the TEC of a single-machine feasible schedule $\mathcal{S}_h$, for some $h \in \mathcal{H}$ over the time slots $t = 1, 2, \ldots, \hat{K}$, with $\hat{K} \leq K$. We refer to such an algorithm as *Exact Single-machine Rescheduler* (ESR). ESR computes the assignment of the jobs scheduled on machine h so as to minimize the energy cost of the jobs scheduled on the machine while preserving the processing sequence of such jobs. The design of ESR is inspired by the work of Chen et al. (2021), who investigated the problem of minimizing the TEC of a preemptive schedule on a single machine. Toward the end of describing ESR, we first denote by $\mathcal{U}$ the set $\{j^{(1)}, j^{(2)}, \ldots, j^{(U)}\}$ of the $U > 0$ jobs scheduled on the machine, and, with a slight abuse of notation, we define

$$q^{(i)} = p_{j^{(i)}}, \quad i = 1, 2, \ldots, U$$

for easier readability. We refer to

$$\mathcal{S}^\star(i,t), \quad i = 1, 2, \ldots, U,\ t = 1, 2, \ldots, \hat{K} - \sum_{u=i}^{U} q^{(u)} + 1$$

as the optimal schedule for jobs $j^{(i)}, j^{(i+1)}, \ldots, j^{(U)}$ in the time slots $t, t+1, \ldots, \hat{K} - \sum_{u=i}^{U} q^{(u)} + 1$. Observe that a feasible schedule for the last i jobs in the processing sequence on machine h requires at least $\sum_{u=i}^{U} q^{(u)}$ time slots for processing. We also define

$$B_t^{t+l-1} = \sum_{k=t}^{t+l-1} c_k, \quad t = 1, 2, \ldots, \hat{K},\ l = 1, 2, \ldots, \hat{K} - t + 1,$$

as the cumulative cost of time slots $t, t+1, \ldots, t+l-1$. For each $i = 1, 2, \ldots, U$ and $t = 1, 2, \ldots, \hat{K}$, we refer to

$$V_{i,t} := E(\mathcal{S}^\star(i,t))$$

as the energy cost of the schedule $\mathcal{S}^\star(i,t)$. Then, ESR employs the following recursive relation for the value of the energy cost of the optimal schedule $\mathcal{S}^\star(i,t)$:

$$V_{i,t} = \min\left\{V_{i,t+1}, B_t^{t+q^{(i)}-1} + V_{i+1,t+q^{(i)}}\right\}, \quad i = 1, 2, \ldots, U,$$
$$t = \sum_{u=1}^{i-1} q^{(u)} + 1, \ldots, \hat{K} - \sum_{u=i}^{U} q^{(u)}. \tag{27}$$

Observe that the first $i-1$ jobs and the last $i+1$ jobs in the processing sequence require at least $\sum_{u=1}^{i-1} q^{(u)}$ and $\sum_{u=i+1}^{U} q^{(u)}$ time slots for processing, respectively. Eq. (27) recursively expresses the cost of the optimal schedule $\mathcal{S}^\star(i,t)$ as the minimum between (i) the cost $V_{i,t+1}$ of the optimal schedule $\mathcal{S}^\star(i,t+1)$ (where job $j^{(i)}$ does not start on time slot t) and (ii) the cost of scheduling job $j^{(i)}$ starting from time slot t plus the cost of optimally scheduling the last $i+1$ jobs in the processing sequence in the time slots after $t + q^{(i)} - 1$, that is the cost $V_{i+1,t+q^{(i)}}$ of the optimal schedule $\mathcal{S}^\star(i+1, t+q^{(i)})$. The base case conditions for Eq. (27) are given by

$$V_{i,\hat{K}-\sum_{u=i}^{U} q^{(u)}+1} = B^{\hat{K}}_{\hat{K}-\sum_{u=i}^{U} q^{(u)}+1}, \quad i = 1, 2, \ldots, U, \tag{28}$$

$$V_{U+1,t} := 0, \quad t = 1, 2, \ldots, \hat{K}. \tag{29}$$

The base case condition (28) ensures that the optimal cost $V_{i,\hat{K}-\sum_{u=i}^{U} q^{(u)}+1}$ of scheduling the last i jobs in the processing sequence on the machine in the last $\sum_{u=i}^{U} q^{(u)}$ time slots is indeed the sum of the costs of the slots from $\hat{K} - \sum_{u=i}^{U} q^{(u)} + 1$ to $\hat{K}$, for $i = 1, 2, \ldots, U$. Condition (29) allows correctly expressing $V_{U,t}$ as $\min\{V_{U,t+1}, B_t^{t+q^{(U)}-1}\}$ for $t = 1, 2, \ldots, \hat{K} - q^{(U)}$. Finally, we refer to

$$W_{i,t} := s_{j^{(i)}}(\mathcal{S}^\star(i,t)), \tag{30}$$

as the first time slot of job $j^{(i)}$ in schedule $\mathcal{S}^\star(i,t)$.

The pseudo-code of ESR is reported in Algorithm 5.6. ESR takes

Algorithm 5.6 Exact single-machine rescheduler (ESR).

Input: A feasible single-machine schedule $\mathcal{S}_h$ on some machine $h \in \mathcal{H}$ for a set of $U > 0$ jobs $\mathcal{U} = \{j^{(1)}, j^{(2)}, \ldots, j^{(U)}\} \subseteq \mathcal{J}$ in the time slots $t = 1, 2, \ldots, \hat{K}$ with $\hat{K} \leq K$

Output: The optimal schedule $\mathcal{S}_h^\star$ for the jobs in $\mathcal{U}$ in the time slots $t = 1, 2, \ldots, \hat{K}$.

1: // Parameter initialization
2: Let $l_i \leftarrow 0$ and $r_i \leftarrow 0$ for $i = 0, 1, \ldots, U$
3: **for** $i \leftarrow 1$ **to** U **do**
4: Set $l_i \leftarrow q^{(i)} + l_{i-1}$, $r_i \leftarrow q^{(U-i+1)} + r_{i-1}$
5: **end for**
6: // Initialization of base case values
7: Let $V_{i,t} \leftarrow 0$, $W_{i,t} \leftarrow 0$ for $i = 1, 2, \ldots, U+1$, $t = 1, 2, \ldots, \hat{K}$
8: **for** $i \leftarrow 1$ to U **do**
9: Let $t \leftarrow \hat{K} - r_{U-i+1} + 1$
10: $V_{i,t} \leftarrow B_t^{\hat{K}}$
11: $W_{i,t} \leftarrow t$
12: **end for**
13: // Main loop
14: **for** $i \leftarrow U$ **downto** 1 **do**
15: **for** $t \leftarrow \hat{K} - r_{U-i+1}$ **downto** $l_i + 1$ **do**
16: **if** $B_t^{t+q^{(i)}-1} + V_{i+1,t+q^{(i)}} < V_{i,t+1}$ **then**
17: Set $V_{i,t} \leftarrow B_t^{t+q^{(i)}-1} + V_{i+1,t+q^{(i)}}$, $W_{i,t} \leftarrow t$
18: **else**
19: Set $V_{i,t} \leftarrow V_{i,t+1}$, $W_{i,t} \leftarrow W_{i,t+1}$
20: **end if**
21: **end for**
22: **end for**
23: // Generation of a schedule with the computed optimal cost
24: Let $\mathcal{S}_h^\star \leftarrow \emptyset$, $k \leftarrow 0$
25: **for** $i \leftarrow 1$ **to** U **do**
26: Set $k \leftarrow W_{i,k+1}$
27: $\mathcal{S}_h^\star \leftarrow \mathcal{S}_h^\star \cup \{j^{(i)}, h, \{k, k+1, \ldots, k+q^{(i)}-1\}\}$
28: Set $k \leftarrow k + q^{(i)} - 1$
29: **end for**
30: **return** $\mathcal{S}_h^\star$

a feasible single-machine schedule $\mathcal{S}_h$ for some machine $h \in \mathcal{H}$ as input and returns a feasible schedule that achieves the minimum energy cost on machine h while preserving the original job processing sequence in the schedule $\mathcal{S}_h$. First, ESR initializes the parameters l_i, r_i for $i = 0, 1, \ldots, U$, so that l_i and r_i are equal to the sum of the processing times of the first and the last i jobs in the processing sequence, respectively, and $l_0 = r_0 = 0$ (lines 2–5). Subsequently, ESR declares the optimal cost $V_{i,t}$ and the first time slot $W_{i,t}$ for job i for $i = 1, 2, \ldots, U$, $t = 1, 2, \ldots, \hat{K}$, and it also sets $V_{U+1,t} = 0$ for $t = 1, 2, \ldots, \hat{K}$ according to condition (29) (line 7). Then, it iterates over each $i = 1, 2, \ldots, U$ (lines 8–12) to initialize $V_{i,t}$ with $t = \hat{K} - \sum_{u=i}^{U} q^{(u)} + 1 = \hat{K} - r_{U-i+1} + 1$ according to (28). ESR also consistently sets the start time $W_{i,t}$ of job i to t (line 11). Afterward, ESR enters the main loop of the algorithm (lines 14–22), which provides a bottom-up implementation of the recursive relation (27). Finally, ESR exploits the information computed at the previous lines to build a schedule with optimal cost given by $V_{1,1}$ (lines 24–29). Toward this end, the new schedule $\mathcal{S}_h^\star$ and the auxiliary variable k are set to an empty schedule and to 0, respec-

tively (line 24). Then, at each iteration of lines 25–29, (i) the start time $W_{i,k+1}$ of job i is assigned to the variable k (line 26), (ii) $\mathcal{S}_h^\star$ is updated by assigning job $j^{(i)}$ to the location $(h, \{k, k+1, \ldots, k+q^{(i)}-1\})$ (line 27), and finally (iii) k is updated so that the next iteration (if $i < U$) considers the optimal assignment for job $i+1$ after time slot $k+q^{(i)}-1$, i.e., after the last slot used to process job i (line 28). Lastly, ESR returns the new schedule $\mathcal{S}_h^\star$. Observe that, at the first iteration, i.e., when $i = 1$, k is first set as $W_{1,1}$.

5.3.4. Enhanced heuristic scheduler

We are finally able to describe *Enhanced heuristic scheduler* (EHS), which combines A-SGH, R-ES, and ESR to solve instances of the BPMSTP. This algorithm computes a set of non-dominated solutions for an instance $\mathcal{I}$ of the BPMSTP with a very low computational burden. Clearly, EHS benefits from the aforementioned computational improvements of A-SGH and R-ES since it applies such two algorithms sequentially. However, such a combination entails a further advantage with respect to SGS-ES. Specifically, within SGS-ES, it is useless to consider EPS swaps in ES involving an EPS-J and an empty EPS-I before the first improving EPS move since ES is preceded by SGH, i.e., a constructive greedy heuristic based on the LPT rule. Indeed, as empty EPS-Is correspond to free locations, if there was such an improving EPS swap, then the empty EPS-I involved in the swap would have been greedily chosen by SGH as a free location for the job in the involved EPS-J. The further improvements of the schedules on the various machines enabled by ESR provide very effective results.

The pseudo-code of EHS is reported in Algorithm 5.7. EHS is

Algorithm 5.7 Enhanced heuristic scheduler (EHS).

