Exact and Heuristic Algorithms for Energy-Efficient Scheduling

by

Roberto Ronco

Submitted to the Department of Computer Science, Bioengineering, Robotics, and Systems Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Systems Engineering

at the

University of Genoa

January 2022

Certified by .....
Massimo Paolucci
Associate Professor
Thesis Supervisor

Certified by .....
Raffaele Pesenti
Full Professor
Thesis Supervisor

Accepted by .....
Chairman name
Chairman, Department Committee on Doctoral Theses

Contents

List of Figures

List of Tables

Preface

The combined increase of energy demand and environmental pollution at a global scale is entailing a rethinking of the production models in sustainable terms. As a consequence, energy suppliers are starting to adopt strategies that flatten demand peaks in power plants by means of pricing policies that stimulate a change in the consumption practices of customers. A representative example is the *Time-of-Use* (TOU)-based tariffs policy, which encourages electricity usage at off-peak hours by means of low prices, while penalizing peak hours with higher prices. To avoid a sharp increment of the energy supply costs, manufacturing industry must carefully reschedule the production process, by shifting it towards less expensive periods. The TOU-based tariffs policy induces an implicit partitioning of the time horizon of the production into a set of time slots, each associated with a non-negative cost that becomes a part of the optimization objective.

This thesis focuses on a representative bi-objective energy-efficient job scheduling problem on parallel identical machines under TOU-based tariffs by delving into the description of its inherent properties, mathematical formulations, and solution approaches. Specifically, the thesis starts by reviewing the flourishing literature on the subject, and providing a useful framework for theoreticians and practitioners. Subsequently, it describes the considered problem and investigates its theoretical properties. In the same chapter, it presents a first mathematical model for the problem, as well as a possible reformulation that exploits the structure of the solution space so as to achieve a considerable increase in compactness. Afterwards, the thesis introduces a sophisticated heuristic scheme to tackle the inherent hardness of the problem, and an exact algorithm that exploits the mathematical models. Then, it shows the computational efficiency of the presented solution approaches on a wide test benchmark. Finally, it presents a perspective on future research directions for the class of energy-efficient scheduling problems under TOU-based tariffs as a whole.

Chapter 1

Introduction

This thesis deals with an energy-efficient job scheduling problem on parallel identical machines under TOU-based tariffs that requires to find a schedule so as to simultaneously minimize the maximum completion time of the jobs, and a measure of the energy consumption cost of the schedule. The purpose of the thesis is to tackle such problem by means of a fast heuristic scheme and an efficient exact algorithm that exploits structural properties of the solutions space.

This chapter is organized as follows. Section 1.1 stresses the compelling need and hard challenge of properly integrating energy efficiency criteria and objectives in the production processes. Afterwards, Section 1.2 introduces a formal framework that proves useful to provide a solid foundation for the basic definitions, properties, and notation used in scheduling under TOU-based tariffs. Then, Section 1.3 presents a thorough literature review on the subject. Section 1.4 concludes by describing the structure of the remainder of the thesis.

1.1 Motivation

The dramatic rise of worldwide energy consumption in the last decades, together with the related environmental pollution, became a compelling matter for global demand supply. The World Energy Outlook 2019 reported 14314 million tonnes of oil-equivalent demand from primary energy consumption, corresponding to 33.2 gigatonnes of CO_2 emission in the sole 2018 [42]. The sectors related to manufacturing, agriculture, mining, and construction collectively consumed 54% of the worldwide energy demand in 2012 [41]. In 2018, the total energy consumption of the sole manufacturing sector amounted to 19.436

trillion British thermal units [40] and, all together, the above sectors were forecast to grow at a rate of 1.2% per year until 2040 [41].

Rethinking the production processes under a sustainable lens, and simultaneously fostering environment-aware consumption practices in customers, appear to be a necessary condition to invert this trend on the long term. One of the first actions undertaken by energy suppliers to relieve this trend in the short term consisted of flattening the peaks of demand in power plants by means of strategies aimed at reducing the high economic burdens related to the generation of high energy loads in short periods of time and, consequently, the environmental impact related to energy production. These strategies mostly consist of pricing policies that stimulate a change in the consumption practices of customers [59]. One example of such policies is provided by the *Time-of-Use* (TOU)-based tariffs, that spur electricity usage at off-peak hours by means of low prices, while penalizing peak hours with higher prices. The identification of the optimal prices with regard to the inherent stochasticity of the demand can be carried out, e.g., by means of robust optimization approaches, such as those described in [59], or stochastic programming models, such as those discussed in [83]. Overall, TOU-based tariff policies proved effective to flatten the peaks of demand so far, by ensuring, at the same time, a good service stability [16, 59]. How long they will succeed to contain a world-scale increase in energy consumption, however, still remain obscure to predict.

In the context of manufacturing, an immediate reaction to TOU-based tariffs consisted of carefully rescheduling the production processes during periods characterized by low energy supply costs. One of the requirements to accomplish this task is the rescheduling the jobs involved in such processes. Job Scheduling under TOU-based tariffs (or *TOU scheduling* for short) is similar to classical job scheduling problems, in that a processing order of a number of jobs must be identified and carried out on one or possibly multiple machines [88]. The processing of the jobs must comply with time requirements (e.g., a due date or a common deadline) [88]. However, the presence of TOU-based tariffs induces an implicit partitioning of the time horizon of the production into a set of time slots, each associated with a non-negative cost called *TOU costs* [19] or *TOU prices* [37]. This partitioning is usually referred to as *TOU pricing scheme* [18, 37, 98]. The sum of the TOU costs of slots associated with some job processing in the considered time horizon, also referred to as the *Total Energy Cost* (TEC), becomes part of the classical objectives of a scheduling problem or, in the context of multi-objective optimization problems, adds to the classical objectives in scheduling, e.g., the flow time or the total weighted tardiness [88].

In addition, the TEC, being a non-regular performance measure [88], often represents a conflicting objective with respect to the usual classical ones.

The literature on sustainable manufacturing systems offers a number of recent surveys on energy-efficient scheduling. For instance, Gahm et al. [44] study the impact of energy-aware practices in sustainable production and propose a classification of energy-efficient job scheduling based on energetic coverage, energy supply and energy demand. Choudhury et al. [29] focus on the technological aspects of manufacturing, by analyzing the service supply chain and its environmental-related concerns. Despeisse et al. [36] propose a conceptual framework to model single factory units as ecosystems aimed at enabling sustainable performance improvements while limiting environmental pollution and unrestrained exploitation of natural resources. Giret et al. [48] focus on energy-efficiency operations scheduling by exploring the conditions for its sustainability. They distinguish between (i) approaches based on the input data (e.g., as machines, jobs, and scheduling horizon), (ii) approaches based on the environmental output (e.g., pollution, waste), and (iii) approaches based on mixed objectives. Finally, Gao et al. [45] focus on energy-efficient scheduling in intelligent production systems, by proposing a general classification of the most important problems in sustainable manufacturing and of the corresponding solution approaches. None of the above surveys, however, comprehensively discusses the models, methods, and algorithms for job scheduling under TOU-based energy tariffs presented in the literature. This chapter closes this gap, by providing researchers and practitioners with (i) a framework that may summarize the most important theoretical results and practical applications on the topic as well as (ii) a guide that may direct new research efforts towards unexplored directions in TOU scheduling for sustainable manufacturing.

1.2 Basic definitions and notation

This section formalizes the job scheduling problems in presence of TOU-based energy tariffs, by discussing the properties that time slots, jobs, and machines may have, as well as the different problems that occur in the literature.

Let us introduce some definitions and notation. Given three positive integers N, M, and K, let $\mathcal{J} = \{1, 2, \ldots, N\}$, $\mathcal{H} = \{1, 2, \ldots, M\}$, and $\mathcal{T} = \{1, 2, \ldots, K\}$, be a set of jobs, machines, and time slots, respectively. Each job in $\mathcal{J}$ has to be processed on a machine of the set $\mathcal{H}$, until its completion, in a subset of the K consecutive time slots in $\mathcal{T}$ that constitute the time horizon of the production. Furthermore, each machine $h \in \mathcal{H}$ can

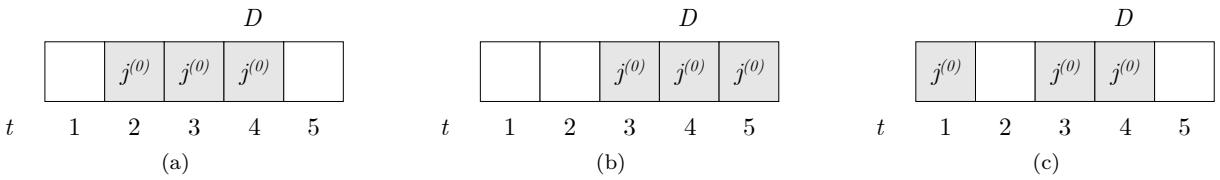

Figure 1: An example of three possible schedules for a dummy instance of the JSTP including a single job j with processing time 3, a time horizon consisting of 5 time slots, a deadline $D = 4$, and TOU costs all equal to one and omitted for the sake of clarity. The schedule shown in Figure 1(a) is non-preemptive, while the one shown in Figure 1(c) is preemptive. The schedule shown in Figure 1(a) is feasible as the processing of the job is completed within the deadline. The other two schedules are infeasible. In particular, the schedule shown in Figure 1(b) violate the deadline and the schedule shown in Figure 1(c) violates non-preemption.

process at most one job at a time.

Let $p_{j,h}$ denote a positive integer encoding the *processing time* of the job $j \in \mathcal{J}$ on machine $h \in \mathcal{H}$, i.e., the number of time slots required to process j on h. If the processing time of the jobs does not depend on h, such as in the case of parallel identical machines, we omit the second subscript from $p_{j,h}$. Let also c_t denote a positive real representing the TOU cost associated with the time slot $t \in \mathcal{T}$. The sequence $\{c_t, t = 1, 2, \ldots, K\}$ is referred to as $\{c_t\}$. Moreover, $\{c_t\}$ is *pyramidal* [43] if there is some slot t' such that $c_1 < c_2 < \ldots < c_{t'-1} < c_{t'} > c_{t'+1} > \ldots > c_{K-1} > c_K$. When the costs c_t are non-increasing (non-decreasing) with respect to $t = 1, 2, \ldots, K$, $\{c_t\}$ is simply said to be non-increasing (non-decreasing).