Input: A BPMSTP instance $\mathcal{I}$
Output: A set of non-dominated heuristic solutions to $\mathcal{I}$

1: Let $\underline{K}(\mathcal{I})$ be the lower bound defined in (25)
2: Let $\mathcal{O} \leftarrow \emptyset$, $\hat{K} \leftarrow K$, $\mathcal{S}' \leftarrow \emptyset$
3: **while** $\hat{K} \geq \underline{K}(\mathcal{I})$ **do**
4: Let $\mathcal{D}(\hat{K})$ be an instance as in (26)
5: // If it is the first iteration of the loop, compute the schedule with SGH
6: **if** $\mathcal{S}' = \emptyset$ **then**
7: $\mathcal{S} \leftarrow \text{SGH}(\mathcal{D}(\hat{K}))$
8: **else** // Otherwise, compute it with A-SGH, by exploiting previous job assignments
9: $\mathcal{S} \leftarrow \text{A-SGH}(\mathcal{D}(\hat{K}), \mathcal{S}')$
10: **end if**
11: **if** $\mathcal{S}$ is unfeasible **then** // Checks if $\mathcal{S} = \emptyset$
12: **break**
13: **end if**
14: $\mathcal{S}' \leftarrow \text{R-ES}(\mathcal{S})$ // Apply R-ES to improve the TEC of $\mathcal{S}$
15: // Apply ESR to further improve the TEC by minimizing the cost on each machine
16: Let $\mathcal{S}'' \leftarrow \emptyset$
17: **for** $h \in \mathcal{H}$ **do**
18: $\mathcal{S}_h^\star \leftarrow \text{ESR}(\mathcal{S}'_h)$
19: $\mathcal{S}'' \leftarrow \mathcal{S}'' \cup \mathcal{S}_h^\star$
20: **end for**
21: Set $\mathcal{O} \leftarrow \mathcal{O} \cup \{\mathcal{S}''\}$
22: $\hat{K} \leftarrow \hat{K} - 1$
23: **end while**
24: **return** the set of non-dominated solutions in $\mathcal{O}$

based on the ϵ-constraint method, like the exact algorithm described in Section 5.1. In more detail, it first initializes the set $\mathcal{O}$ of computed solutions to an empty set and $\hat{K}$ to K (line 2). Then, EHS iterates over $\hat{K}$ from K to the lower bound $\underline{K}(\mathcal{I})$ (lines 3–23). At the first iteration, i.e., when $\hat{K} = K$, EHS cannot exploit any previous assignment, $\mathcal{S}'$ is empty, and the solution $\mathcal{S}$ for $\mathcal{D}(K)$ is generated with SGH (line 7). Instead, for $\hat{K} = K-1, K-2, \ldots, \underline{K}(\mathcal{I})$, it is possible to leverage the assignments in the feasible schedule $\mathcal{S}'$ to compute $\mathcal{S}$. Hence, in this case, EHS computes the solution $\mathcal{S}$ for $\mathcal{D}(\hat{K})$ by using A-SGH (line 9). The loop is stopped at line 12 if $\mathcal{S}$ is unfeasible. Otherwise, $\mathcal{S}$ is improved through R-ES (line 14) as well as ESR (lines 17–20), and the resulting schedule is assigned to $\mathcal{S}''$. Then, EHS updates the set $\mathcal{O}$ by adding $\mathcal{S}''$ (line 21) and decreases $\hat{K}$ by 1 (line 22). Finally, it returns the set of non-dominated solutions in $\mathcal{O}$ (line 24).

6. Numerical results

In this section, we report the results of the experimental tests aimed at evaluating the performance of the solution approaches described in Section 5. Specifically, the tests are motivated by the following goals:

(a) investigate the differences between Formulation 1 and Formulation 2 when used in the exact algorithm from an experimental standpoint;
(b) assess the effectiveness of EHS with respect to the state-of-the-art heuristic SGS-ES (Anghinolfi et al., 2021), as well as the constructive heuristic CH by Wang et al. (2018) and an implementation of the Non-dominated Sorting Genetic Algorithm (NSGA-III) (Deb & Jain, 2013) initialized with CH;
(c) measure the speed-up achieved by the exact algorithm when provided with an initial solution computed by EHS;
(d) evaluate the performance of EHS with respect to the exact algorithm with Formulation 2. In particular, the comparison of their computational times allows us to investigate the impact of the trade-off between solution quality and computational efficiency.

The remainder of this section is organized as follows. Section 6.1 provides a description of the test instances and the implementation of the algorithms. Section 6.2 presents the metrics used to evaluate the performance of the algorithms. Section 6.3 focuses on goal (a), Section 6.4 deals with goal (b), while Section 6.5 addresses both goals (c) and (d).

6.1. Test instances and implementation details

The experimental tests were carried out on a set of 90 BPMSTP instances, numbered from 1 to 90. The first 60 instances were originally proposed by Wang et al. (2018), while the last set of 30 instances was introduced by Anghinolfi et al. (2021). Instances 1–30, 31–60, and 61–90 are called small-scale, medium-scale, and large-scale instances, respectively. The instances differ in the values of the number of jobs N, the number of machines M, and the number of time slots K. In particular, the values of N, M, and K for instances 1–30 are no greater than the values of N, M, and K, respectively, for instances 31–60. The same applies to instances 31–60 and 61–90. Appendix A provides a description of all the considered 90 instances.

Concerning implementation, we used the Java 16 programming language for the exact algorithm (Algorithm 5.1), EHS, SGS-ES, CH, and NSGA-III. The exact algorithm also exploits the Java CPLEX 20.1.0 API. In particular, we set the maximum optimality gap of the MILP solver to 10^{-6} and a time limit of 4 hours. As regards CH and NSGA-III, we reimplemented and adapted the MATLAB code shared by Wang et al. (2018). More specifically, we developed the implementation of NSGA-III consistently with the design proposed by Wang et al. (2018) for NSGA-II, i.e., we used the same solution representation, initialization procedure, crossover and mutation operators, as well as the same parameter settings (more details can be

found in Wang et al., 2018). We performed all the experimental tests on a Windows 10 system equipped with an Intel Core i9-9900K Octa-core 3.6 GHz processor and 16 GB of RAM. For the sake of computational performance, the implementation of the exact algorithm does not generate a CPLEX representation of the mathematical formulation for $\mathcal{D}(\hat{K})$ from scratch at each iteration, as it may be suggested by the pseudo-code at line 3 of Algorithm 5.1. Instead, such a representation is generated only at the first iteration, i.e., for $\hat{K} = K$. The CPLEX representations in the subsequent iterations are obtained from the first one by adding proper constraints, which set the decision variables related to the slots that are not in $\mathcal{D}(\hat{K})$ to zero as follows:

$$X_{j,h,t} = 0, \quad j \in \mathcal{J}, h \in \mathcal{H}, \hat{K} - p_j + 1 < t \leq K$$

for Formulation 1, and

$$Y_{d,h,t} = 0, \quad d \in \mathcal{P}, h \in \mathcal{H}, \hat{K} - d + 1 < t \leq K$$

for Formulation 2. Such constraints avoid assigning jobs with completion times exceeding $\hat{K}$. As a consequence, the makespan cannot exceed $\hat{K}$, as desired. According to experimental evidence, the computational overhead caused by such additional constraints is lower than the one due to the generation of new CPLEX representations at each iteration. The numerical results obtained with the algorithm implementations are available at "https://github.com/ORresearcher/Exact-and-Heuristic-Solution-Approaches-for-Energy-Efficient-Identical-Parallel-Machine-Scheduling".

6.2. Performance metrics

Comparing different solutions to the same instance of a multi-objective optimization problem is not a straightforward task. In fact, in contrast to single-objective optimization problems, the solution to an instance of a multi-objective problem is a set of non-dominated points. As a consequence, we employ two distinct state-of-the-art metrics for the purpose of comparing different sets of non-dominated points, so as to provide an in-depth analysis of the performances achieved by the proposed solution approaches. More specifically, we use two *convergence and distribution* metrics, according to the classification proposed by Audet et al. (2021). Given a set of non-dominated points, such metrics are indeed able to quantify at the same time how close such a set is to the Pareto front and how it is distributed in the objectives space.

Let $\mathcal{O} \subseteq \mathbb{R}^n$ be a set of non-dominated points in the objective space, and $\mathcal{F}$ be its reference Pareto-optimal front. The first metric used in this paper, called *Hypervolume* (Guerreiro et al., 2021; Zitzler & Thiele, 1999), measures the hypervolume covered by $\mathcal{O}$ with respect to a reference point in the space of the objectives. As such, it can be used to compare two or more fronts by assuming a common reference point. Formally, let r be a reference point in $\mathbb{R}^n$. Then, the Hypervolume of $\mathcal{O}$ is the measure of the region weakly dominated by $\mathcal{O}$ and bounded above by r, i.e.,

$$\mathrm{H}(\mathcal{O}) = \Lambda(\{q \in \mathbb{R}^n : \exists p \in \mathcal{O} : p \leq q \leq r\}),$$

where $\Lambda(\cdot)$ is the Lebesgue measure. Larger Hypervolume values denote a better approximation of the Pareto-optimal front. We observe that the Hypervolume metric is Pareto-compliant, as pointed out by Zitzler et al. (2007).

The other metric used for comparisons is called *Modified inverted generation distance* (IGD^+) (Ishibuchi et al., 2015). IGD^+ is a metric that extends the Inverted generation distance (IGD) (Coello Coello & Reyes Sierra, 2004) to achieve Pareto-compliance. Both IGD and IGD^+ measure the quality of a set of non-dominated points $\mathcal{O}$ computed by an algorithm in comparison to a set of reference solutions $\mathcal{F}$, which typically consists of a set of Pareto-optimal solutions. In particular, IGD measures the average distance from each point in $\mathcal{F}$ to the nearest point in $\mathcal{O}$ by generally using the Euclidean distance metric. Lower values for IGD correspond to a better approximation of the Pareto front. However, IGD fails to be compliant with the Pareto-dominance relation when comparing two different non-dominated sets. In fact, IGD may not always assign the lower value to the dominating set. IGD^+ overcomes this drawback by employing a different distance metric that we denote by $d^+(\cdot,\cdot)$. Specifically, given a reference point $p \in \mathcal{F}$ and a non-dominated point $q \in \mathcal{O}$, such a distance is computed as

$$d^+(p,q) = \left(\sum_{i=1}^{n} (\max\{q_i - p_i, 0\})^2 \right)^{\frac{1}{2}},$$

where p_i and q_i are the values of p and q, respectively, associated with the ith objective. Then, the IGD^+ value for the set $\mathcal{O}$ of non-dominated points is given by

$$\mathrm{IGD}^+(\mathcal{O}) = \frac{1}{|\mathcal{F}|} \sum_{p \in \mathcal{F}} \min_{q \in \mathcal{O}} d^+(p,q).$$

6.3. Comparing the two mathematical formulations

This subsection evaluates the two mathematical formulations introduced in Section 3 when used in the exact algorithm, by comparing the computational times achieved on the test instances. Toward this end, the MILP solver used at line 3 of Algorithm 5.1 was first implemented with Formulation 1, and then with Formulation 2.

Table 1 reports the results of the comparison for instances 1–60. The exact algorithm was not able to solve large-scale instances 61–90 with Formulation 1 due to the memory constraints of the experimental setup. On the other hand, Formulation 2 was capable of solving all of such instances, as reported in later subsections. Table 1 shows the computational times obtained by using Formulation 1 and Formulation 2, along with their percentage deviation Δ. Formally, given the computational times $\delta_{F1,i}$ and $\delta_{F2,i}$, $i = 1, 2, \ldots, 60$, obtained with Formulation 1 and Formulation 2, respectively, the value of the percentage deviation for the ith instance is given by $1 - \delta_{F2,i}/\delta_{F1,i}$, so that an improvement of Formulation 2 with respect to Formulation 1 is always expressed as a percentage between 0% and 100%. In the table, all the values of the computational times were rounded to 4 decimal places. The best result for each instance is highlighted in bold, according to the actual (non-rounded) value. At the bottom of the table, the rows "Avg." and "Std dev." report the averages and the standard deviation, respectively, of the computational times and percentage deviations on instances 1–30 and 31–60. The row "N. best" shows the number of instances where one of the implementations of the exact algorithm outperforms or equals the other, while "N. draw" indicates the number of instances such that the various approaches obtain the same performance. Finally, the last row displays the p-values obtained with the non-parametric Wilcoxon signed rank test (Gibbons & Chakraborti, 2020), aimed at assessing the statistical significance of the difference between the computational times obtained by Formulation 1 and Formulation 2. In more detail, we assume that, if p is smaller than 0.05, then we can reject the null hypothesis that the two formulations generate non-significantly different results. In the tables presented in the remainder of the section, we adopt the same conventions used in Table 1 as regards boldface highlighting, decimal rounding, and rows "Avg.", "Std dev." "N. best", "N. draw", and p. The following subsections display the percentage deviation for Hypervolume and IGD^+ as well. Throughout the section, the symbol Δ, used for the percentage deviation, reports the solution approach that is evaluated for the considered metric as a subscript, and the one used as a reference as a superscript. For instance, in this subsection, we aim at showing the

Table 1
Comparison of the computational times obtained by the exact algorithm when using Formulation 1 (F1) and Formulation 2 (F2) for instances 1–30 and 31–60.

Instance	CPU time (s)			Instance	CPU time (s)		
	F1	F2	Δ_{F2}^{F1} (%)		F1	F2	Δ_{F2}^{F1} (%)
1	0.7670	**0.3525**	54.04	31	7.4852	**0.9499**	87.31
2	0.6523	**0.4799**	26.43	32	21.4092	**1.1458**	94.65
3	0.3438	**0.3041**	11.54	33	67.1348	**1.5192**	97.74
4	**0.3968**	0.5265	−32.67	34	178.3155	**2.2330**	98.75
5	0.4713	**0.4288**	9.02	35	265.2156	**2.5779**	99.03
6	0.6932	**0.4466**	35.57	36	10.3682	**0.9013**	91.31
7	0.6179	**0.3865**	37.44	37	29.6823	**1.3302**	95.52
8	0.9727	**0.5586**	42.57	38	129.6473	**2.5866**	98.00
9	0.5401	**0.2809**	47.99	39	219.0398	**2.7987**	98.72
10	0.9134	**0.7209**	21.07	40	416.4091	**4.1294**	99.01
11	0.7175	**0.4399**	38.69	41	16.1153	**0.9398**	94.17
12	1.2644	**0.7382**	41.62	42	38.0861	**1.7630**	95.37
13	0.8214	**0.4236**	48.43	43	124.4411	**2.6687**	97.86
14	1.2381	**0.6709**	45.81	44	333.3671	**3.7989**	98.86
15	0.7826	**0.3676**	53.04	45	650.1111	**5.8304**	99.10
16	1.7645	**0.7276**	58.76	46	45.6422	**3.4909**	92.35
17	1.1950	**0.5931**	50.37	47	98.6533	**4.2710**	95.67
18	2.4932	**1.0311**	58.64	48	296.3851	**7.3489**	97.52
19	1.2564	**0.4866**	61.27	49	1242.2259	**17.5062**	98.59
20	1.8639	**0.6527**	64.98	50	1291.4013	**11.3853**	99.12
21	2.2242	**0.6678**	69.98	51	132.9453	**7.2481**	94.55
22	2.8476	**0.9895**	65.25	52	101.9938	**5.3167**	94.79
23	1.5190	**0.4635**	69.49	53	509.0894	**10.1980**	98.00
24	4.1561	**1.2782**	69.24	54	819.4229	**12.6210**	98.46
25	1.1941	**0.2998**	74.89	55	2225.5660	**18.1268**	99.19
26	3.1824	**0.8721**	72.60	56	93.0159	**4.1815**	95.50
27	1.6282	**0.4519**	72.25	57	231.8769	**10.9855**	95.26
28	4.6395	**1.1910**	74.33	58	537.1396	**13.1024**	97.56
29	3.7396	**0.9733**	73.97	59	1705.2297	**21.4852**	98.74
30	4.3741	**0.9687**	77.85	60	2329.4400	**20.7640**	99.11
Avg.	1.6424	**0.6258**	49.82	**Avg.**	472.2285	**6.7735**	96.66
Std dev.	1.2561	**0.2731**	24.57	**Std dev.**	646.3344	**6.2401**	2.80
N. best	1	29		**N. best**	0	30	
N. draw	0			**N. draw**	0		
p	2.3534E-06			***p***	1.7344E-06		

computational times obtained with Formulation 2 in comparison with the ones of Formulation 1, and we consistently express the percentage deviation between the computational times as Δ_{F2}^{F1}.

The implementation of the exact algorithm based on Formulation 2 outperforms the one exploiting Formulation 1 in all instances with the only exception of instance 4, where Formulation 2 is 32.67% worse than Formulation 1. However, the average computational time of the former implementation on medium-scale instances 31–60 is two orders of magnitude lower than the average time achieved by the latter, with a remarkable average percentage deviation equal to 96.66%. Moreover, the computational advantage of Formulation 2 with respect to Formulation 1 on the small-scale instances, i.e., instances 1–30, is still noteworthy, with an average percentage deviation of 49.82%. We observe that the small values of p denote the statistical significance of the numerical results.

6.4. Comparing EHS with the other heuristics

This subsection first compares EHS, i.e., the novel heuristic proposed in this paper, with the state-of-the-art heuristic SGS-ES. Tables 2–4 report the values of the performance metrics introduced in Section 6.2, i.e., Hypervolume and IGD$^+$, and the computational times obtained by EHS and SGS-ES on instances 1–30, 31–60 and 61–90, respectively. Each of the three tables adopts the same conventions used for Table 1. In particular, $\Delta_{EHS}^{SGS\text{-}ES}$ denotes the percentage deviation of the performance of EHS with respect to SGS-ES. For instance, a positive value of $\Delta_{EHS}^{SGS\text{-}ES}$ for the computational times indicates that EHS requires less time than SGS-ES. On the other hand, a positive value of $\Delta_{EHS}^{SGS\text{-}ES}$ for the Hypervolume highlights that EHS is able to compute a better Pareto front with respect to SGS-ES. Throughout the section, whenever the percentage deviation cannot be computed due to a division by zero, we write a dash instead. Since both EHS and SGS-ES perform stochastic choices, we averaged the results obtained over 10 independent runs of the algorithms.

The numerical results show that EHS achieves better results than SGS-ES in terms of Hypervolume and IGD$^+$ for most of instances 1–30 and 31–60, and for all instances 61–90. The results obtained in terms of Hypervolume are coherent with those obtained in terms of IGD$^+$. Furthermore, EHS outperforms SGS-ES in terms of computational times on 83 instances over 90, with the average of 10.12%, 48.97%, and 63.04% improvements on instances 1–30, 31–60, and 61–90, as shown in Tables 2–4, respectively. Such improvements are significant also in the case of instances 1–30, where the CPU times are in the orders of few milliseconds, and where also the exact algorithm using Formulation 2 is able to find a solution always in less than 5 seconds (see Table 1). The small values of p support the relevance of these results, proving that the difference between the series of data is statistically significant. This is more evident for instances 61–90, where the p-values are in the order of 10^{-6} or 10^{-8}. Thus, we conclude that EHS proves to be more computationally efficient with respect to SGS-ES while preserving the quality of the computed solutions. In more detail, EHS consistently outperforms SGS-ES on all the large-scale instances 61–90, as regards the values of Hypervolume, IGD$^+$, and computational times. These results showcase the capability of EHS to gen-

Table 2
Comparison of the results obtained by EHS and SGS-ES for instances 1–30. The table shows the values of the Hypervolume, IGD$^+$, and computational times achieved by the two heuristics, along with the related percentage deviations.

Instance	Hypervolume			IGD$^+$			CPU time (s)		
	EHS	SGS-ES	Δ_{EHS}^{SGS-ES} (%)	EHS	SGS-ES	Δ_{EHS}^{SGS-ES} (%)	EHS	SGS-ES	Δ_{EHS}^{SGS-ES} (%)
1	**0.6976**	**0.6976**	0.00	**0.0000**	**0.0000**	–	0.0042	**0.0024**	−71.26
2	**0.7683**	**0.7683**	0.00	**0.0000**	**0.0000**	–	0.0060	**0.0059**	−1.03
3	**0.7425**	**0.7425**	0.00	**0.0000**	**0.0000**	–	**0.0034**	0.0037	8.63
4	**0.7438**	**0.7438**	0.00	**0.0000**	**0.0000**	–	**0.0067**	0.0091	25.89
5	**0.4531**	**0.4531**	0.00	**0.0000**	**0.0000**	–	**0.0043**	0.0054	20.93
6	**0.7532**	**0.7532**	0.00	**0.0000**	**0.0000**	–	**0.0085**	0.0116	26.73
7	**0.7460**	**0.7460**	0.00	**0.0000**	**0.0000**	–	**0.0035**	0.0059	39.66
8	**0.7745**	**0.7745**	0.00	**0.0013**	0.0014	0.79	**0.0082**	0.0098	16.12
9	**0.7336**	0.7317	0.27	**0.0205**	0.0208	1.50	**0.0045**	0.0046	3.58
10	**0.7514**	**0.7514**	0.00	**0.0003**	0.0004	20.00	**0.0118**	0.0137	14.24
11	0.8013	**0.8017**	−0.05	0.0009	**0.0004**	−163.64	**0.0055**	0.0077	28.42
12	**0.8212**	0.8212	0.00	**0.0024**	0.0025	2.41	**0.0102**	0.0148	30.89
13	**0.7259**	0.7256	0.04	**0.0000**	0.0004	100.00	0.0032	**0.0031**	−2.97
14	**0.8139**	0.8137	0.03	**0.0009**	0.0012	23.05	**0.0095**	0.0104	8.71
15	**0.7563**	0.7559	0.05	**0.0081**	0.0087	7.39	**0.0051**	0.0054	6.28
16	**0.7722**	0.7722	0.00	**0.0013**	0.0013	0.80	**0.0106**	0.0153	30.75
17	**0.8077**	**0.8077**	0.00	**0.0000**	**0.0000**	–	**0.0058**	0.0079	26.62
18	**0.8485**	0.8484	0.01	**0.0027**	0.0029	5.33	**0.0135**	0.0193	30.12
19	**0.6900**	**0.6900**	0.00	**0.0000**	**0.0000**	–	0.0039	**0.0038**	−2.73
20	**0.7533**	0.7533	0.00	**0.0016**	0.0016	2.93	**0.0104**	0.0104	0.30
21	0.7508	**0.7509**	−0.01	0.0012	**0.0011**	−9.50	0.0063	**0.0060**	−5.09
22	**0.7989**	0.7987	0.03	**0.0005**	0.0009	42.77	**0.0164**	0.0196	16.40
23	0.7063	**0.7064**	−0.01	**0.0006**	**0.0006**	0.00	**0.0072**	0.0082	12.19
24	**0.8026**	0.8022	0.05	**0.0042**	0.0048	12.39	**0.0148**	0.0185	20.12
25	**0.5789**	0.5785	0.06	**0.0006**	0.0011	42.54	0.0051	**0.0042**	−19.62
26	**0.7294**	**0.7294**	0.00	**0.0000**	**0.0000**	–	**0.0080**	0.0084	4.99
27	**0.7455**	0.7455	0.00	**0.0005**	0.0005	3.45	**0.0068**	0.0070	3.36
28	0.7967	**0.7968**	−0.01	0.0002	**0.0001**	−60.00	**0.0125**	0.0143	12.50
29	**0.7706**	0.7703	0.04	**0.0037**	0.0038	1.63	**0.0069**	0.0075	7.60
30	0.8402	**0.8403**	−0.01	0.0018	**0.0016**	−13.58	**0.0192**	0.0217	11.29
Avg.	**0.7491**	0.7490	0.02	**0.0018**	0.0019	1.01	**0.0081**	0.0095	10.12
Std dev.	**0.0764**	0.0764	0.05	**0.0039**	0.0040	48.92	**0.0041**	0.0054	20.36
N. best	13	5		15	4		24	6	
N. draw	12			11			0		
p	0.0642			0.0442			0.0001		

erate high-quality Pareto fronts while preserving uniformity in the distribution of non-dominated points.

We now compare EHS with the previous heuristics proposed by Wang et al. (2018), i.e., the constructive heuristic named CH and NSGA-III (Deb & Jain, 2013), initialized with CH. In this case, the comparison is not possible for all the instances 1–90. In fact, as also pointed out in (Anghinolfi et al., 2021, pages 419 and 429), the original CH heuristic by Wang et al. (2018) may be unable to build feasible schedules, even if a feasible schedule exists, when no location including only free adjacent slots is available for a job due to previous assignments.

Table 5 reports the results in terms of Hypervolume and IGD$^+$ only for the instances such that CH is able to compute a feasible solution. The computational times of NSGA-III include the time needed to perform initialization, i.e., to run CH. The percentage deviation Δ_{EHS}^{best} is the deviation of the best heuristic between CH and NSGA-III with respect to EHS. The *p*-values in the columns related to NSGA-III are obtained by comparing EHS with NSGA-III, while the *p*-values in the columns for CH refer to the comparison between EHS and CH. The results highlight that CH is unable to find a feasible solution for most of the large instances 61–90. Overall, CH can find a feasible solution for only 48 out of 90 instances. For such instances, the superiority of EHS with respect to the other heuristics is evident, both in terms of accuracy and computational times. In more detail, EHS achieves the best result on 47, 46, and 45 instances (out of 48 instances) in terms of Hypervolume, IGD$^+$, and computational time, respectively.