Let us define a *job-machine-time slots assignment* as a triplet $(j, h_j, \mathcal{T}_j)$ such that j is a job in $\mathcal{J}$, h_j is a machine in $\mathcal{H}$, and $\mathcal{T}_j$ is a non-empty subset of $\mathcal{T}$ encoding the time slots during which h_j processes j. Then, let us define a *schedule* $\mathcal{S}$ as a set of job-machine-time slots assignments satisfying the following two conditions: (i) exactly one pair $(j, \mathcal{T}_j)$ exists for each $j \in \mathcal{J}$ and (ii) for any pair of distinct jobs $j', j'' \in \mathcal{J}$, the intersection $\mathcal{T}_{j'} \cap \mathcal{T}_{j''}$ is empty if $h_j = h_{j'}$, i.e.,

$$\mathcal{S} = \{(j, h_j, \mathcal{T}_j) : \emptyset \neq \mathcal{T}_j \subseteq \mathcal{T}, \forall\, j \in \mathcal{J}, \mathcal{T}_{j'} \cap \mathcal{T}_{j''} = \emptyset \;\forall\; j', j'' \in \mathcal{J}, j' \neq j'', h_j = h_{j'}\}. \quad (1.1)$$

Then, the *makespan* $C_{\max}$ of $\mathcal{S}$ is

$$C_{\max}(\mathcal{S}) = \max_{t \in \bigcup_{j \in \mathcal{J}} \mathcal{T}_j} t, \quad (1.2)$$

i.e., it is the overall amount of time necessary to process the jobs in $\mathcal{J}$ on the given machine.

A schedule $\mathcal{S}$ is *preemptive* if, for some $j \in \mathcal{J}$, $\mathcal{T}_j$ contains p_j non consecutive slots, and *non-preemptive* otherwise. For example, Figure 1 shows three possible schedules for a job j characterized by a processing time $p_j = 3$ over a time horizon of 5 slots. The schedule shown in Figure 1(a) is non-preemptive as job j is processed consecutively during 3 time slots. The schedule shown in Figure 1(c) is instead preemptive, as job j is not processed consecutively during a number of time slots equal to p_j.

Each machine $h \in \mathcal{H}$ is associated with a positive energy consumption rate u_h. The energy consumption rate reflects into the energy cost of the scheduled jobs. Formally, if job $j \in \mathcal{J}$ is scheduled in $\mathcal{T}_j$ on machine $h_j \in \mathcal{H}$, the cost associated with the processing of the job j is $u_{h_j} \sum_{k \in \mathcal{T}_j} c_k$. Then, let us define the *Total Energy Cost* (TEC) of a schedule $\mathcal{S}$ as

$$E(\mathcal{S}) = \sum_{j \in \mathcal{J}} u_{h_j} \sum_{k \in \mathcal{T}_j} c_k, \tag{1.3}$$

i.e., as the sum of the TOU costs associated to the job-machine-time slots assignments in $\mathcal{S}$.

This thesis deals with the problem of finding a schedule of a set $\mathcal{J}$ of independent jobs, with no release times and no due dates, on a set $\mathcal{H}$ of parallel, identical machines, over the time horizon given as the set of time slots $\mathcal{T}$, so as to simultaneously minimize (1.2) and (1.3). However, with intent of laying a basic foundation to the discussion, let us begin from one of the simplest and most representative TOU scheduling problems, and then progressively expand over it to obtain a comprehensive view of the subject.

Given a set of jobs $\mathcal{J}$, a set of integer processing times associated with the jobs in $\mathcal{J}$, a time horizon $\mathcal{T}$, a set $\{c_k\}$ of TOU costs associated with the time slots in $\mathcal{T}$, and a positive integer D, called *deadline*, the Job Scheduling with Time-of-Use Cost Problem (JSTP) is to find a non-preemptive schedule $\mathcal{S}$ on a single machine that minimizes the total energy cost (1.3), and whose makespan (1.2) is smaller than or equal to the given deadline, i.e.,

$$\min_{\mathcal{S}} \quad E(\mathcal{S}) \tag{1.4}$$

$$s.t. \quad C_{max}(\mathcal{S}) \leq D. \tag{1.5}$$

As an example, by referring again to the three alternative schedules shown in Figure

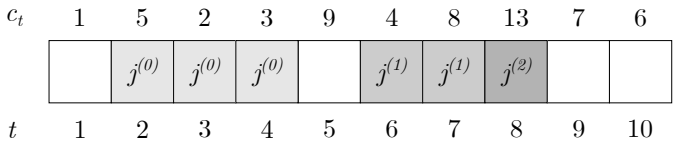

Figure 2: An example of a possible schedule for an instance of the JSTP having three jobs $\{j^{(0)}, j^{(1)}, j^{(2)}\}$, processing times $\{3, 2, 1\}$, a time horizon of 10 time slots, and TOU costs $c_t = \{1, 5, 2, 3, 9, 4, 8, 13, 7, 6\}$. The energy consumption rate of the machine is 1, and the deadline D is assumed to be equal to 10.

1, the schedule shown in Figure 1(a) is a feasible solution for the toy instance of the JSTP including the single job j with processing time 3, a time horizon of 5 time slots, a deadline $D = 4$, and TOU costs all equal to 1 (and omitted for the sake of clarity), as its makespan satisfies (1.5). Instead, the schedule in Figure 1(b) is infeasible, as part of job j is processed after the deadline. Similarly, the schedule in Figure 1(c) satisfies the deadline D, but it is not feasible as well because it is preemptive. Figure 2 shows instead an example of a possible schedule for an instance of the JSTP. This schedule has makespan and TEC equal to 8 and 35, respectively. Observe that if job j was instead assigned to slot 1, the resulting schedule would achieve a makespan equal to 7 and a TEC equal to 23, which is provably minimum.

The literature on TOU scheduling describes several *versions* of the JSTP. As a significant example, Chen and Zhang [19] considered a JSTP version called *Scheduling with Time-of-Use Costs Problem* (STOUCP). The difference between the JSTP and the STOUCP lies in the definition of the time horizon, and in the consequent relaxation of the integrality of the start times of the jobs. Specifically, in the STOUCP, for a given positive integer K, the time horizon is given as a continuous interval $[0, K] \subseteq \mathbb{R}$. As a consequence, a job $j \in \mathcal{J}$ can start at any $s_j \in [0, K]$. Moreover, the set of time slots $\mathcal{T}$ is partitioned into the set π of *time periods*, or *time intervals*. Each time interval in π is a set of time slots that have the same TOU cost. The time horizon is often presented as a set of time periods in TOU scheduling problems [20, 43, 108].

In some JSTP versions, the machines in $\mathcal{H}$ can be regulated by an "on/off" switching mechanism. In such case, each machine can be in one of these three states: (a) "processing", (b) "idle", or (c) "shutdown". A machine can process a job, i.e., be in the state (a), only while "on". If a machine is still "on" but it is not processing any job, then it is in state (b). In such state, a machine requires energy despite not processing any job, but it is ready to start processing a job at any time. In state (c), the machine is shut down, i.e., it is "off". Its energy consumption is zero, but it cannot process any job without first transitioning into state (a). However, in order to do so, a machine requires time and

Property	**Notation (β field)**
Jobs power demands	q_j
Fixed jobs processing sequence (on a single machine)	seq
Continuous start time of the jobs	$s_j \in \mathbb{R}_{0+}$
Machines regulation via "on/off" switching mechanism	on/off

Table 1: Three-field notation useful for TOU scheduling.

energy. As an example, Shrouf et al. [95] tackle a JSTP version that is identical to the JSTP, except for the "on/off" switching mechanism that adds to the machine functionality. The objective function takes into account both the TEC of the schedule and the energy consumed by the machines due to the state transitions.

For each $j \in \mathcal{J}$, a power demand $q_j > 0$ may be associated with job j, such as in the one of the problems studied by Fang et al. [43]. From a manufacturing standpoint, the introduction of power demands for jobs reflects the need to model energy-intensive and non-energy-intensive jobs. Formally, if $j \in \mathcal{J}$ is processed in $\mathcal{T}_j \in \mathcal{T}$ on machine h, the cost of processing j is $q_j \sum_{t \in \mathcal{T}_j} c_t$. More generally, if the power demand of $j \in \mathcal{J}$ also depends on the machine $h \in \mathcal{H}$ that processes j, then the power demand of j is expressed as $q_{j,h} > 0$.

In some JSTP versions, the jobs in $\mathcal{J}$ are constrained to follow a specific processing sequence on machines. Such processing sequence induces a total ordering on the start times s_j, $j \in \mathcal{J}$. It is interesting to observe that some JSTP versions with such a total ordering can be solved to optimality in polynomial [5, 8] or pseudo-polynomial [4] time.

In order to describe the TOU scheduling problems, the remainder of the paper makes extensive use of the $\alpha|\beta|\gamma$ notation introduced by Graham et al. [51], also referred to as the *three-field* notation. The $\alpha|\beta|\gamma$ notation is a classical formal tool of the literature of job scheduling that allows one to concisely state different scheduling problems. Specifically, the α field characterizes the machines environment. For instance, the case of parallel identical machines is represented with Pm in the α field. Instead, the β field specifies the constraints and the processing restrictions, such as jobs preemption, represented with $prmp$. Finally, the γ field contains the optimization objectives, such as the minimization of the makespan. For more detailed and complete information on the naming conventions used in the three-field notation for classical scheduling, see Appendix A or the seminal book of Pinedo [88].

Table 1 summarizes the three-field notation useful to TOU scheduling. As a conclusive

remark for this section, the presence of TOU costs is usually denoted by *slotcost* in the β field [103] in the literature. However, since this thesis only consider TOU scheduling problems in this article, it will be disregarded from now on.