6.5. The exact algorithm and EHS

This subsection first presents an evaluation of the computational advantage introduced by the use of EHS to provide an initial solution for the MILP solver used by the implementation of the exact algorithm (Algorithm 5.1) based on Formulation 2. Furthermore, this subsection compares the exact algorithm with EHS to measure the trade-off between the quality of solutions and computational requirements.

Table 6 shows the values of the computational times obtained for instances 1–90 by the exact algorithm based on Formulation 2 with and without providing the solution computed by EHS as an initial solution to the MILP solver. In the table, F2 denotes the results obtained by using the exact algorithm using Formulation 2, while F2-init denotes the results obtained with the exact algorithm based on Formulation 2 when performing initialization with EHS. The table adopts the same conventions used for Tables 1–4. In particular, $\Delta_{F2\text{-init}}^{F2}$ indicates the percentage deviation of the computational times of the exact algorithm exploiting the initial solution provided by EHS with respect to the implementation of the exact algorithm without initialization. We observe that the times reported for the latter implementation include the computational overhead due to the execution of EHS for initialization. Table 6 highlights that, differently from the implementation of the exact algorithm with Formulation 1, the memory requirements of Formulation 2 enable to solve the large-scale instances 61–90 as well (we recall that Formulation 1 was able to find a solution only for the first 60 instances, as pointed out in Section 6.3). However,

Table 3
Comparison of the results obtained by EHS and SGS-ES for instances 31–60. The table shows the values of the Hypervolume, IGD^+, and computational times achieved by the two heuristics, along with the related percentage deviations.

Instance	Hypervolume			IGD^+			CPU time (s)		
	EHS	SGS-ES	Δ_{EHS}^{SGS-ES} (%)	EHS	SGS-ES	Δ_{EHS}^{SGS-ES} (%)	EHS	SGS-ES	Δ_{EHS}^{SGS-ES} (%)
31	**0.8816**	0.8813	0.03	**0.0017**	0.0023	26.67	**0.0243**	0.0384	36.88
32	**0.8116**	0.8109	0.09	**0.0007**	0.0016	59.20	**0.0383**	0.0551	30.42
33	0.7766	**0.7769**	−0.03	0.0012	**0.0009**	−22.41	**0.0563**	0.0767	26.67
34	0.6915	**0.6916**	−0.03	0.0103	**0.0102**	−1.17	**0.0878**	0.0946	7.17
35	**0.6011**	0.6011	0.01	**0.0000**	0.0001	86.67	0.0510	**0.0483**	−5.72
36	**0.8545**	0.8541	0.05	**0.0021**	0.0033	37.30	**0.0299**	0.0641	53.27
37	0.8709	**0.8710**	−0.01	**0.0033**	0.0034	4.18	**0.0477**	0.0914	47.77
38	0.8393	**0.8397**	−0.05	0.0071	**0.0066**	−6.42	**0.0768**	0.1351	43.15
39	**0.7702**	0.7700	0.02	**0.0026**	0.0028	6.80	**0.1072**	0.2147	50.05
40	**0.7762**	0.7730	0.42	**0.0056**	0.0083	32.88	**0.1778**	0.2913	38.96
41	0.9259	**0.9259**	−0.01	0.0026	**0.0024**	−8.47	**0.0319**	0.0651	50.98
42	0.8528	**0.8529**	0.00	0.0013	**0.0012**	−4.73	**0.0474**	0.1035	54.17
43	0.8613	**0.8614**	−0.01	**0.0016**	0.0020	16.97	**0.0814**	0.1566	48.04
44	**0.8229**	0.8223	0.07	**0.0050**	0.0056	10.51	**0.1221**	0.2399	49.10
45	**0.7904**	0.7899	0.07	**0.0034**	0.0040	16.40	**0.1993**	0.4059	50.91
46	**0.8166**	0.8162	0.05	0.0096	**0.0093**	−3.22	**0.1764**	0.3468	49.12
47	0.8493	**0.8494**	0.00	**0.0041**	0.0044	6.57	**0.2868**	0.6161	53.46
48	**0.8742**	0.8725	0.19	**0.0026**	0.0040	35.48	**0.4227**	0.8131	48.01
49	**0.8039**	0.8020	0.23	**0.0023**	0.0034	33.22	**0.6800**	1.1943	43.06
50	**0.8222**	0.8158	0.78	**0.0063**	0.0109	42.01	**0.8537**	1.5257	44.05
51	**0.8377**	0.8370	0.09	**0.0139**	0.0152	8.88	**0.2841**	0.6305	54.94
52	**0.8667**	0.8656	0.12	**0.0038**	0.0051	25.71	**0.3933**	0.9671	59.33
53	**0.8880**	0.8875	0.06	**0.0038**	0.0045	15.90	**0.5025**	1.3249	62.07
54	**0.8836**	0.8836	0.00	**0.0024**	0.0024	2.62	**0.6990**	1.8315	61.83
55	**0.8450**	0.8445	0.07	**0.0004**	0.0009	51.02	**0.9189**	2.8734	68.02
56	**0.8831**	0.8824	0.08	**0.0042**	0.0053	20.25	**0.3502**	0.9294	62.32
57	**0.7440**	0.7432	0.11	**0.0111**	0.0122	9.14	**0.4964**	1.4716	66.27
58	**0.9058**	0.9057	0.01	**0.0008**	0.0012	27.27	**0.6180**	2.0238	69.46
59	**0.8893**	0.8880	0.14	**0.0033**	0.0040	16.45	**0.9788**	3.8813	74.78
60	**0.8173**	0.8169	0.04	**0.0029**	0.0034	13.13	**1.0671**	3.6342	70.64
Avg.	**0.8285**	0.8277	0.09	**0.0040**	0.0047	18.63	**0.3302**	0.8715	48.97
Std dev.	**0.0668**	0.0668	0.16	**0.0034**	0.0037	22.46	**0.3223**	1.0614	17.49
N. best	22	8		24	6		29	1	
N. draw	0			0			0		
p	0.0004			0.0001			1.9209E-06		

Table 4
Comparison of the results obtained by EHS and SGS-ES for instances 61–90. The table shows the values of the Hypervolume, IGD^+, and computational times achieved by the two heuristics, along with the related percentage deviations.

Instance	Hypervolume			IGD^+			CPU time (s)		
	EHS	SGS-ES	Δ_{EHS}^{SGS-ES} (%)	EHS	SGS-ES	Δ_{EHS}^{SGS-ES} (%)	EHS	SGS-ES	Δ_{EHS}^{SGS-ES} (%)
61	**0.8178**	0.8162	0.19	**0.0039**	0.0050	21.31	**8.0704**	18.6766	56.79
62	**0.8272**	0.8262	0.13	**0.0034**	0.0040	16.76	**16.0995**	53.8304	70.09
63	**0.7660**	0.7645	0.20	**0.0036**	0.0046	21.43	**10.9416**	22.1941	50.70
64	**0.7842**	0.7827	0.19	**0.0035**	0.0044	19.77	**17.9779**	49.0354	63.34
65	**0.7221**	0.7205	0.22	**0.0064**	0.0072	11.37	**13.6809**	25.6876	46.74
66	**0.8008**	0.7984	0.30	**0.0045**	0.0060	24.54	**22.1340**	65.3891	66.15
67	**0.7558**	0.7549	0.12	**0.0028**	0.0033	15.12	**11.7066**	22.0816	46.98
68	**0.7388**	0.7372	0.22	**0.0045**	0.0054	16.19	**28.1576**	72.8456	61.35
69	**0.7369**	0.7356	0.17	**0.0043**	0.0047	9.62	**14.0772**	22.0728	36.22
70	**0.7481**	0.7450	0.41	**0.0065**	0.0087	24.61	**37.2852**	77.6561	51.99
71	**0.7997**	0.7980	0.21	**0.0046**	0.0055	16.53	**8.1154**	21.7973	62.77
72	**0.8724**	0.8703	0.24	**0.0031**	0.0044	29.50	**14.4530**	56.3769	74.36
73	**0.7929**	0.7913	0.20	**0.0042**	0.0051	16.68	**10.9099**	26.4831	58.80
74	**0.8486**	0.8471	0.17	**0.0030**	0.0039	23.90	**18.1652**	67.1380	72.94
75	**0.7850**	0.7835	0.19	**0.0033**	0.0042	20.50	**11.5784**	30.1399	61.58
76	**0.8503**	0.8488	0.18	**0.0037**	0.0047	20.53	**20.8766**	79.8189	73.85
77	**0.7453**	0.7442	0.14	**0.0049**	0.0056	12.48	**14.0398**	31.9174	56.01
78	**0.7884**	0.7866	0.23	**0.0041**	0.0051	20.47	**31.7094**	92.9936	65.90
79	**0.7472**	0.7451	0.28	**0.0039**	0.0052	25.92	**20.5086**	40.2177	49.01
80	**0.8133**	0.8117	0.19	**0.0031**	0.0041	23.92	**30.8912**	86.7287	64.38
81	**0.8158**	0.8140	0.23	**0.0047**	0.0058	19.43	**9.2991**	32.5687	71.45
82	**0.8630**	0.8615	0.18	**0.0043**	0.0055	22.14	**17.8626**	79.0559	77.41
83	**0.8174**	0.8159	0.18	**0.0038**	0.0048	21.92	**13.1299**	36.6585	64.18
84	**0.8270**	0.8261	0.11	**0.0032**	0.0039	16.53	**21.7891**	97.6430	77.68
85	**0.8210**	0.8200	0.12	**0.0035**	0.0040	12.59	**14.7378**	42.2191	65.09
86	**0.8283**	0.8272	0.14	**0.0039**	0.0045	14.18	**24.7579**	111.6060	77.82
87	**0.8054**	0.8040	0.17	**0.0042**	0.0049	13.36	**16.2661**	45.2810	64.08
88	**0.8364**	0.8351	0.16	**0.0032**	0.0041	21.61	**33.0019**	128.1692	74.25
89	**0.7622**	0.7588	0.44	**0.0051**	0.0076	32.06	**23.8458**	63.2728	62.31
90	**0.8080**	0.8068	0.15	**0.0042**	0.0049	14.24	**45.2837**	136.8648	66.91
Avg.	**0.7975**	0.7959	0.20	**0.0040**	0.0050	19.31	**19.3784**	57.8807	63.04
Std dev.	**0.0401**	0.0402	0.08	**0.0009**	0.0011	5.36	**9.1422**	32.9438	10.33
N. best	30	0		30	0		30	0	
N. draw	0			0			0		
p	1.7344E-06			1.7344E-06			1.7344E-06		

Table 5
Comparison of the results obtained by EHS and NSGA-III and CH for all the instances such that CH returns a feasible solution. The table shows the values of the Hypervolume, IGD^+, and computational times achieved by the three approaches, along with the related percentage deviations computed with respect to the best competitor of EHS.

Instance	Hypervolume				IGD^+				CPU time (s)			
	EHS	NSGA-III	CH	Δ_{EHS}^{best} (%)	EHS	NSGA-III	CH	Δ_{EHS}^{best} (%)	EHS	NSGA-III	CH	Δ_{EHS}^{best} (%)
1	**0.6976**	0.6962	0.6906	0.21	**0.0000**	0.0023	0.0066	100.00	**0.0042**	10.9688	0.0206	79.77
2	**0.7683**	0.7683	0.7671	0.01	**0.0000**	0.0001	0.0020	100.00	**0.0060**	10.9884	0.0215	72.13
3	0.7425	0.7425	**0.7425**	0.00	**0.0000**	**0.0000**	**0.0000**	–	**0.0034**	10.8791	0.0099	65.75
7	**0.7460**	0.7452	0.7449	0.11	**0.0000**	0.0007	0.0011	100.00	**0.0035**	10.6625	0.0082	56.87
10	**0.7514**	0.7510	0.7506	0.05	**0.0003**	0.0005	0.0010	40.00	**0.0118**	10.9952	0.0312	62.27
12	**0.8212**	0.8208	0.8208	0.05	0.0024	**0.0023**	0.0032	−8.00	**0.0102**	10.8785	0.0435	76.56
13	**0.7259**	0.7243	0.7243	0.22	**0.0000**	0.0020	0.0020	100.00	**0.0032**	10.8018	0.0067	51.33
14	**0.8139**	0.8121	0.8121	0.22	**0.0009**	0.0034	0.0035	72.40	**0.0095**	10.6095	0.0213	55.16
16	**0.7722**	0.7688	0.7687	0.43	**0.0013**	0.0057	0.0058	76.47	**0.0106**	10.9737	0.0396	73.26
17	**0.8077**	0.8063	0.8063	0.17	**0.0000**	0.0017	0.0017	100.00	**0.0058**	10.8916	0.0182	68.05
18	**0.8485**	0.8461	0.8461	0.28	**0.0027**	0.0052	0.0052	48.43	**0.0135**	10.8807	0.0504	73.30
19	**0.6900**	0.6862	0.6852	0.56	**0.0000**	0.0035	0.0045	100.00	**0.0039**	10.8885	0.0079	51.06
21	**0.7508**	0.7452	0.7451	0.74	**0.0012**	0.0064	0.0064	81.08	**0.0063**	10.6952	0.0122	48.31
22	**0.7989**	0.7931	0.7930	0.73	**0.0005**	0.0062	0.0063	91.95	**0.0164**	10.7534	0.0747	78.08
23	**0.7063**	0.7021	0.7021	0.60	**0.0006**	0.0057	0.0057	90.32	**0.0072**	10.9184	0.0103	29.93
25	**0.5789**	0.5725	0.5723	1.11	**0.0006**	0.0065	0.0067	90.55	**0.0051**	10.8099	0.0074	30.92
26	**0.7294**	0.7198	0.7170	1.32	**0.0000**	0.0060	0.0086	100.00	**0.0080**	10.8309	0.0700	88.56
27	**0.7455**	0.7440	0.7440	0.21	**0.0005**	0.0028	0.0028	83.53	**0.0068**	10.7334	0.0075	9.65
28	**0.7967**	0.7943	0.7943	0.31	**0.0002**	0.0027	0.0027	91.62	**0.0125**	10.8425	0.0764	83.57
31	**0.8816**	0.8715	0.8715	1.15	**0.0017**	0.0106	0.0106	84.02	**0.0243**	11.0500	0.1584	84.68
32	**0.8116**	0.7980	0.7980	1.68	**0.0007**	0.0118	0.0118	94.34	**0.0383**	11.3557	0.1010	62.05
33	**0.7766**	0.7733	0.7733	0.43	**0.0012**	0.0044	0.0044	73.68	**0.0563**	11.7489	0.0634	11.30
34	**0.6915**	0.6794	0.6793	1.75	**0.0103**	0.0212	0.0213	51.40	0.0878	12.3398	**0.0622**	−41.16
35	**0.6011**	0.5875	0.5875	2.27	**0.0000**	0.0150	0.0150	99.95	**0.0510**	13.1657	0.0578	11.62
36	**0.8545**	0.8437	0.8434	1.26	**0.0021**	0.0072	0.0074	70.85	**0.0299**	11.4679	0.1026	70.81
37	**0.8709**	0.8658	0.8658	0.58	**0.0033**	0.0075	0.0075	56.02	**0.0477**	11.6202	0.1056	54.79
38	**0.8393**	0.8372	0.8372	0.25	**0.0071**	0.0087	0.0087	19.31	**0.0768**	12.2492	0.2738	71.96
39	**0.7702**	0.7651	0.7651	0.66	**0.0026**	0.0071	0.0071	62.80	**0.1072**	12.9709	0.1297	17.30
40	**0.7762**	0.7592	0.7592	2.19	**0.0056**	0.0194	0.0194	71.30	0.1778	13.8076	**0.1656**	−7.39
41	**0.9259**	0.9253	0.9253	0.06	**0.0026**	0.0036	0.0036	28.84	**0.0319**	11.5631	0.1355	76.46
42	**0.8528**	0.8505	0.8505	0.28	**0.0013**	0.0039	0.0039	67.44	**0.0474**	12.0115	0.1300	63.52
44	**0.8229**	0.8197	0.8196	0.39	**0.0050**	0.0077	0.0077	35.46	**0.1221**	13.1930	0.1538	20.61
45	**0.7904**	0.7800	0.7800	1.32	**0.0034**	0.0113	0.0113	70.19	0.1993	14.5432	**0.1684**	−18.35
46	**0.8166**	0.8088	0.8084	0.95	**0.0096**	0.0149	0.0152	35.45	**0.1764**	13.1718	2.0441	91.37
47	**0.8493**	0.8463	0.8463	0.36	**0.0041**	0.0062	0.0062	33.48	**0.2868**	13.2982	2.1061	86.38
48	**0.8742**	0.8654	0.8654	1.01	**0.0026**	0.0092	0.0092	71.98	**0.4227**	13.7993	2.1209	80.07
50	**0.8222**	0.8003	0.8003	2.66	**0.0063**	0.0233	0.0233	72.89	**0.8537**	15.1520	2.3195	63.20
51	**0.8377**	0.8281	0.8280	1.15	**0.0139**	0.0163	0.0174	15.04	**0.2841**	15.2817	3.9061	92.73
52	**0.8667**	0.8636	0.8635	0.35	**0.0038**	0.0071	0.0071	45.93	**0.3933**	15.8807	3.9766	90.11
53	**0.8880**	0.8847	0.8847	0.38	**0.0038**	0.0072	0.0072	47.72	**0.5025**	16.2283	4.0585	87.62
55	**0.8450**	0.8411	0.8411	0.47	**0.0004**	0.0034	0.0034	87.32	**0.9189**	17.6802	4.1856	78.05
56	**0.8831**	0.8794	0.8788	0.42	**0.0042**	0.0079	0.0086	46.15	**0.3502**	17.8474	6.1874	94.34
57	**0.7440**	0.7363	0.7353	1.03	**0.0111**	0.0152	0.0161	27.05	**0.4964**	18.0846	6.1088	91.87
58	**0.9058**	0.9046	0.9046	0.13	**0.0008**	0.0030	0.0030	72.03	**0.6180**	18.9285	6.1696	89.98
60	**0.8173**	0.8148	0.8148	0.31	**0.0029**	0.0054	0.0054	45.76	**1.0671**	20.9968	6.3643	83.23
66	**0.8008**	0.7805	0.7805	2.53	**0.0045**	0.0195	0.0195	76.84	**22.1340**	223.5118	206.0299	89.26
74	**0.8486**	0.8379	0.8379	1.26	**0.0030**	0.0105	0.0105	71.87	**18.1652**	149.1959	132.1338	86.25
86	**0.8283**	0.8157	0.8157	1.52	**0.0039**	0.0123	0.0123	68.47	**24.7579**	149.0026	129.1161	80.83
Avg.	**0.7955**	0.7896	0.7893	0.75	**0.0028**	0.0076	0.0079	67.28	**1.5140**	22.8568	10.8161	60.79
Std dev.	**0.0724**	0.0730	0.0733	0.69	**0.0032**	0.0057	0.0057	26.90	**5.3139**	40.4773	38.9220	32.12
N. best	47	0	1		46	1	0		45	0	3	
N. draw		0				1				0		
p		2.3968E-09	1.7378E-09			2.7281E-09	2.3968E-09			1.6310E-09	3.4284E-08	

the maximum time limit of 4 hours for the MILP solver was not sufficient to compute the optimal Pareto fronts for instances 62, 79, 83, 85, 86, 88, and 89. On the contrary, the initialization of the exact algorithm through EHS allows solving instances 62, 83, 85, 86, and 89 within the time limit. Moreover, such an initialization enables large time savings in 88 instances over 90, with the exception of instances 79 and 88, where the time limit is again reached.

With a little abuse of notation, in the remainder of this subsection, we simply refer to the implementation of Algorithm 5.1 based on Formulation 2 with the initialization provided by EHS as "exact algorithm with initialization". Tables 7–9 report the results of the comparison between the exact algorithm with initialization and EHS in terms of the performance metrics introduced in Section 6.2 and computational times for instances 1–30, 31–60 and 61–90 respectively. We remark that both tables adopt the conventions introduced in the previous subsections.

We first focus on Tables 7 and 8. It is interesting to observe that EHS is able to compute the optimal Pareto front for instances 1, 2, 4, 5, 6, 7, 10, 13, 17, 19, and 26. In fact, EHS and the exact algorithm achieved the same values of Hypervolume and IGD^+. For the other instances, the quality of results provided by the exact algorithm is always larger than the one given by EHS. The computational times of EHS are always less than the ones of the exact algorithm with initialization, with the minimum percentage deviation being equal to 71.14%, and an average of 97.28% and 89.47% deviation for instances 1–30 and 31–60, respectively. Similar arguments can be reported as regards the results for the large-scale instances 61–90 shown in Table 9, where the percentage deviation

Table 6
Comparison of the computational times obtained by the exact algorithm when using Formulation 2 (F2) and Formulation 2 with initialization provided by EHS (F2-init) for instances 1–30, 31–60, and 61–90.

Instance	CPU time (s)			Instance	CPU time (s)			Instance	CPU time (s)		
	F2	F2-init	$\Delta^{F2}_{F2\text{-init}}$ (%)		F2	F2-init	$\Delta^{F2}_{F2\text{-init}}$ (%)		F2	F2-init	$\Delta^{F2}_{F2\text{-init}}$ (%)
1	0.3525	**0.2150**	39.01	31	0.9499	**0.7694**	19.01	61	827.0859	**611.6478**	26.05
2	0.4799	**0.1387**	71.10	32	1.1458	**0.8805**	23.15	62	14400.0000	**1151.5676**	92.00
3	0.3041	**0.0892**	70.68	33	1.5192	**1.0644**	29.94	63	1376.1149	**1081.9362**	21.38
4	0.5265	**0.1154**	78.09	34	2.2330	**1.8488**	17.21	64	2019.4856	**1603.4638**	20.60
5	0.4288	**0.0789**	81.60	35	2.5779	**1.5583**	39.55	65	1165.4502	**1013.6229**	13.03
6	0.4466	**0.1060**	76.26	36	0.9013	**0.4714**	47.70	66	2180.1869	**1778.8279**	18.41
7	0.3865	**0.3000**	22.40	37	1.3302	**0.8897**	33.12	67	1426.0655	**1117.5993**	21.63
8	0.5586	**0.4035**	27.77	38	2.5866	**2.1766**	15.85	68	5291.3348	**4939.7465**	6.64
9	0.2809	**0.2154**	23.31	39	2.7987	**1.9215**	31.34	69	2543.0730	**2165.1171**	14.86
10	0.7209	**0.2146**	70.23	40	4.1294	**3.2607**	21.04	70	2690.2322	**2357.9327**	12.35
11	0.4399	**0.1624**	63.09	41	0.9398	**0.3561**	62.11	71	1017.8704	**815.0992**	19.92
12	0.7382	**0.3179**	56.94	42	1.7630	**0.6580**	62.68	72	906.8871	**717.3885**	20.90
13	0.4236	**0.2772**	34.56	43	2.6687	**1.9790**	25.84	73	1201.6998	**965.8806**	19.62
14	0.6709	**0.4220**	37.10	44	3.7989	**2.9698**	21.83	74	1340.3735	**1178.4611**	12.08
15	0.3676	**0.2575**	29.95	45	5.8304	**4.2192**	27.63	75	1299.6371	**1065.3111**	18.03
16	0.7276	**0.3325**	54.30	46	3.4909	**1.0783**	69.11	76	1348.7682	**1142.3735**	15.30
17	0.5931	**0.3609**	39.15	47	4.2710	**2.7043**	36.68	77	1860.7451	**1612.8176**	13.32
18	1.0311	**0.5939**	42.40	48	7.3489	**4.9430**	32.74	78	2998.1649	**2592.9869**	13.51
19	0.4866	**0.2932**	39.73	49	17.5062	**13.3570**	23.70	79	**14400.0000**	**14400.0000**	0.00
20	0.6527	**0.4862**	25.52	50	11.3853	**8.9311**	21.56	80	2741.5972	**2290.5488**	16.45
21	0.6678	**0.4605**	31.04	51	7.2481	**0.9845**	86.42	81	828.6165	**667.3496**	19.46
22	0.9895	**0.7531**	23.89	52	5.3167	**1.2420**	76.64	82	2708.3709	**1344.9350**	50.34
23	0.4635	**0.2670**	42.40	53	10.1980	**4.1180**	59.62	83	14400.0000	**1096.4566**	92.39
24	1.2782	**1.1608**	9.19	54	12.6210	**4.6798**	62.92	84	2891.7570	**1755.8605**	39.28
25	0.2998	**0.1972**	34.23	55	18.1268	**10.7669**	40.60	85	14400.0000	**1236.8127**	91.41
26	0.8721	**0.4775**	45.25	56	4.1815	**1.5448**	63.06	86	14400.0000	**1775.3635**	87.67
27	0.4519	**0.2971**	34.25	57	10.9855	**1.7778**	83.82	87	1402.1250	**1157.3959**	17.45
28	1.1910	**0.7695**	35.39	58	13.1024	**2.7290**	79.17	88	**14400.0000**	**14400.0000**	0.00
29	0.9733	**0.7052**	27.55	59	21.4852	**15.7179**	26.84	89	14400.0000	**8863.0899**	38.45
30	0.9687	**0.8249**	14.85	60	20.7640	**7.7110**	62.86	90	3606.3123	**3108.4430**	13.81
Avg.	0.6258	**0.3764**	42.71	**Avg.**	6.7735	**3.5770**	43.46	**Avg.**	4882.3985	**2666.9345**	28.21
Std dev.	0.2731	**0.2550**	19.84	**Std dev.**	6.2401	**3.9109**	22.29	**Std dev.**	5422.8772	**3563.0504**	26.98
N. best	0	30		**N. best**	0	30		**N. best**	0	28	
N. draw	0			**N. draw**	0			**N. draw**	2		
p	1.7344E-06			***p***	1.7344E-06			***p***	3.7896E-06		