1.3 A literature review

In this section, we propose a systematic classification of the literature of TOU scheduling by dividing the considered problems into two different classes.

The first class consists of the problems that are easily representable with the three-field notation. Among the problems in the first class, we further distinguish between (a) problems that are solvable to optimality with a polynomial or a pseudo-polynomial time algorithm (Subsection 1.3.1), and (b) problems that have been proven to be NP-hard, or that do not have a polynomial or pseudo-polynomial exact solution algorithm yet (Subsection 1.3.2).

The second class is instead constituted by (a) problems with particular characteristics that would require non-standard notation, such as technological features or unusual objective functions (Subsection 1.3.3), and (b) applicative problems inspired by a practical case study (Subsection 1.3.4).

The problem studied in this thesis is $\mathcal{NP}$-hard and belongs to the first class, as it can be stated as $Pm||C^{\max}$, TEC. It was first studied in the seminal work of Wang et al. [108], and then thoroughly investigated in our recent article [9].

The purpose of this section is to provide a thorough reference that should serve as a general overview for scheduling researchers and industry decision makers.

1.3.1 Problems with an exact algorithm running in polynomial or pseudo-polynomial time

We summarize the problems that have been proven to admit a polynomial or a pseudo-polynomial algorithm in Table 2. The seminal work of Wan and Qi [103] laid the foundations for single-machine scheduling under TOU costs. Specifically, Wan and Qi [103] considered five different JSTP versions, corresponding to five different objective functions obtained by summing the TEC with five regular functions often used in non-TOU scheduling: the total flow time, the maximum lateness, the maximum tardiness, the weighted number of tardy jobs, and the total tardiness. The authors proved the strong NP-hardness

of all the five problems with a reduction from the 3-partition problem. In addition, they proposed polynomial algorithms for the problems under specific assumptions. Chen and Zhang [19] focused on $1|s_j \in \mathbb{R}_{0^+}|TEC$, proved its strong NP-hardness again with a reduction from the 3-partition problem. Similarly, the authors provided different polynomial and pseudo-polynomial algorithms based on several assumption on the time slot costs. Such work is essential in highlighting the relations between the structure of $\{c_k\}$ and the existence of efficient solution algorithms. Fang et al. [43] and Chen et al. [20] provided several results for preemptive TOU scheduling. The former authors proved the existence of polynomial algorithms for TOU scheduling with jobs weights and power demands under specific assumptions. Instead, the latter authors specifically provided optimal and approximation polynomial algorithms for different JSTP versions, considering fixed jobs sequence, jobs weights and release dates. To the best of our knowledge, Penn and Raviv [87] are the only authors to provide a polynomial-time algorithm for a preemptive TOU scheduling problem that considers due dates. Instead, Aghelinejad et al. extensively studied the class of single-machine TOU scheduling problems with on/off switching with the objective of minimizing the TEC. Aghelinejad et al. first relaxed the problem by allowing preemptive schedules [2]. In a subsequent work, Aghelinejad et al. introduced the assumption of the fixed jobs sequence [5, 8]. In this setting, they later considered the jobs' power consumption and different machine speeds [4]. Finally, Wang et al. [109] considered a two-machines permutation flow shop scheduling problem, and they provided a polynomial-time algorithm when the jobs sequence is fixed.

1.3.2 Problems with approximation or heuristic algorithms

In this section, we deal with TOU scheduling problems that have not been proven to be solvable in polynomial or pseudo-polynomial time. Table 3 reports a classification of such problems according to the type of the processing machines and the number of the objective functions. Let us first deal with single-objective single-machine TOU scheduling problems. Chen and Zhang [19] provided a FPTAS for the STOUCP by exploiting the notion of *valley* for a set of time periods π, which is a time interval with a TOU cost lower than its neighbors. Specifically, the authors found that the STOUCP admits a FPTAS when the set of time periods π has at most 2 valleys. Chen et al. [20] and Kulkarni and Munagala [65] are the only authors, at the best of our knowledge, to provide a PTAS for a preemptive scheduling problem. Fang et al. [43] provided a $\sum_{k=1}^{K}(c_k \,/\, \min_{t\in\mathcal{T}} c_t)^{\alpha\,//(\alpha-1)}$-

Article	Problem	Computational complexity of the solution algorithm	Assumptions
Wan and Qi [103]	$1\|\|\sum_{j\in J} C_j + TEC$	$O(K^2)$	$\{c_k\}$ non-increasing
	$1\|\|L_{max} + TEC$	$O(K^2)$	$\{c_k\}$ non-increasing
	$1\|\|T_{max} + TEC$	$O(K^2)$	$\{c_k\}$ non-increasing
	$1\|\|\sum_{j\in J} w_j U_j + TEC$	(a) $O(K^2)$ or (b) $O(NK^3)$	tardy jobs as (a) lost sales or (b) backlogging
	$1\|\|\sum_{j\in J} T_j + TEC$	$O(N^4K^3)$	$\{c_k\}$ monotone non-increasing
Fang et al. [43]	$1\|q_j, prmp\|TEC$	No explicit expression.	
	$1\|q_j\|TEC$	No explicit expression.	$\{c_k\}$ pyramidal
	$1\|w_j, q_j, prmp\|TEC$	No explicit expression.	$c_k > 0, k \in \mathcal{T}$
Chen and Zhang [19]	$1\|\|TEC$	$O(\|\pi\|)$	π has 1 valley
		$O((\sum_{j\in\mathcal{J}} p_j)^v(N + \|\pi\|^v))$	π has $v \geq 2$ valley
		$O(\|\pi\|^2 N^{2r-1})$	π has 2 valleys, $\|\mathcal{P}_\mathcal{J}\| = r$
		$O(\|\pi\|(\|\pi\| + N)N^3)$	Bounded lateness, π has 1 valley
		$O(\|\pi\|^{\pi+2} N^{3\|\pi\|-1})$	$\sum_{j\in J} p_j$ bounded; π has 1 valley
Chen et al. [20]	$1\|prmp\|\sum_{j\in J} C_j + TEC$	No explicit expression.	
	$1\|prmp, seq, w_j\|\sum_{j\in J} w_j C_j + TEC$	$O(NK)$	
Penn and Raviv [87]	$1\|r_j, d_j, prmp\|TEC$	$O(log_2 K(K + N^2 log_2 K))$	
Aghelinejad et al. [2]	$1\|\text{on/off}, prmp\|TEC$	$O(K^3)$	
Aghelinejad et al. [5, 8]	$1\|\text{on/off}, seq\|TEC$	$O(K^3)$	
Aghelinejad et al. [4]	$1\|\text{on/off}, seq, q_j\|TEC$	$O(K^3)$	
	$1\|\text{on/off}, seq, q_j, \text{speed}\|TEC$	$O(T^2V(T+V))$	
Wang et al. [109]	$F2\|prmu, seq\|TEC$	$O(NK^4)$	

Table 2: Optimal polynomial and pseudo-polynomial time algorithms for different JSTP versions.

approximation algorithm for $1|w_j, q_j|TEC$ (the authors also studied the preemptive version of such problem, see Section 1.3.1). Instead, Che et al. [17] considered the problem $1|s_j \in \mathbb{R}|TEC$, proven to be strongly NP-hard by Fang et al. [43]. Che et al. proposed a simple MILP formulation with two different types of assignment variables to handle the condition $s_j \in \mathbb{R}$, and they presented a greedy insertion heuristic by building upon the work of Fang et al. [43]. Shrouf et al. were the first authors to study $1|\text{on/off}|TEC$ [95], as well as to provide a MILP formulation and to develop a GA for the problem. Aghelinejad et al. also studied $1|\text{on/off}|TEC$: by building upon Shrouf et al.'s effort, they provided an improved MILP formulation [1] that they later strenghtened with further lower bounds [8]. In [3], Aghelinejad et al. also developed a heuristic and a GA as well. Finally, Cheng et al. [24] presented and compared two different MILP formulations for $1|batch(b)|TEC$.

In the literature, the only problems that have been proven to admit a PTAS only involve a single machine and a single objective. One of the first efforts in single-machine TOU scheduling with multiple objectives is due to Cheng et al. [22], who developed a bi-objective MILP formulation for the NP-hard problem $1|batch(b)|C_{max}, TEC$. The authors developed an algorithm based on the ϵ-constraint method for multi-objective optimization that is based on solving a sequence of single-objective formulations of the original problems. In [23], Cheng et al. simplified their original MILP formulation [22], and they built a faster optimal ϵ-constraint algorithm upon their results. The general

idea at the core of such algorithm was further used, developed and refined in subsequent works involving bi-objective scheduling [9, 108, 111]. Specifically, for the same batch scheduling problem tackled by Cheng et al., Wu et al. [111] proposed two new fast ϵ-constraint-based constructive heuristics that improve Cheng et al.'s algorithm from a computational standpoint. Zhou et al. [128] also considered a batch scheduling problem with release dates, that requires the simultaneous minimization of the makespan and TEC. Rubaiee and Yildirim [91] considered the same objectives in a single-machine problem with preemption.

Wang et al. [108] considered one of the foundational problems in multi-objective parallel identical machines TOU scheduling, $Pm||C_{max}, TEC$, providing a constructive heuristic and a MILP formulation. Anghinolfi et al. [9] built upon this work and provided a heuristic constituted by an improved constructive heuristic and a local search algorithm; furthermore, they presented an improved MILP formulation. The authors that considered TOU scheduling problems with parallel batch processing machines provided a MILP formulation and one or a few metaheuristics [89, 90, 101, 127]. At the best of our knowledge, Tan et al. [101] are the only authors to deal with non-identical parallel machines.

In the literature, there are also several research efforts in unrelated parallel machines TOU scheduling. The foundational problem $Rm||TEC$ has first been studied by Ding et al. [37], who showed the NP-hardness of the problem and proposed a MILP formulation, as well as a Dantzig–Wolfe decomposition based reformulation of the problem along with a column generation heuristic. Cheng et al. [26] improved the MILP formulation proposed by Ding et al. [37] by using less decision variables, and Saberi-Aliabad et al. [92] further improved the formulation by introducing several strengthening inequalities. Che et al. [18] instead considered a version of the problem characterized by continuous start times. The problem $Rm||KC_{max} + TEC$, for a positive real K, received particular attention during the last decade, with the presentation of multiple MILP formulations and metaheuristics [27, 67, 69, 81]. Pei et al. [85] instead tackle $Rm||C_{max}, TEC$, the only unrelated parallel machines problem with multiple objectives. The authors first linearly combined the two objectives and provided a MINLP formulation for the resulting single-objective problem; subsequently, they exploited a relaxation of the problem to develop an approximate algorithm.

As regards flow shop scheduling, Wang et al. [109] studied a two-machines permutation flow shop scheduling problem with the usual objective of minimizing the TEC. The authors proposed a MILP formulation, as well as a Johnson's rule and a dynamic pro-

gramming based heuristic. Ho et al. [58] further extended Wang et al.'s work by proposing a novel formulation and heuristic for the problem. Aghelinejad et al. instead performed a mathematic modeling effort for the two-machines flow shop scheduling problem with on/off switching mechanisms [6, 7].

Luo et al. [78] are among the first authors to consider a single-objective flow shop TOU scheduling problems. In more detail, Luo et al. studied a flexible (or hybrid) flow shop scheduling problem requiring the minimization of the TEC, and proposed a ACO-based multi-objective metaheuristics. Zhang et al. [121] extended Luo et al.'s work by considering different machine speeds, and they proposed a MILP formulation for the problem as well as a modified "biogeography-based" algorithm combined with VNS. Zhang et al. [122] also extended Luo et al.'s work, but opposite to Zhang et al. [121], they considered an on/off switching mechanism for the considered machines. Peng et al. [86] and Cheng et al. [28] proposed several metaheuristics for two distinct flow shop scheduling problems that require the minimization of the TEC. The first is a general multiple machines flow shop, while the second is a no-wait permutation flow shop problem.

Finally, we deal with the few multi-objective job shop scheduling efforts in the literature. Among these, Kurniawan et al. [70] consider a bi-objective job shop scheduling problem requiring to minimize the TEC and the total weighted tardiness. The authors decompose the problem into the subproblems of a) sequencing the different operations on the machines and b) determining their start time. They propose a "distributed-elite" local search based upon a genetic algorithm that encodes the operations sequence and their start time into two different gene representations. Li et al. [73] instead study a batch scheduling setting with the minimization of the makespan and TEC. In their concise work, they present a mathematical formulation along with some experimental results. Lastly, Jiang and Wang [62] consider a flexible job shop with the minimization of the makespan and TEC, and present both a MILP model and a hybrid MOEA/D. Such metaheuristic employs specific operators to generate new solutions by exploiting information in their neighborhoods, and it also embeds two different intensification operators.

Class	Problem	Article	Solution Approach
Single-objective single-machine scheduling	$1\|prmp, w_j\|\sum_{j\in J} w_j C_j + TEC$	Chen et al. [20]	PTAS
	$1\|prmp, w_j, r_j\|\sum_{j\in J} w_j(C_j - r_j) + TEC$	Kulkarni and Munagala [65]	PTAS (at most two distinct values in $\{c_k\}$)
	$1\|\|TEC$	Chen and Zhang [19]	FPTAS (π has at most 2 valleys)
	$1\|w_j\|TEC$	Zhang et al. [119]	Greedy insertion heuristic algorithm
	$1\|s_j \in \mathbb{R}_{0^+}\|TEC$	Che et al. [17]	MILP formulation, Greedy insertion heuristic
	$1\|w_j, q_j\|TEC$	Fang et al. [43]	$\sum_{k=1}^{K}(c_k \,/\, \min_{t\in\mathcal{T}} c_t)^{\alpha\,/\,(\alpha-1)}$-approximation algorithm
	$1\|$on/off$\|TEC$	Shrouf et al. [95]	MILP, GA
	$1\|$on/off$\|TEC$	Aghelinejad et al. [1]	MILP
	$1\|$on/off$\|TEC$	Aghelinejad et al. [3]	MILP, Heuristic, GA
	$1\|$on/off$\|TEC$	Aghelinejad et al. [8]	New lower bounds for MILP formulation
	$1\|batch(b)\|TEC$	Cheng et al. [24]	MILP
Multi-objective single-machine scheduling	$1\|prmp\|C_{max}, TEC$	Rubaiee and Yildirim [91]	MILP, ACO-based algorithms
	$1\|\|\sum_{j\in\mathcal{J}} w_j T_j$, TEC	Kurniawan et al. [68]	MILP, Hybrid GA with random insertion
	$1\|batch(b)\|C_{max}, TEC$	Cheng et al. [22]	MILP
		Cheng et al. [23]	MILP, exact ϵ-constraint algorithm
		Cheng et al. [25]	Heuristic-based ϵ-constraint heuristics
		Zhang et al. [124]	MILP, Heuristics
		Wu et al. [111]	MILP, Heuristics
	$1\|batch(b), r_j\|C_{max}, TEC$	Zhou et al. [128]	MILP, Hybrid multi-objective meta-heuristic algorithm
Multi-objective parallel identical machines	$Pm\|\|C_{max}, TEC$	Wang et al. [108]	MILP, Constructive heuristic, GA
		Anghinolfi et al. [9]	MILP, "Split-greedy" heuristic, "Exchange-search" local search
	$Pm\|batch(b)\|C_{max}, TEC$	Qian et al. [89]	MILP, Multi-objective EA based on adaptive clustering
	$Pm\|batch(b)\|\sum_{j\in\mathcal{J}} w_j T_j, TEC$	Rocholl et al. [90]	MILP, NSGA-II with embedded heuristics
	$Pm\|batch(b), r_j\|C_{max}, TEC$	Zhou et al. [127]	MILP, Multi-objective differential EA
Non-identical parallel machines	$Qm\|batch(b)\|TEC$	Tan et al. [101]	MILP, SPGA, MPGA
Single-objective unrelated parallel machines	$Rm\|\|TEC$	Ding et al. [37]	MILP
		Cheng et al. [26]	MILP
		Saberi-Aliabad et al. [92]	MILP
	$Rm\|s_j \in \mathbb{R}\|TEC$	Che et al. [18]	MILP, Two-stage heuristic
	$Rm\|prmp\|C_{max} + TEC$	Chen et al. [20]	$O(K^2)$ + time for solving $Rm\|prmp\|C_{max}$ by IP [71]
	$Rm\|\|KC_{max} + TEC$, $K \in \mathbb{R}_{>0}$	Moon et al. [81]	MILP, Hybrid GA
		Kurniawan et al. [67]	MILP, GA
		Cheng et al. [27]	MILP
		Kurniawan et al. [69]	"Triple-chromosome" GA
Multi-objective unrelated parallel machines	$Rm\|\|C_{max}, TEC$	Pei et al. [85]	MILP, Approximation algorithm
Single-objective flow shops	$F2\|prmu\|TEC$	Wang et al. [109]	MILP, ILS
		Ho et al. [58]	Heuristics
	$F2\|$on/off$\|TEC$	Aghelinejad et al. [6]	Two MIP formulations
		Aghelinejad et al. [7]	LP formulation
	$Fm\|\|TEC$	Peng et al. [86]	ILP, PSO
	$Fm\|nwt, prmu\|TEC$	Cheng et al. [28]	MILP, GA
	$FF2(Qm, Rm')\|\|TEC, m > m'$	Zhang et al. [120]	MILP, TS - Greedy insertion algorithm
	$FFc\|\|TEC$	Luo et al. [78]	ACO-based metaheuristic
	$FFc\|$speed$\|TEC$	Zhang et al. [121]	MILP formulation, Modified "biogeography-based" algorithm combined with VNS
	$FFC\|$on/off$\|TEC$	Zhang et al. [122]	SPEA2
Multi-objective flow shops	$Fm\|\|TEC, \sum_{j\in\mathcal{J}} E_j + T_j$	Badri et al. [11]	Single-objective LP after multi-objective fuzzy programming
	$Fm\|prmu\|\sum_{j\in\mathcal{J}} w_j T_j, TEC$	Kurniawan and Fujimura [66]	SPEA2-based metaheuristic
	$Fm\|prmu\|C_{max}, TEC$	Wang et al. [104]	MILP
	$FFc(Qm)\|d_j\|\sum_{j\in\mathcal{J}} T_j, TEC$	Ding et al. [38]	MILP, Hybrid PSO
	$FFc(Pm, batch(b))\|$on/off$\|C_{max}, TEC$	Wang et al. [110]	MILP, Constructive heuristic with local search, TS, ACO
Multi-objective job shops	$Jm\|\|\sum_{j\in\mathcal{J}} w_j T_j, TEC$	Kurniawan et al. [70]	Distributed-elite LS based on GA
	$Jm\|batch(b)\|C_{max}, TEC$	Li et al. [73]	MILP
	$FJc\|\|C_{max}, TEC$	Jiang and Wang [62]	MILP, Hybrid MOEA/D

Table 3: Integer programming algorithms, approximation algorithms, heuristics and metaheuristics for different JSTP problem versions.

1.3.3 Problem with particular features or objectives

In this subsection, we consider JSTP versions that do not fit into our classification due to peculiar features of the considered manufacturing system or particular optimization objectives.