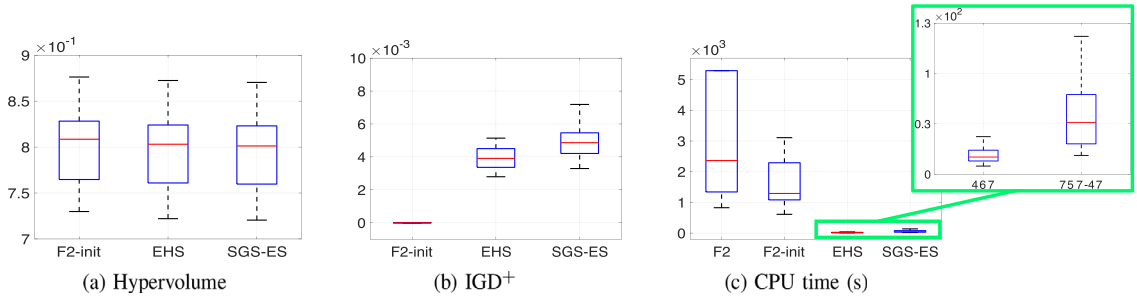

Fig. 6. Box-and-whiskers plots for the values of Hypervolume (a), IGD$^+$ (b), and computational times (c) of the solutions computed by the exact algorithm exploiting Formulation 2 with initialization provided by EHS (F2-init), EHS itself, and SGS-ES on instances 61–90. In (c), we also show the computational times of the exact algorithm exploiting Formulation 2 (F2), and we provide an enlargement of the box-and-whiskers plot of EHS and SGS-ES at a finer scale. We enhanced the readability of all the plots by omitting the outliers.

of EHS with respect to the exact algorithm with initialization is equal to 98.85% on the average and never lower than 97%. In this case, the quality of the solutions of the exact algorithm is always better than the one of EHS, except for instances 79 and 88. In fact, the exact algorithm could not compute the optimal Pareto front for such two instances within the 4 hours time limit. This motivates also the better average values of EHS with respect to F2-init, even if the remaining 28 instances are characterized by the superiority of F2-init. However, for the other instances, the values of Hypervolume and IGD$^+$ are equal up to the second decimal place, hence proving the effectiveness of EHS in finding a solution.

Overall, we can conclude that EHS constitutes an excellent trade-off between the quality of solutions and computational effort. Furthermore, the solutions computed by EHS provide a significant speed-up to the exact algorithm, which is able to compute the optimal solution to the considered instances within a time that is only two orders of magnitude greater than the one required by EHS. Finally, we observe that all the aforementioned considerations are always supported by the small p-values, which confirm the statistical difference between the results provided by the two approaches.

Figs. 6 and 7 provide visual support to our discussion. The figures also display the results achieved by SGS-ES so as to provide a more comprehensive evaluation of the novel solution approaches. Specifically, Fig. 6 provides the box-and-whiskers plots for Hypervolume (a), IGD$^+$ (b), and computational times (c) of the solu-

Table 7
Comparison of the results obtained by the exact algorithm when using Formulation 2 with initialization provided by EHS (denoted by F2-init) and EHS itself for instances 1–30. The table shows the values of the Hypervolume, IGD$^+$, and computational times achieved by the two approaches, along with the related percentage deviations.

Instance	Hypervolume			IGD$^+$			CPU time (s)		
	F2-init	EHS	$\Delta^{EHS}_{F2\text{-}init}$ (%)	F2-init	EHS	$\Delta^{EHS}_{F2\text{-}init}$ (%)	F2-init	EHS	$\Delta^{F2\text{-}init}_{EHS}$ (%)
1	**0.6976**	**0.6976**	0.00	**0.0000**	**0.0000**	–	0.2150	**0.0042**	98.06
2	**0.7683**	**0.7683**	0.00	**0.0000**	**0.0000**	–	0.1387	**0.0060**	95.68
3	**0.7425**	0.7425	0.00	**0.0000**	**0.0000**	–	0.0892	**0.0034**	96.19
4	**0.7438**	**0.7438**	0.00	**0.0000**	**0.0000**	–	0.1154	**0.0067**	94.16
5	**0.4531**	**0.4531**	0.00	**0.0000**	**0.0000**	–	0.0789	**0.0043**	94.56
6	**0.7532**	**0.7532**	0.00	**0.0000**	**0.0000**	–	0.1060	**0.0085**	91.99
7	**0.7460**	**0.7460**	0.00	**0.0000**	**0.0000**	–	0.3000	**0.0035**	98.82
8	**0.7756**	0.7745	0.14	**0.0000**	0.0013	100.00	0.4035	**0.0082**	97.96
9	**0.7507**	0.7336	2.27	**0.0000**	0.0205	100.00	0.2154	**0.0045**	97.93
10	**0.7514**	**0.7514**	0.00	**0.0000**	0.0003	100.00	0.2146	**0.0118**	94.52
11	**0.8019**	0.8013	0.08	**0.0000**	0.0009	100.00	0.1624	**0.0055**	96.62
12	**0.8220**	0.8212	0.10	**0.0000**	0.0024	100.00	0.3179	**0.0102**	96.79
13	**0.7259**	**0.7259**	0.00	**0.0000**	**0.0000**	–	0.2772	**0.0032**	98.83
14	**0.8153**	0.8139	0.17	**0.0000**	0.0009	100.00	0.4220	**0.0095**	97.74
15	**0.7687**	0.7563	1.61	**0.0000**	0.0081	100.00	0.2575	**0.0051**	98.04
16	**0.7732**	0.7722	0.13	**0.0000**	0.0013	100.00	0.3325	**0.0106**	96.82
17	**0.8077**	**0.8077**	0.00	**0.0000**	**0.0000**	–	0.3609	**0.0058**	98.39
18	**0.8506**	0.8485	0.25	**0.0000**	0.0027	100.00	0.5939	**0.0135**	97.73
19	**0.6900**	**0.6900**	0.00	**0.0000**	**0.0000**	–	0.2932	**0.0039**	98.68
20	**0.7548**	0.7533	0.19	**0.0000**	0.0016	100.00	0.4862	**0.0104**	97.86
21	**0.7520**	0.7508	0.16	**0.0000**	0.0012	100.00	0.4605	**0.0063**	98.64
22	**0.7993**	0.7989	0.05	**0.0000**	0.0005	100.00	0.7531	**0.0164**	97.83
23	**0.7067**	0.7063	0.05	**0.0000**	0.0006	100.00	0.2670	**0.0072**	97.30
24	**0.8067**	0.8026	0.52	**0.0000**	0.0042	100.00	1.1608	**0.0148**	98.73
25	**0.5794**	0.5789	0.08	**0.0000**	0.0006	100.00	0.1972	**0.0051**	97.42
26	**0.7294**	**0.7294**	0.00	**0.0000**	**0.0000**	–	0.4775	**0.0080**	98.32
27	**0.7458**	0.7455	0.04	**0.0000**	0.0005	100.00	0.2971	**0.0068**	97.72
28	**0.7970**	0.7967	0.03	**0.0000**	0.0002	100.00	0.7695	**0.0125**	98.37
29	**0.7743**	0.7706	0.48	**0.0000**	0.0037	100.00	0.7052	**0.0069**	99.02
30	**0.8413**	0.8402	0.13	**0.0000**	0.0018	100.00	0.8249	**0.0192**	97.67
Avg.	**0.7508**	0.7503	0.22	**0.0000**	0.0018	100.00	0.3764	**0.0081**	97.28
Std dev.	0.0768	**0.0767**	0.49	**0.0000**	0.0039	0.00	0.2550	**0.0041**	1.64
N. best	19	0		19	0		0	30	
N. draw	11			11			0		
p	1.3183E-04			1.3183E-04			1.7344E-06		

Table 8
Comparison of the results obtained by the exact algorithm when using Formulation 2 with initialization provided by EHS (denoted by F2-init) and EHS itself for instances 31–60. The table shows the values of the Hypervolume, IGD$^+$, and computational times achieved by the two approaches, along with the related percentage deviations.

Instance	Hypervolume			IGD$^+$			CPU time (s)		
	F2-init	EHS	$\Delta^{EHS}_{F2\text{-}init}$ (%)	F2-init	EHS	$\Delta^{EHS}_{F2\text{-}init}$ (%)	F2-init	EHS	$\Delta^{F2\text{-}init}_{EHS}$ (%)
31	**0.8829**	0.8816	0.15	**0.0000**	0.0017	100.00	0.7694	**0.0243**	96.85
32	**0.8122**	0.8116	0.08	**0.0000**	0.0007	100.00	0.8805	**0.0383**	95.65
33	**0.7783**	0.7766	0.21	**0.0000**	0.0012	100.00	1.0644	**0.0563**	94.71
34	**0.7021**	0.6915	1.52	**0.0000**	0.0103	100.00	1.8488	**0.0878**	95.25
35	**0.6012**	0.6011	0.00	**0.0000**	0.0000	100.00	1.5583	**0.0510**	96.72
36	**0.8563**	0.8545	0.21	**0.0000**	0.0021	100.00	0.4714	**0.0299**	93.65
37	**0.8731**	0.8709	0.26	**0.0000**	0.0033	100.00	0.8897	**0.0477**	94.64
38	**0.8464**	0.8393	0.84	**0.0000**	0.0071	100.00	2.1766	**0.0768**	96.47
39	**0.7733**	0.7702	0.40	**0.0000**	0.0026	100.00	1.9215	**0.1072**	94.42
40	**0.7823**	0.7762	0.77	**0.0000**	0.0056	100.00	3.2607	**0.1778**	94.55
41	**0.9267**	0.9259	0.08	**0.0000**	0.0026	100.00	0.3561	**0.0319**	91.04
42	**0.8535**	0.8528	0.08	**0.0000**	0.0013	100.00	0.6580	**0.0474**	92.79
43	**0.8630**	0.8613	0.19	**0.0000**	0.0016	100.00	1.9790	**0.0814**	95.89
44	**0.8279**	0.8229	0.61	**0.0000**	0.0050	100.00	2.9698	**0.1221**	95.89
45	**0.7936**	0.7904	0.40	**0.0000**	0.0034	100.00	4.2192	**0.1993**	95.28
46	**0.8266**	0.8166	1.21	**0.0000**	0.0096	100.00	1.0783	**0.1764**	83.64
47	**0.8523**	0.8493	0.34	**0.0000**	0.0041	100.00	2.7043	**0.2868**	89.40
48	**0.8760**	0.8742	0.21	**0.0000**	0.0026	100.00	4.9430	**0.4227**	91.45
49	**0.8064**	0.8039	0.31	**0.0000**	0.0023	100.00	13.3570	**0.6800**	94.91
50	**0.8320**	0.8222	1.19	**0.0000**	0.0063	100.00	8.9311	**0.8537**	90.44
51	**0.8459**	0.8377	0.97	**0.0000**	0.0139	100.00	0.9845	**0.2841**	71.14
52	**0.8681**	0.8667	0.17	**0.0000**	0.0038	100.00	1.2420	**0.3933**	68.33
53	**0.8903**	0.8880	0.25	**0.0000**	0.0038	100.00	4.1180	**0.5025**	87.80
54	**0.8848**	0.8836	0.14	**0.0000**	0.0024	100.00	4.6798	**0.6990**	85.06
55	**0.8454**	0.8450	0.04	**0.0000**	0.0004	100.00	10.7669	**0.9189**	91.47
56	**0.8852**	0.8831	0.23	**0.0000**	0.0042	100.00	1.5448	**0.3502**	77.33
57	**0.7517**	0.7440	1.02	**0.0000**	0.0111	100.00	1.7778	**0.4964**	72.08
58	**0.9060**	0.9058	0.02	**0.0000**	0.0008	100.00	2.7290	**0.6180**	77.35
59	**0.8921**	0.8893	0.31	**0.0000**	0.0033	100.00	15.7179	**0.9788**	93.77
60	**0.8198**	0.8173	0.31	**0.0000**	0.0029	100.00	7.7110	**1.0671**	86.16
Avg.	**0.8318**	0.7503	0.42	**0.0000**	0.0040	100.00	3.5770	**0.3302**	89.47
Std dev.	**0.0658**	0.0767	0.41	**0.0000**	0.0034	0.00	3.9109	**0.3223**	8.27
N. best	30	0		30	0		0	30	
N. draw	0			0			0		
p	1.7344E-06			1.7344E-06			1.7344E-06		

Table 9
Comparison of the results obtained by the exact algorithm when using Formulation 2 with initialization provided by EHS (denoted by F2-init) and EHS itself for instances 61–90. The table shows the values of the Hypervolume, IGD$^+$, and computational times achieved by the two approaches, along with the related percentage deviations.