Let us first specifically consider those problems that require the minimization of an unusual or particular objective function. As an example, Penn and Raviv [87] consider a single-machine problem that consists in non-preemptively assigning jobs to time slots in order to maximize a measure of profit associated to the resulting schedule. Specifically, profit is obtained as the difference between the total revenue of the jobs, which is given as input data to the problem, and the energy consumption cost associated with the jobs themselves, depending on their assignment in the schedule. While the general case for such problem is NP-hard, the authors present a pseudo-polynomial time algorithm for the problem when preemption is allowed. Zeng et al. [114] study a uniform parallel machines environment where the total number of machines, a very unusual objective in the literature, has to be minimized along with the TEC. Gong et al. [50] instead propose a flexible job scheduling problem with job recirculation and operation sequence-dependent setups. Alongside the makespan and the TEC, the authors take into account three other different objectives that may arise in a practical manufacturing setting: the total labor cost, the maximal workload, and the total workload. Batista Abikarram et al. [13] expand the model of Shrouf et al. [95] by taking into account demand charges as additional energy consumption in a parallel machines environment. Tan et al. [100] instead aggregate energy consumption with additional costs due to production load shifting (such as employee overtime costs and gas emissions penalties) in a complex batch scheduling setting. Finally, Chen et al. [21] and Zhang et al. [117] also address environmental concerns by integrating carbon and CO_2 emissions within the optimization objectives, respectively. Lee et al. [72] instead consider the minimization of the sum of the TEC and the just-in-time (JIT) cost of a schedule for a single-machine problem. The JIT cost is given as the sum of the mean squared earliness and the tardiness of the jobs, which may be a relevant performance measures for manufacturing productions that require a certain degree of timeliness. The authors experimentally test their dynamic control method on a real case involving a milling process performed by a HAAS machine. Gong et al. [49] consider manufacturing unit processes by studying the problem of minimizing the TEC of the schedule of a set of jobs with due dates on a single machine. More specifically, the authors use finite state machines to describe the energetic transitions of the machines, and they also implemented

a genetic algorithm for the problem. Finally, they validated their approach on a real case study based on the measurements of a griding machines and the energy prices.

Researchers also devoted efforts to model particular situations within the manufacturing chain. For instance, Geng et al. [47] propose a problem based on a type of flexible-flow shop called "re-entrant" [52], and tackle it with an improved multi-objective "ant lion" optimization algorithm. Differently from Geng et al. [47], Wang [106] consider a single-machine setting with a particular technological feature, which is the calibration phase, required to process jobs. Specifically, if the calibration starts at time t, then the machine can process any job in the time slots $t, t+1, \ldots, t+C$ for some fixed $C \in \mathbb{N}$. The objective is to minimize the TEC, while calibrating the machine at most a fixed amount of times. Kong et al. [63] instead consider the dynamic disruptions that can affect manufacturing systems. The authors model such incovenients as "arrival" jobs, and they formulate a rescheduling problem that requires to schedule such jobs alongside a set of the "original" jobs intended for processing in order to minimize the TEC. Tan and Cui [102] again consider the minimization of the TEC, but they introduce a constraint on the maximum power at the peak usage.

Several research efforts successfully integrated the TOU framework within project scheduling [14] problems. As an example, Najafzad et al. [82] consider a bi-objective multi-skill project scheduling problem that consists in scheduling activities in order to minimize the total completion time and project cost. The decision-maker has to perform a trade-off between low-energy processing in off-peak hours and the resulting increase in wages due to shift differentials. Similarly to Najafzad et al. [82], Javanmard et al. [61] consider a project scheduling problem with a multi-skilled workforce, with the aim of the reducing the TEC while respecting the projects priorities as much as possible. While Maghsoudlou et al. [79] present again a multi-skilled project scheduling problem under TOU pricing with the objective of generating a preemptive schedule to minimize the TEC, Wu et al. [112] instead aim at the optimizing the consumption of the multiple resources involved. Du et al. [39] propose a bi-objective model for a resource-constrained (project scheduling) problem with activity splitting and recombining, to optimize the project delay and TEC.

Some authors also explored the possibility of optimizing production while taking into account preventive [10, 97] and planned [31, 32] maintenance. Zhang et al. [118] instead study multiple factories in an electrical grid: facilities can exchange information within the grid, while aiming at the reduction of energy consumption. Finally, Sharma et al. [94]

provide an "econological" model of a manufacturing enterprise constituted by multiple speed-scaling machines by integrating both economic and ecological objectives.

1.3.4 Practical case studies

The literature also offers several efforts towards the efficient exploitation of TOU pricing schemes as a means of cutting expenses in industry production. Hadera et al. [55] and Zeng and Sun [115] are among the first authors to integrate TOU prices in iron and steel production scheduling. While the former authors focus on the melt shop section of a stainless steel plant, the latter specifically focus on steam power systems such as boilers and steam turbines, taking into account surplus byproduct gas flows. Zhao et al. [126] specifically address the ecological concern of reducing the byproduct gases in steel production. Cao et al. [15] describe the problem of simultaneously minimizing the makespan and TEC for the production planning of an iron-steel plant. However, in this setting, the TEC also depends on both the self-generation electricity costs and by the on-grid electrovalence. The authors present a mathematical model, alongside a version of the SPEA2 metaheuristic tailored for the problem. Zhao et al. [125] face a multi-stage production problem that specifically models the rolling process of steel within an electric grid. The authors first formulate the problem as a MINLP model with generalized disjunctive programming constraints, and then they reformulate it as a MILP model. Yang et al. [113] integrate uncertainty by considering stochastic metal elements concentration in scrap steel charge within the production scheduling of a scrap steel melt shop. The proposed robust optimization approach is experimentally validated by comparison with two deterministic models and two multi-stage optimization approaches. Guirong and Qiqiang [54] also consider uncertain data in scheduling, but as a part of the steelmaking-continuous casting production, and tackle the problem with the Monte Carlo based "cascade" cross-entropy algorithm. Pan et al. [84] consider a full steelmaking-refining-continuous casting framework, where TOU-based tariffs model the different time prices for electrical load tracking scheduling, and propose both a MINLP model and an improved SPEA2 metaheuristic for the problem. As a conclusion to the discussion of the applications in metallurgy, Tan et al. [99] manage to combine the Hot Rolling Batch Scheduling Problem with the job-shop scheduling problem. Wang et al. [107] face a real-world glass manufacturing problem, that falls within single-machine batch scheduling, with two heuristics based on decomposition concepts to deal with large-scale instances. Zeng et al. [116] instead specifically

optimize the production scheduling of tissue article mills, which they tackle by means of a multi-objective evolutionary algorithm based on decomposition and "teaching-learning" optimization combined with VNS.

Finally, researchers have also extensively considered energy supply scheduling. Liu et al. [74] study the Virtual Power Plant (VPP), an efficient tool for smarter energy supply in an electric grid under TOU costs with trasversal demand-side resources management. Sichilalu et al. [96] also focus on the electric grid and present a model for the problem of minimizing the energy cost of photovoltaic systems connected by a grid that supply power to heat pump water heaters. Wang et al. [105] instead investigate the impact of TOU prices on the power system operation within a unit scheduling problem for temperature control appliances; similarly, Heydarian-Forushani et al. [57] aim at determining the optimal TOU prices to ensure demand-side flexibility, while also optimizing the supply-side operations for a more secure and sustainable power grid.

1.4 Structure of the thesis

The remainder of this thesis is organized as follows. Chapter 2 introduces the problem and provides two mathematical models. The first one is based on a time-indexed formulation of the problem. The second model builds upon the first one, and exploits inherent symmetries of the solutions space so as to achieve a significant improvement in formulation compactness. Chapter 3 introduces an effective heuristic scheme, as well as an exact algorithm that heavily relies on the mathematical models, to tackle the hardness of the problem. Chapter 4 shows the numerical results obtained with the solution approaches presented in the thesis on a large test benchmark. Chapter 5 concludes the thesis by showing future research directions for a broad class of TOU scheduling problems. Appendix A reports notation for classical scheduling that is useful for TOU scheduling as well. Moreover, it reports the acronyms of the discrete optimization algorithms used in the literature of TOU scheduling, and presented in this chapter. Appendix B presents a compendium of the main mathematical symbols introduced in the definitions throughout the thesis. Finally, Appendix C reports a brief summary of the main notions, definitions, and algorithms introduced in thesis, so as to provide a convenient reference.

Chapter 2

The scheduling problem

The chapter begins with the introduction of the problem at the core of the thesis in Section 2.1. Subsequently, it provides two different mathematical models in Section 2.2. Specifically, after the presentation of the first mathematical model, it describes a characterizing property of the solutions space that enables a compact reformulation of the model. The chapter concludes with an in-depth comparison of the two models aimed at laying the foundations of the exact approach.

2.1 Problem statement

Let $\mathcal{J} = \{1, \ldots, N\}$ be the set of jobs, $\mathcal{H} = \{1, \ldots, M\}$ the set of identical machines, and $\mathcal{T} = \{1, \ldots, K\}$ the set of available time slots. Jobs are non-preemptible, and are characterized by an integer *processing time* $p_j \leq K$, $j \in \mathcal{J}$, corresponding to an integer number of distinct time slots. Machines are endowed with an *energy consumption rate* $u_h \geq 0$, $h \in \mathcal{H}$. Moreover, a non-negative cost $c_t \geq 0$, $t \in \mathcal{T}$, is associated with each time slot. An *assignment* of a job $j \in \mathcal{J}$ to a subset $\mathcal{T}_j \subseteq \mathcal{T}$ of p_j time slots on machine $h \in \mathcal{H}$ entails the processing of j during the time slots in $\mathcal{T}_j$ by machine h. In such case, job j is *scheduled* in the time slots in $\mathcal{T}_j$ on machine h. Then, we recall that a schedule

$$\mathcal{S} = \{(j, h_j, \mathcal{T}_j) : h_j \in \mathcal{H}, \mathcal{T}_j \subseteq \mathcal{T}, \forall j \in \mathcal{J}\} \tag{2.1}$$

is a set of the assignments of the jobs in $\mathcal{J}$ such that each $j \in \mathcal{J}$ 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, then the schedule $\mathcal{S}$ is

feasible. In this section, whenever $\mathcal{S}$ is referred to simply as a schedule, it is implied that $\mathcal{S}$ is feasible.

The *completion time* of a job $j \in \mathcal{J}$ is the largest time slot in $\mathcal{T}_j$, i.e., $C_j(\mathcal{S}) = \max_{t \in \mathcal{T}_j} t$, $j \in \mathcal{J}$. Then, the makespan $C^{\max}$ of a schedule $\mathcal{S}$ is the largest among the completion times of the jobs in $\mathcal{J}$, i.e.,

$$C^{\max}(\mathcal{S}) = \max\{C_j(\mathcal{S}), j \in \mathcal{J}\}. \tag{2.2}$$

The energy cost associated with the processing of job j on machine h_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

$$E(\mathcal{S}) = \sum_{j \in \mathcal{J}} \sum_{h \in \mathcal{H}} u_{h_j} \sum_{t \in \mathcal{T}_j} c_t. \tag{2.3}$$

Then, the *Bi-objective Identical Parallel Machine Scheduling with Time-of-Use Costs Problem* (BPMSTP) consists in finding a schedule $\mathcal{S}$ that simultaneously minimizes (2.2) and (2.3). The BPMSTP can be stated as $Pm||C^{max}, TEC$ by means of the $\alpha|\beta|\gamma$ notation. Observe that since both $Pm||C^{max}$ [46] and $1||TEC$ [19] are strongly $\mathcal{NP}$-hard, the BPMSTP is strongly $\mathcal{NP}$-hard as well.

The ordered tuple $(\mathcal{J}, \{p_j, j \in \mathcal{J}\}, \mathcal{H}, \{u_h, h \in \mathcal{H}\}, \mathcal{T}, \{c_t, t \in \mathcal{T}\})$ will be referred to as an *instance* $\mathcal{I}$ of the BPMSTP. Since a schedule $\mathcal{S}$ is a feasible solution to $\mathcal{I}$, the expressions "schedule" and "feasible solution" will be used interchangeably in the rest of the chapter. Moreover, to avoid burdening the notation, the dependence of C_j, $C^{\max}$ and E on $\mathcal{S}$ is omitted from now on. As a final remark, this thesis will also consider the BPMSTP from a practical standpoint, so as to comply with the needs of the decision makers in manufacturing industry as well. Specifically, finding all the Pareto-optimal solutions to $\mathcal{I}$ will be a further purpose of the thesis, and the following chapters will be developed accordingly.

2.2 Mathematical models

Let us describe the first mixed-integer programming model for the BPMSTP [9], referred to as "Formulation 1". To this end, let us denote by $x_{j,h,t} \in \{0, 1\}$, $j \in \mathcal{J}$, $h \in \mathcal{H}$, $t \in \mathcal{T}$, a decision variable that is equal to 1 if t is the first time slot of job j on machine h, and 0 otherwise. Moreover, let us express the makespan in (2.2) and the TEC in (2.3) with

the decision variables $C^{max} \in \mathbb{N}$ and $E \geq 0$, respectively.

Formulation 1.

$$\min C^{\max}, \tag{2.4}$$

$$\min E, \tag{2.5}$$

subject to

$$E = \sum_{h \in \mathcal{H}} u_h \sum_{j \in \mathcal{J}} \sum_{t=1}^{K-p_j+1} x_{j,h,t} \left(\sum_{i=t}^{t+p_j-1} c_i \right), \tag{2.6}$$

$$\sum_{h \in \mathcal{H}} \sum_{t=1}^{K-p_j+1} x_{j,h,t} = 1, \quad j \in \mathcal{J}, \tag{2.7}$$

$$\sum_{j \in \mathcal{J}} \sum_{i=\max\{1,t-p_j+1\}}^{t} x_{j,h,i} \leq 1, \quad h \in \mathcal{H}, t \in \mathcal{T}, \tag{2.8}$$

$$\sum_{h \in \mathcal{H}} \sum_{t=1}^{K-p_j+1} (t + p_j - 1) x_{j,h,t} \leq C^{\max}, \quad j \in \mathcal{J}, \tag{2.9}$$

$$C^{\max} \leq K, \tag{2.10}$$

$$C^{\max} \geq 0, \quad E \geq 0, \quad x_{j,h,t} \in \{0,1\}, j \in \mathcal{J}, h \in \mathcal{H}, t \in \mathcal{T}. \tag{2.11}$$

The objectives (2.4) and (2.5) account for the minimization of the makespan and the TEC, respectively, according to the definition of the makespan introduced in (2.2) and of the TEC in (2.6). Constraints (2.7) impose that each job $j \in \mathcal{J}$ starts in a single slot on a single machine. Constraints (2.8) avoid that more than one job is processed in the same time slot on the same machine. The left-hand side of (2.9) 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 (2.10). Lastly, (2.11) 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 (2.6)–(2.10). The main drawback of Formulation 1 lies in the number of decision variables that may become large as the size of the BPMSTP instances increases and, above all, presents many equivalent solutions.

Next, we introduce a novel model that overcomes the above limitations. Toward this

end, we first formally define the concept of equivalence. Let

$$\mathcal{P}_{\mathcal{J}'} \coloneqq \{d : \exists\, j \in \mathcal{J}', p_j = d\} \tag{2.12}$$

be the set of distinct processing times of the jobs in $\mathcal{J}' \subseteq \mathcal{J}$. Moreover, let

$$\mathcal{J}_d \coloneqq \{j : j \in \mathcal{J}, p_j = d\}, \quad d \in \mathcal{P} \tag{2.13}$$

be the subset of jobs with processing time equal to d. 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, we show that the BPMSTP may present exponentially many equivalent solutions.

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

Proof. Let the schedule $\mathcal{S}$ be given as in (2.1). Let also $\mathcal{Z} = \{\{h_j, \mathcal{T}_j\}, j \in \mathcal{J}\}$ be the set of all distinct unordered pairs $\{h_j, \mathcal{T}_j\}$ 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 $\mathcal{S}$. Observe that $\mathcal{Z}$ can be rewritten as $\bigcup_{d\in\mathcal{P}_\mathcal{J}} \mathcal{Z}_d$, 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 of time slots, all the assignments of the jobs in $\mathcal{J}_d$ to the elements of $\mathcal{Z}_d$, for each $d \in \mathcal{P}_\mathcal{J}$, 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 of permutations of the jobs in $\mathcal{J}_d$, i.e., $|\mathcal{J}_d|!$, then the distinct number of assignments of the jobs in $\mathcal{J}$ to $\mathcal{Z}$ is the product of $|\mathcal{J}_d|!$ for each $d \in \mathcal{P}_\mathcal{J}$. Observing that the schedule $\mathcal{S}$ is one of such assignments concludes the proof. □

Finally, let $b_{d,t} = \sum_{k=t}^{t+d-1} c_k$, $d \in \mathcal{P}_\mathcal{J}$, $t = 1, \ldots, K - d + 1$, be the cumulative cost associated with the time slots $t, t+1, \ldots, t+d-1$. 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 $y_{d,h,t} \in \{0, 1\}$, $d \in \mathcal{P}_\mathcal{J}$, $h \in \mathcal{H}$, $t \in \mathcal{T}$, be a binary decision variable that is equal to 1 if t is the start time of a job with processing time equal to d on machine h, and 0 otherwise.

Formulation 2.

$$\min C^{\max}, \tag{2.14}$$

$$\min E, \tag{2.15}$$

subject to

$$E = \sum_{h\in\mathcal{H}} u_h \sum_{d\in\mathcal{P}_\mathcal{J}} \sum_{t=1}^{K-d+1} b_{d,t}\, y_{d,h,t}, \tag{2.16}$$

$$\sum_{h\in\mathcal{H}} \sum_{t=1}^{K-d+1} y_{d,h,t} = |\mathcal{J}_d|, \quad d \in \mathcal{P}_\mathcal{J}, \tag{2.17}$$

$$\sum_{d\in\mathcal{P}_\mathcal{J}} \sum_{i=\max\{1,t-d+1\}}^{t} y_{d,h,i} \leq 1, \quad h \in \mathcal{H}, t \in \mathcal{T}, \tag{2.18}$$

$$(t+d-1)\, y_{d,h,t} \leq C^{\max}, \quad d \in \mathcal{P}_\mathcal{J}, h \in \mathcal{H}, t = 1,\ldots,K-d+1, \tag{2.19}$$

$$C^{\max} \leq K, \tag{2.20}$$

$$C^{\max} \geq 0, \quad E \geq 0, \quad y_{d,h,t} \in \{0,1\}, d \in \mathcal{P}_\mathcal{J}, h \in \mathcal{H}, t \in \mathcal{T}. \tag{2.21}$$

The objectives (2.14) and (2.15) account for the minimization of the makespan and the TEC, respectively, with the TEC here given by (2.16). Constraints (2.17) 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. Equation (2.18) guarantees that, on each machine, at most a single job is processed in a time slot. The left-hand side of (2.19) 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 (2.20). Lastly, (2.21) defines the domain of the decision variables.

Each feasible solution of Formulation 2 defines a class of equivalent schedules. Indeed, Formulation 2 guarantees that, for each $y_{d,h,t} = 1$, a job $j \in \mathcal{J}$ with processing time d is non-preemptively scheduled in the slots $t, t+1, \ldots, t+d-1$ on machine h. Formulation 2 also ensures that each job $j \in \mathcal{J}$ is scheduled once and only once.

Algorithm 1 generates a possible schedule out of the class of schedules defined by a solution of Formulation 2. In more detail, Algorithm 1 first initializes the schedule $\mathcal{S}$ to the empty set at line 1 and the sets needed for subsequent computations at 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 (see lines 5–9). Finally, the computed schedule $\mathcal{S}$

Algorithm 1 Generate-Schedule

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

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

is returned at line 9. At the end of the algorithm, $\mathcal{J}'_d = \emptyset$ for each $d \in \mathcal{P}_{\mathcal{J}}$, all the jobs in $\mathcal{J}$ are assigned, and there are not slots on the same machine assigned to more than one job.