Instance	Hypervolume			IGD$^+$			CPU time (s)		
	F2-init	EHS	$\Delta^{EHS}_{F2\text{-}init}$ (%)	F2-init	EHS	$\Delta^{EHS}_{F2\text{-}init}$ (%)	F2-init	EHS	$\Delta^{F2\text{-}init}_{EHS}$ (%)
61	**0.8223**	0.8178	0.55	**0.0000**	0.0039	100.00	611.6478	**8.0704**	98.68
62	**0.8313**	0.8272	0.50	**0.0000**	0.0034	100.00	1151.5676	**16.0995**	98.60
63	**0.7704**	0.7660	0.57	**0.0000**	0.0036	100.00	1081.9362	**10.9416**	98.99
64	**0.7886**	0.7842	0.56	**0.0000**	0.0035	100.00	1603.4638	**17.9779**	98.88
65	**0.7299**	0.7221	1.06	**0.0000**	0.0064	100.00	1013.6229	**13.6809**	98.65
66	**0.8064**	0.8008	0.70	**0.0000**	0.0045	100.00	1778.8279	**22.1340**	98.76
67	**0.7590**	0.7558	0.41	**0.0000**	0.0028	100.00	1117.5993	**11.7066**	98.95
68	**0.7446**	0.7388	0.77	**0.0000**	0.0045	100.00	4939.7465	**28.1576**	99.43
69	**0.7418**	0.7369	0.66	**0.0000**	0.0043	100.00	2165.1171	**14.0772**	99.35
70	**0.7563**	0.7481	1.09	**0.0000**	0.0065	100.00	2357.9327	**37.2852**	98.42
71	**0.8052**	0.7997	0.69	**0.0000**	0.0046	100.00	815.0992	**8.1154**	99.00
72	**0.8762**	0.8724	0.43	**0.0000**	0.0031	100.00	717.3885	**14.4530**	97.99
73	**0.7982**	0.7929	0.66	**0.0000**	0.0042	100.00	965.8806	**10.9099**	98.87
74	**0.8522**	0.8486	0.42	**0.0000**	0.0030	100.00	1178.4611	**18.1652**	98.46
75	**0.7890**	0.7850	0.50	**0.0000**	0.0033	100.00	1065.3111	**11.5784**	98.91
76	**0.8550**	0.8503	0.55	**0.0000**	0.0037	100.00	1142.3735	**20.8766**	98.17
77	**0.7514**	0.7453	0.81	**0.0000**	0.0049	100.00	1612.8176	**14.0398**	99.13
78	**0.7935**	0.7884	0.64	**0.0000**	0.0041	100.00	2592.9869	**31.7094**	98.78
79	0.3657	**0.7472**	−104.34	0.1910	**0.0039**	−4849.50	14400.0000	**20.5086**	99.86
80	**0.8171**	0.8133	0.47	**0.0000**	0.0031	100.00	2290.5488	**30.8912**	98.65
81	**0.8217**	0.8158	0.72	**0.0000**	0.0047	100.00	667.3496	**9.2991**	98.61
82	**0.8683**	0.8630	0.61	**0.0000**	0.0043	100.00	1344.9350	**17.8626**	98.67
83	**0.8221**	0.8174	0.57	**0.0000**	0.0038	100.00	1096.4566	**13.1299**	98.80
84	**0.8314**	0.8270	0.53	**0.0000**	0.0032	100.00	1755.8605	**21.7891**	98.76
85	**0.8250**	0.8210	0.49	**0.0000**	0.0035	100.00	1236.8127	**14.7378**	98.81
86	**0.8333**	0.8283	0.60	**0.0000**	0.0039	100.00	1775.3635	**24.7579**	98.61
87	**0.8106**	0.8054	0.64	**0.0000**	0.0042	100.00	1157.3959	**16.2661**	98.59
88	0.7039	**0.8364**	−18.82	0.0375	**0.0032**	−1062.61	14400.0000	**33.0019**	99.77
89	**0.7683**	0.7622	0.80	**0.0000**	0.0051	100.00	8863.0899	**23.8458**	99.73
90	**0.8131**	0.8080	0.63	**0.0000**	0.0042	100.00	3108.4430	**45.2837**	98.54
Avg.	**0.7851**	0.7503	−3.52	0.0076	**0.0040**	−103.74	2666.9345	**45.2837**	98.85
Std dev.	0.0896	**0.0767**	19.37	0.0353	**0.0009**	921.09	3563.0504	**9.1422**	0.43
N. best	28	2		28	2		0	30	
N. draw	0			0			0		
p	3.5888E-04			3.5888E-04			1.7344E-06		

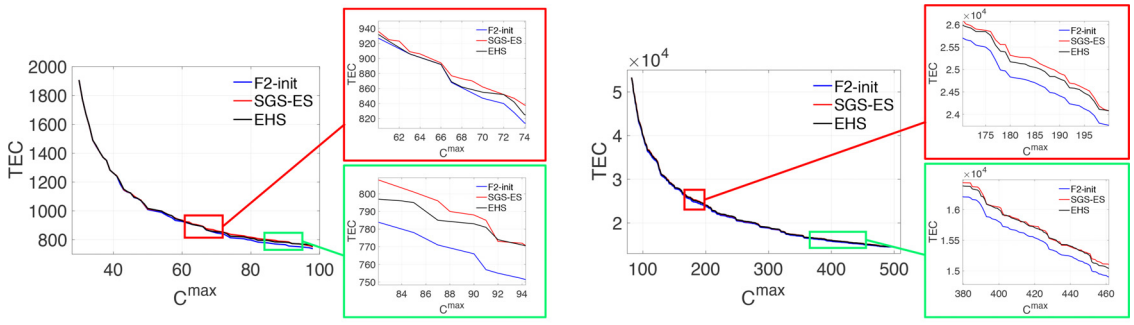

Fig. 7. Pareto fronts of instance 40 (a) and instance 90 (b) computed by the exact algorithm exploiting Formulation 2 with initialization provided by EHS (F2-init), EHS itself, and SGS-ES. Enlargements of two different parts of the fronts are also reported to appreciate the difference between the various algorithms at a finer scale.

tions computed by the exact algorithm with initialization, EHS itself and SGS-ES on the large-scale instances, i.e., 61–90. In particular, Fig. 6(c) also displays the computational times obtained by the exact algorithm. The plots are consistent with the conclusions drawn in this section on the comparisons between the exact algorithm exploiting Formulation 2 (with and without initialization) and EHS, as well as EHS itself and SGS-ES (see Subsection 6.4), according to the numerical results presented in the tables. Fig. 6(c) magnifies the box-and-whiskers plots related to EHS and SGS-ES so as to provide a more accurate comparison of their computational times on a different scale. Finally, Fig. 7 provides a graphical representation of the Pareto fronts computed by the exact algorithm with initialization and EHS for instances 40 (a) and 90 (b). Enlargements of different portions of the figure allow the reader to appreciate the differences between the displayed Pareto fronts with a finer level of detail.

7. Conclusions

Energy-efficient manufacturing has become a compelling matter in the latest years, owing to the pressing environmental issues and the consequent desire to shift toward sustainable production. In this context, the development of suitable scheduling models and efficient solution approaches plays an essential role in the definition of a new paradigm that encompasses both productivity goals and environmental awareness.

In this paper, we have considered the bi-objective scheduling problem of simultaneously minimizing the makespan and the TEC

of a set of independent jobs on parallel identical machines, called the Bi-objective scheduling on parallel identical machines with time-of-use costs problem (BPMSTP). We have introduced a combinatorial property that has provided a novel description of the solution space of the BPMSTP. This characterizing property has enabled the development of a compact formulation that constitutes the foundation of an exact algorithm for the problem. Such an algorithm significantly improves upon the previous exact approaches, as it has reduced the computational times by two orders of magnitude for several instances in the test dataset. We have also presented a novel heuristic for the BPMSTP, called Enhanced heuristic scheduler (EHS). Such a heuristic is based on several ideas that enable an accurate and fast resolution of both small and large instances. We have shown that EHS outperforms the previous state-of-the-art heuristic for the BPMSTP, called Split-greedy scheduler with exchange search (SGS-ES). Furthermore, EHS has also proved able to speed up the exact algorithm, by further reducing its computational burden.

Summarizing, the novelty of this paper constitutes a significant step in the development of effective solution approaches, both exact and heuristic, for the BPMSTP. This work will hopefully enable additional research on identical parallel machine scheduling under TOU costs in the future, by encouraging the design of simpler and/or faster approaches. In this respect, as future research directions, we plan to address the generalization of the combinatorial properties of the BPMSTP to similar problems. For instance, classical objectives in scheduling, such as the total weighted tardiness or the maximum lateness, could be considered along with the makespan or the TEC in order to increase the capability of the problem to model real manufacturing systems. From a mathematical standpoint, we will consider different representations of the time horizon, so as to possibly develop continuous-time or sequence-based formulations that are ubiquitous in classical scheduling. From the computational viewpoint, we will investigate the possibility of speeding up the exact algorithm by tackling the sequential single-objective problems through parallel computing, so as to fully exploit the available CPU and memory resources. Finally, we will focus on extending the local search used by EHS to consider larger sets of improving moves, which may further enhance the quality of solutions at the expense of slightly increasing computational times.

Acknowledgment

The authors are in debt with Prof. Raffaele Pesenti for his useful comments on the first version of the manuscript. The authors are also grateful to the anonymous referees for their valuable suggestions that allowed us to significantly improve the manuscript.

Appendix A. Test instances

In this Appendix, we report the details of the set of test instances used in Section 6. For each instance, the number of jobs N, the number of machines M, and the number of time slots K is given by an element (N, M, K) of the Cartesian product between

- $\{6, 10, 15, 20, 25\}$, $\{3, 5, 7\}$, and $\{50, 80\}$ for instances 1–30;
- $\{30, 60, 100, 150, 200\}$, $\{8, 16, 25\}$, and $\{100, 300\}$ for instances 31–60, except for instances 41–44, where the number of machines is equal to 20 instead of 25;
- $\{250, 300, 350, 400, 500\}$, $\{25, 30, 40\}$, and $\{350, 500\}$ for instances 61–90.

The processing times $p_j \in \mathbb{Z}_+$, for each j in the set of jobs $\mathcal{J}$, were randomly drawn from the uniform distributions $U[1, 4]$, $U[1, 4]$, and $U[1, 12]$ for instances 1–30, 31–60, and 61–90, respectively. Similarly, the consumption rates $u_h \in \mathbb{Z}_+$, for each h in the set of machines $\mathcal{H}$, were drawn from $U[1, 3]$, $U[1, 3]$, and $U[1, 6]$ for instances 1–30, 31–60, and 61–90, respectively. Furthermore, the time slot costs $c_t \in \mathbb{Z}_+$, for each t in the set of time slots $\mathcal{T}$, belong to the sets $\{1, 2, 3, 4\}$, $\{1, 2, 3, 4\}$, and $\{1, 2, \ldots, 8\}$ for instances 1–30, 31–60, and 61–90, respectively. The number $|\mathcal{P}|$ of distinct processing times is in $\{3, 4, 5\}$ for instances 1–30 and $\{3, 4\}$ for instances 31–60, while it is equal to 12 for instances 61–90. Finally, the maximum processing time is equal to $|\mathcal{P}|$ for all the instances, except for instances 1, 3, 4, 6, 9, 12, and 15, where it is equal to 5.