Formulation 2 employs $|\mathcal{P}_{\mathcal{J}}|MK + 2$ decision variables and $|\mathcal{P}_{\mathcal{J}}| + MK + |\mathcal{P}_{\mathcal{J}}|M \sum_{d \in \mathcal{P}_{\mathcal{J}}}(K - d + 1) + 2$ constraints. The former number is due to the $|\mathcal{P}_{\mathcal{J}}|MK$ 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 (2.16)–(2.20). Let us finally compare the number of variables needed by Formulation 1 and Formulation 2. The worst case for Formulation 2 occurs when $|\mathcal{P}_{\mathcal{J}}| = N$, 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 as 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 $|\mathcal{P}_{\mathcal{J}}| = 1$. In such case, Formulation 1 is still characterized by $NMK + 2$ decision variables, while Formulation 2 has only $MK + 2$ variables. Thus Formulation 2 uses less decision variables than Formulation 1, except for the case $|\mathcal{P}_{\mathcal{J}}| = N$ when the two models are equivalent.

Let us now better characterize the worst case for the number of decision variables of Formulation 2. To this end, 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.,

$$N \leq \sum_{j \in \mathcal{J}} p_j \leq MK, \tag{2.22}$$

where the equality $\sum_{j\in\mathcal{J}} p_j = N$ holds when $p_j = 1$ for each $j \in \mathcal{J}$. Let us then formulate the following stronger necessary condition for feasibility by building upon (2.22).

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:*

$$|\mathcal{P}_{\mathcal{J}}| \leq \left\lfloor \frac{1+\sqrt{1+8MK}}{2} \right\rfloor. \tag{2.23}$$

Proof. First,

$$\sum_{j\in\mathcal{J}} p_j = \sum_{d\in\mathcal{P}_{\mathcal{J}}} |\mathcal{J}_d| d \geq \sum_{i=1}^{|\mathcal{P}_{\mathcal{J}}|} i = \frac{|\mathcal{P}_{\mathcal{J}}|(|\mathcal{P}_{\mathcal{J}}|+1)}{2} \tag{2.24}$$

since $|\mathcal{J}_d| \geq 1$ and the elements in $\mathcal{P}_{\mathcal{J}}$ are pairwise distinct positive integers. By combining (2.22) with (2.24), we obtain

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

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

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

□

Observe that since (2.22) and Proposition 1 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 infeasible solutions with respect to (2.22), and therefore it enables to avoid solving several instances for Formulation 2 by simply checking the validity of (2.23) beforehand.

In order to illustrate how Proposition 1 allows a better description of the worst case of Formulation 2 as regards the number of decision variables, let us 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 correspond to the case $p_j = 1$ for all $j \in \mathcal{J}$, and it is equal to $MK = 2 \cdot 10^3$, owing to (2.22). In this case, the number of decision variables of Formulation 1 is $4 \cdot 10^6 + 2$, whereas it is equal to $2 \cdot 10^3 + 2$ for Formulation 2 since $|\mathcal{P}| = 1$. Observe that, for such an instance, condition (2.23) 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 $N \leq \lfloor (1 + \sqrt{16001}) / 2 \rfloor = 63$. Hence, in order for the considered instance to be possibly feasible, the number of the variables has to be no greater than $1.26 \cdot 10^5 + 2$. The necessary condition (2.22) would instead provide the higher upper bound $M^2K^2 + 2 = 4 \cdot 10^6 + 2$.

To conclude the comparison of Formulation 1 and Formulation 2, observe that Formulation 2 requires a larger number of constraints than Formulation 1 (compare equations (2.19) and (2.9)). However, as it will be discussed in Chapter 3, these constraints can be neglected in the framework of the developed exact algorithm. Moreover, Chapter 4 reports the significantly lower computational effort required to solve Formulation 2 with respect to Formulation 1 in all the considered experimental tests.

Chapter 3

Algorithms

This chapter describes the different original solution approaches for the BPMSTP. First, it introduces Split-Greedy Heuristic (SGH) in Section 3.1. SGH enhances the heuristic algorithm proposed by Wang et al. [108] by considering a larger space of greedy decisions. SGH is used as a sub-routine by the Split-Greedy Scheduler (SGS) algorithm, described in Section 3.3, so as to find all the Pareto-optimal solutions to the problem. Next, Section 3.2 introduces Exchange Search (ES) [9], which is a local search algorithm that uses improving moves enabled by inherent characteristics of the BPMSTP. Lastly, this chapter concludes by describing the complete heuristic scheme used to solve the BPMSTP, called Split-Greedy Scheduler (SGS-ES), which is a natural extension of SGS obtained by combining SGH and ES.

The main contributions of this section are the introduction of SGH and ES. Specifically, SGH enables to increase the quality and the number of the feasible heuristic solutions with respect to the heuristic algorithm of Wang et al.. The idea at the core of SGH is to greedily assign jobs in order to minimize the TEC, while possibly allowing particular violations to the non-preemption constraint. It is however shown that such an infeasible solution can always be converted into a feasible and equivalent one. ES searches for specific improving moves so as to improve the TEC of such a feasible schedule. Despite its computational burden, it proves very useful in improving the outcomes of the greedy choices performed by SGH.

3.1 Split-Greedy Heuristic

Let us first introduce some useful definitions. Two slots t and $t+k$, $1 \leq t \leq K-1$, on a machine $h \in \mathcal{H}$ are *adjacent* if $k = 1$. Moreover, for a given schedule $\mathcal{S}$, a slot $t \in \mathcal{T}$ on machine $h \in \mathcal{H}$ is *free*, or *idle*, if no job is assigned to t on h.

Definition 1. (Free-consecutive slots) *For a given schedule $\mathcal{S}$, the slots t and $t+k$, $1 \leq t \leq t+2 \leq t+k \leq K$, are* **free-consecutive** *on machine $h \in \mathcal{H}$ if t and $t+k$ are free, and slots $t+1, t+2, \ldots, t+k-1$ are not free in $\mathcal{S}$.*

Let us refer to a pair $l = (h, \mathcal{F})$, with $h \in \mathcal{H}, \mathcal{F} \subseteq \mathcal{T}$, as a *location*. Furthermore, let $u_h \sum_{t \in F} c_t$ be the *energy cost* of location l. Let us consider the example in Figure 3. The sets of slots with different shades of grey correspond to the locations $(h, \{2,3,4\})$, $(h, \{6,7\})$, and $(h, \{8\})$, from the lightest to the darkest shade, with energy costs 10, 12, and 13, respectively.

Definition 2. (Free location) *For a given schedule $\mathcal{S}$, a* **free location** *for a job j is a location $(h, \mathcal{F})$ such that $\mathcal{F}$ contains only free adjacent and/or free-consecutive slots on machine h in $\mathcal{S}$, and $|\mathcal{F}| = p_j$.*

Figure 4 reports an example of $K = 6$ slots on a machine h where a single job j with processing time $p_j = 2$ is scheduled. Slots 2 and 5 are free-consecutive slots on h. Furthermore, the location $(h, \mathcal{A})$, $\mathcal{A} = \{2,5\}$, is a free location for a job j' with processing time $p_{j'} = 2$ on h. It is also worth to observe that, if j' was assigned to $\mathcal{A}$, the resulting schedule would be infeasible due to the preemption of j'.

Definition 3. (Assigned location) *For a given schedule $\mathcal{S}$, if a job j is assigned to the set $\mathcal{S}_j$ of p_j adjacent slots on machine h_j in $\mathcal{S}$, then the location $(h_j, \mathcal{S}_j)$ is the* **assigned location** *of j.*

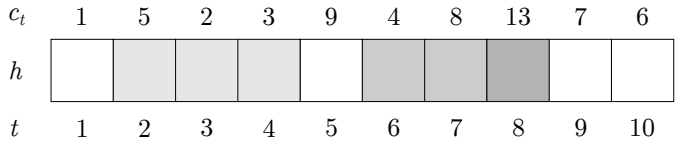

Figure 3: An example of three distinct locations on machine h, colored with different shades of grey. The location highlighted with the lightest shade is $(h, \{2,3,4\})$, the one with the intermediate shade is $(h, \{6,7\})$, while the one with the darkest shade is $(h, \{8\})$.

If $l_j = (h_j, \mathcal{S}_j)$ is the assigned location of j, we also simply say that j is assigned to l_j. In such case, the start time of j is $s_j \coloneqq min_{t \in \mathcal{S}_j}\{t\}$. In Figure 4, $(h, \{3, 4\})$ is the assigned location of job j.

Let us now describe the intuition underlying SGH and its novelty for the BPMSTP. Toward this end, let us first outline the heuristic proposed by Wang et al. [108]. The core of such heuristic consists in the sequential application of a constructive heuristic and a local search algorithm, which is iterated in order to find all the feasible optimal solutions to the problem. We refer to this heuristic as CH because Wang et al. [108] adopt this naming convention while presenting their computational results.

CH is based on the ϵ-constraint paradigm [56] for multi-objective optimization. The basic idea of such paradigm is to minimize (or maximize) one of the objectives while the other ones are constrained to be lower (or greater) than fixed values. Specifically, given an instance $\mathcal{I}$ of the BPMSTP, CH first (i) sets an upper bound on the makespan, and then (ii) minimizes the TEC. CH iterates over steps (1) and (2) by progressively reducing the upper bound, until a lower bound for the makespan is reached. In this way, CH finds a set of heuristic solutions to the problem that approximate the set of non-dominated points $\mathcal{I}$. Specifically, CH consists in the following steps:

1. the jobs in $\mathcal{J}$ are sorted according to the Longest Processing Time (LPT) rule, i.e., in non-increasing order of processing times;

2. the maximum makespan $K^{\max}$ for the current iteration is initialized to K, and its lower bound is set to $K^{min} = \sum_{j \in J} p_j / M$. Moreover, the set of computed solutions $\mathcal{N}_s$ is initialized to $\emptyset$;

3. if $K^{\max} < K^{min}$, then CH terminates since no feasible solution with makespan less than or equal to $K^{\min}$ exists. Otherwise, for each job $j \in \mathcal{J}$, CH determines the location l^{min} with the smallest cost among the set of free locations for j including only free adjacent slots not exceeding $K^{\max}$. Then, j is assigned to l^{min}. Possible ties are broken by choosing a location so that the start time of j is the smallest possible;

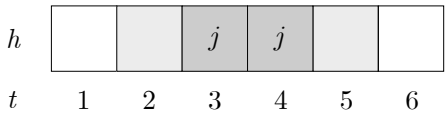

Figure 4: An example of free-consecutive and adjacent slots. Slots 2 and 5 are free-consecutive, while slots 3 and 4 are the adjacent assigned slots of job j.