The whole set of instances is available at "https://github.com/ORresearcher/Exact-and-Heuristic-Solution-Approaches-for-Energy-Efficient-Identical-Parallel-Machine-Scheduling".

References

Absalom, E. E., Shukla, A. K., Nath, R., Akinyelu, A. A., Agushaka, J. O., Chiroma, H., & Muhuri, P. K. (2021). Metaheuristics: A comprehensive overview and classification along with bibliometric analysis. *Artificial Intelligence Review, 54*, 4237–4316.

Agrawal, P., & Rao, S. (2014). Energy-aware scheduling of distributed systems. *IEEE Transactions on Automation Science and Engineering, 11*(4), 1163–1175.

Anghinolfi, D., Paolucci, M., & Ronco, R. (2021). A bi-objective heuristic approach for green identical parallel machine scheduling. *European Journal of Operational Research, 289*(2), 416–434.

Audet, C., Digabel, S. L., Cartier, D., Bigeon, J., & Salomon, L. (2021). Performance indicators in multiobjective optimization. *European Journal of Operational Research, 292*(2), 397–422.

Bambagini, M., Marinoni, M., Aydin, H., & Buttazzo, G. (2016). Energy-aware scheduling for real-time systems: A survey. *ACM Transactions on Embedded Computing Systems, 15*(1), 1–34.

Barak, S., Moghdani, R., & Maghsoudlou, H. (2021). Energy-efficient multi-objective flexible manufacturing scheduling. *Journal of Cleaner Production, 283*, 124610.

Branke, J., Deb, K., Miettinen, K., & Slowiński, R. (2008). *Multiobjective optimization: Interactive and evolutionary approaches*: vol. 5252. Springer Science & Business Media.

Castro, P., Ave, G. D., Engell, S., Grossmann, I., & Harjunkoski, I. (2020). Industrial demand side management of a steel plant considering alternative power modes and electrode replacement. *Industrial and Engineering Chemistry Research, 59*(30), 13642–13656.

Castro, P., Sun, L., & Harjunkoski, I. (2013). Resource–task network formulations for industrial demand side management of a steel plant. *Industrial and Engineering Chemistry Research, 52*(36), 13046–13058.

Catanzaro, D., Pesenti, R., & Ronco, R. (2023). Job scheduling under time-of-use energy tariffs for sustainable manufacturing: A survey. *European Journal of Operational Research, 308*(3), 1091–1109.

Caviglione, L., Gaggero, M., Paolucci, M., & Ronco, R. (2021). Deep reinforcement learning for multi-objective placement of virtual machines in cloud datacenters. *Soft Computing, 25*(19), 12569–12588.

Chankong, V., & Haimes, Y. (2008). Multiobjective decision making: Theory and methodology. *Dover Books on Engineering*. Dover Publications.

Chen, B., & Zhang, X. (2019). Scheduling with time-of-use costs. *European Journal of Operational Research, 274*(3), 900–908.

Chen, L., Megow, N., Rischke, R., Stougie, L., & Verschae, J. (2021). Optimal algorithms for scheduling under time-of-use tariffs. *Annals of Operations Research, 304*, 85–107.

Cheng, J., Chu, F., Liu, M., Wu, P., & Xia, W. (2017). Bi-criteria single-machine batch scheduling with machine on/off switching under time-of-use tariffs. *Computers and Industrial Engineering, 112*, 721–734.

Cheng, J., Chu, F., & Zhou, M. (2018). An improved model for parallel machine scheduling under time-of-use electricity price. *IEEE Transactions on Automation Science and Engineering, 15*(2), 896–899.

Cheng, J., Wu, P., & Chu, F. (2019). Mixed-integer programming for unrelated parallel machines scheduling problem considering electricity cost and makespan penalty cost. In *2019 International conference on industrial engineering and systems management (IESM)* (pp. 1–5). IEEE.

Coello Coello, C. A., & Reyes Sierra, M. (2004). A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In *MICAI 2004: Advances in artificial intelligence: Third mexican international conference on artificial intelligence, Mexico City, Mexico, April 26–30, 2004. Proceedings 3* (pp. 688–697). Springer.

Deb, K. (2001). Multi-objective optimization using evolutionary algorithms. In *Wiley-interscience series in systems and optimization*.

Deb, K., & Jain, H. (2013). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Transactions on evolutionary computation, 18*(4), 577–601.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.

Ding, J.-Y., Song, S., Zhang, R., Chiong, R., & Wu, C. (2016). Parallel machine scheduling under time-of-use electricity prices: New models and optimiza-

tion approaches. *IEEE Transactions on Automation Science and Engineering, 13*(2), 1138–1154.
Fang, K., Uhan, N. A., Zhao, F., & Sutherland, J. W. (2016). Scheduling on a single machine under time-of-use electricity tariffs. *Annals of Operations Research, 238*(1), 199–227.
Fang, K.-T., & Lin, B. (2013). Parallel-machine scheduling to minimize tardiness penalty and power cost. *Computers and Industrial Engineering, 64*, 224–234.
Faria, G., Vieira, S., & Branco, P. (2019). Evolutionary process scheduling approach for energy cost minimization in a yeast production factory: Design, simulation, and factory implementation. *Energy Systems, 10*(1), 113–139.
Forghani, A., Lotfi, M., Ranjbar, M., & Sadegheih, A. (2021). Hierarchical framework for maintenance and production scheduling of continuous ball mills in tile industries under TOU electricity pricing. *Journal of Cleaner Production, 327*, 129440.
Gahm, C., Denz, F., Dirr, M., & Tuma, A. (2016). Energy-efficient scheduling in manufacturing companies: A review and research framework. *European Journal of Operational Research, 248*, 744–757.
Gao, K., Huang, Y., Sadollah, A., & Wang, L. (2020). A review of energy-efficient scheduling in intelligent production systems. *Complex and Intelligent Systems, 6*, 237–249.
Garey, M. R., & Johnson, D. S. (1978). "Strong" NP-completeness results: Motivation, examples, and implications. *Journal of the ACM, 25*, 499–508.
Gibbons, J., & Chakraborti, S. (2020). *Nonparametric statistical inference*. CRC Press.
Giret, A., Trentesaux, D., & Prabhu, V. (2015). Sustainability in manufacturing operations scheduling: A state of the art review. *Journal of Manufacturing Systems, 37*, 126–140.
Golmohamadi, H. (2022). Demand-side management in industrial sector: A review of heavy industries. *Renewable and Sustainable Energy Reviews, 156*, 111963.
Guerreiro, A. P., Fonseca, C. M., & Paquete, L. (2021). The hypervolume indicator. *ACM Computing Surveys, 54*, 1–42.
Haapala, K., Zhao, F., Camelio, J., Sutherland, J., Skerlos, S., Dornfeld, D., & Rickli, J. (2013). A review of engineering research in sustainable manufacturing. *Journal of Manufacturing Science and Engineering, 135*(4), 1–16.
Haimes, Y., Lasdon, L., & Wismer, D. (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics, 1*, 296–297.
Ishibuchi, H., Masuda, H., Tanigaki, Y., & Nojima, Y. (2015). Modified distance calculation in generational distance and inverted generational distance. In *Evolutionary multi-criterion optimization: 8th international conference, EMO 2015, Guimarães, Portugal, March 29–April 1, 2015. Proceedings, part II 8* (pp. 110–125). Springer.
Jia, Z.-H., Zhang, Y.-L., Leung, J., & Li, K. (2017). Bi-criteria ant colony optimization algorithm for minimizing makespan and energy consumption on parallel batch machines. *Applied Soft Computing, 55*, 226–237.
Jiang, E.-d., & Wang, L. (2020). Multi-objective optimization based on decomposition for flexible job shop scheduling under time-of-use electricity prices. *Knowledge-Based Systems, 204*, 106177.
Karimi, S., Kwon, S., & Ning, F. (2021). Energy-aware production scheduling for additive manufacturing. *Journal of Cleaner Production, 278*, 123183.
Lei, D., Li, M., & Wang, L. (2018). A two-phase meta-heuristic for multiobjective flexible job shop scheduling problem with total energy consumption threshold. *IEEE Transactions on Cybernetics, 49*(3), 1097–1109.
Li, K., Zhang, X., Leung, J., & Yang, S.-L. (2016). Parallel machine scheduling problems in green manufacturing industry. *Journal of Manufacturing Systems, 38*, 98–106.
Mitra, S., Grossmann, I., Pinto, J., & Arora, N. (2012). Optimal production planning under time-sensitive electricity prices for continuous power-intensive processes. *Computers and Chemical Engineering, 38*, 171–184.
Moon, J.-Y., Shin, K., & Park, J. (2013). Optimization of production scheduling with time-dependent and machine-dependent electricity cost for industrial energy efficiency. *The International Journal of Advanced Manufacturing Technology, 68*(1), 523–535.
Panda, S., Mohanty, S., Rout, P., Sahu, B., Bajaj, M., Zawbaa, H., & Kamel, S. (2022). Residential demand side management model, optimization and future perspective: A review. *Energy Reports, 8*, 3727–3766.
Pei, Z., Wan, M., Jiang, Z.-Z., Wang, Z., & Dai, X. (2021). An approximation algorithm for unrelated parallel machine scheduling under TOU electricity tariffs. *IEEE Transactions on Automation Science and Engineering, 18*(2), 743–756.
Pinedo, M. (2016). *Scheduling: Theory, algorithms, and systems* (5th ed.). Springer.
Qian, S.-y., Jia, Z.-h., & Li, K. (2020). A multi-objective evolutionary algorithm based on adaptive clustering for energy-aware batch scheduling problem. *Future Generation Computer Systems, 113*, 441–453.
Rocholl, J., Mönch, L., & Fowler, J. (2020). Bi-criteria parallel batch machine scheduling to minimize total weighted tardiness and electricity cost. *Journal of Business Economics, 90*(9), 1345–1381.
Sharma, A., Zhao, F., & Sutherland, J. W. (2015). Econological scheduling of a manufacturing enterprise operating under a time-of-use electricity tariff. *Journal of Cleaner Production, 108*, 256–270.
Sin, I. H., & Chung, B. D. (2020). Bi-objective optimization approach for energy aware scheduling considering electricity cost and preventive maintenance using genetic algorithm. *Journal of Cleaner Production, 244*, 118869.
Tang, D., Dai, M., Salido, M. A., & Giret, A. (2016). Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization. *Computers in Industry, 81*, 82–95.
Wang, S., Wang, X., Yu, J., Ma, S., & Liu, M. (2018). Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan. *Journal of Cleaner Production, 193*, 424–440.
Wang, Y., & Li, L. (2015). Time-of-use electricity pricing for industrial customers: A survey of U.S. utilities. *Applied Energy, 149*, 89–103.
Zeng, Q.-q., Li, J.-q., Li, R.-h., Huang, T., Han, Y.-y., & Sang, H. (2023). Improved NS-GA-II for energy-efficient distributed no-wait flow-shop with sequence-dependent setup time. *Complex and Intelligent Systems, 9*, 825–849.
Zeng, Y., Che, A., & Wu, X. (2018). Bi-objective scheduling on uniform parallel machines considering electricity cost. *Engineering Optimization, 50*(1), 19–36.
Zhang, R., & Chiong, R. (2016). Solving the energy-efficient job shop scheduling problem: A multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption. *Journal of Cleaner Production, 112*, 3361–3375.
Zhou, S., Jin, M., & Du, N. (2020). Energy-efficient scheduling of a single batch processing machine with dynamic job arrival times. *Energy, 209*, 118420.
Zitzler, E., Brockhoff, D., & Thiele, L. (2007). The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In *Evolutionary multi-criterion optimization: 4th international conference, EMO 2007, Matsushima, Japan, March 5–8, 2007. Proceedings 4* (pp. 862–876). Springer.
Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation, 3*(4), 257–271.