4. a local search tries to improve the TEC of the solution computed at the previous step by shifting "blocks" of contiguous jobs on each machine in $\mathcal{H}$ without worsening the makespan. Specifically, for each $h \in \mathcal{H}$, such a local search explores a neighborhood of moves that can modify the schedule on h in an effort of improving the TEC, while preserving the processing sequence of the jobs on h;

5. the makespan $C^{\max}$ and the TEC of the solution improved at the previous step are computed. Then, the solution is added to $\mathcal{N}_s$. Furthermore, the maximum makespan is updated as $K^{\max} \leftarrow C^{\max} - 1$, and the algorithm's control flow returns to step 3. Observe that, after the update, the condition $K^{\max} \geq K^{min}$ at step 3 does not hold if there are no other feasible solutions with makespan less than $C^{\max}$;

6. the set of the non-dominated solutions in $\mathcal{N}_s$ is returned.

It is assumed that the ties at step 1 are broken randomly, as there is no specific indication in this regard by the authors.

Observe that, in general, there may not be a feasible solution for some $K^{\max} \geq K^{min}$, as K^{min} is not a tight lower bound for the makespan. Hence, the solution computed at step (3) might be infeasible. However, CH does not verify its feasibility. As a consequence, CH may return a solution set that contains infeasible schedules. Moreover, according to the original description in [108], CH may not be able to build feasible schedules, even though a feasible schedule exists, when no location including only free adjacent slots is available for a job. Specifically, CH may return an infeasible schedule as a solution to the problem without reporting its infeasibility. Example 1 presents an instance that elicits such behavior.

Example 1. Let us consider two machines $h, h' \in \mathcal{H}$ with energy consumption rate $u_h = 1$ and $u_{h'} = 2$, and a set $\hat{\mathcal{J}} = \{j^{(i)}, i = 0, 1, \ldots, 5\}$ of jobs, with processing time $p_j = 2$ for each $j \in \hat{\mathcal{J}}$. The number K of time slots is 7, and for $t = 1, \ldots, 7$, the energy costs c_t are $10, 1, 1, 10, 1, 1, 10$, respectively. Since the jobs in $\hat{\mathcal{J}}$ have the same processing time, CH considers the jobs in $\hat{\mathcal{J}}$ in random order. Figure 5(a) shows one among the possible smallest cost alternative assignments of jobs $j^{(0)}$, $j^{(1)}$, $j^{(2)}$ and $j^{(3)}$ performed by CH. However, after such assignment, there is no valid location left for $j^{(4)}$ and $j^{(5)}$. Hence, CH stops iterating over $\hat{\mathcal{J}}$, and considers such partial (i.e., infeasible) schedule as a valid return value (i.e., a feasible solution). Furthermore, as shown in Figure 5(b), a feasible solution however actually exists for the problem.

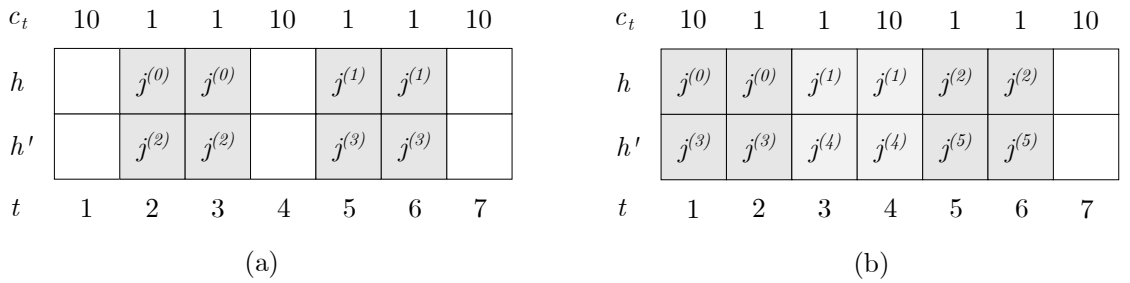

Figure 5: An example of the shortcomings of CH while solving specific instances. Figure 5(a) shows the partial schedule generated by CH for Example 1. Instead, Figure 5(b) shows a feasible schedule for the same example.

In order to overcome this issue, SGH is able to temporarily assign jobs to free locations that, according to Definition 2, may include free-consecutive slots. Let us formalize this intuitive notion.

Definition 4. (Assigned-consecutive slots) *For a given schedule $\mathcal{S}$, the slots t and $t+k$, $1 \leq t \leq t+2 \leq t+k \leq K$, are* **assigned-consecutive** *on machine $h \in \mathcal{H}$ if t and $t+k$ are assigned slots of some job $j \in \mathcal{J}$, and $t+1, t+2, \ldots, t+k-1$ are assigned slots of a subset of jobs of $\mathcal{J} \setminus \{j\}$ in $\mathcal{S}$.*

Definition 5. (Split-location) *For a given schedule $\mathcal{S}$, a* **split-location** *is a free location with at least two free-consecutive slots, or an assigned location with at least two assigned-consecutive slots, in $\mathcal{S}$. A split-location is* **free** *for a job j if it is also a free location for j. A split-location $(h_j, \mathcal{T}_j)$ is* **assigned** *if there is a job $j \in \mathcal{J}$ such that j is assigned to all and only the slots in $\mathcal{T}_j$.*

A job assigned to a split-location l is *split-scheduled* in l. In Figure 6, $j^{(0)}$ is assigned to the location $(h, \{1, 2, 6\})$, which is a split-location since slots 1 and 2 are adjacent, and slots 2 and 6 are free-consecutive.

Definition 6. (Split-schedule) *A* **split-schedule** *$\mathcal{S}$ is a schedule as in* (2.1) *such that, for some $j \in \mathcal{J}$, $(h_j, \mathcal{T}_j)$ is a split-location.*

Observe that the schedule in Figure 6 is a split-schedule. In fact, as opposed to feasible schedules, in a split-schedule at least one job is assigned to a split-location, according to

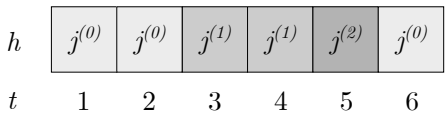

Figure 6: An example of a split-location. The location $(h, \{1, 2, 6\})$ is the assigned split-location of job j.

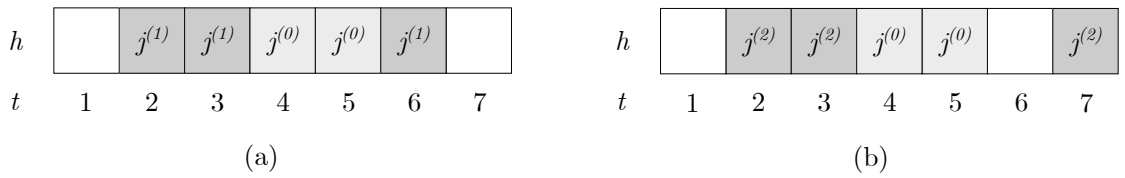

Figure 7: An example of a split-schedule (a), and a preemptive schedule which is not a split-schedule (b). In particular, the schedule in (a) is a split-schedule since $j^{(0)}$ is assigned to adjacent slots, while $j^{(1)}$ is assigned both to adjacent slots and free-consecutive slots. In the schedule in (b), $j^{(0)}$ is assigned to the same adjacent slots, but two of the assigned slots of $j^{(1)}$, i.e., slots 3 and 7, are neither adjacent, nor free-consecutive.

Definition 6. Hence, split-schedules are not feasible and they are a subset of preemptive schedules. Figure 7 highlights the difference between a split-schedule and a preemptive schedule which is not a split-schedule. In more detail, Figure 7(a) shows a split-schedule with a split-scheduled job $j^{(1)}$, whereas Figure 7(b) shows a schedule with a preempted job $j^{(2)}$. Such a job is not split-scheduled since slots 3 and 7 are not assigned-consecutive as slot 6 is free.

Definition 7. (Schedule block) *For a given schedule $\mathcal{S}$, a* **schedule block** *$\mathcal{B}$ on a machine $h \in \mathcal{H}$ of $\mathcal{S}$ is a schedule which involves only subsets of non-free slots that are delimited by two free slots. Formally, $\mathcal{B}$ is a set $\{\{j, h, \mathcal{G}_j\},\ j \in \mathcal{J}' \subseteq \mathcal{J}\}$, such that $\bigcup_{j\in\mathcal{J}'} \mathcal{G}_j$ is equal to a subset $\{i, i+1, \ldots, i+b-1\} \subseteq \mathcal{T}$ of $b \geq 0$ assigned adjacent slots on h and slots $i-1$ and $i+b$, if they are in $\mathcal{T}$, are free. If $\mathcal{B}$ contains at least a split-location, the block is called a split-schedule block; otherwise, it is called a feasible schedule block.*

A split-schedule block identifies a set of consecutive time slots that are assigned to split-scheduled jobs in the block. An example of a split-schedule block and a feasible schedule block is given in Figure 8. The following proposition guarantees the correctness of SGH.

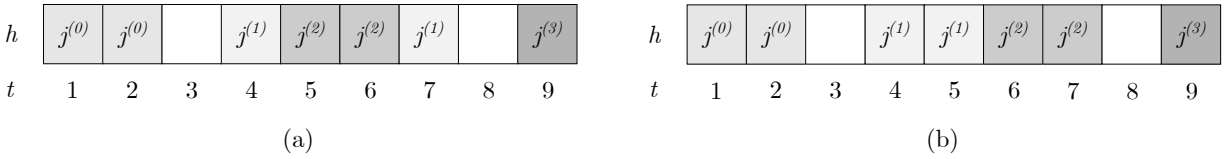

Figure 8: An example of split-schedule block and feasible schedule block. In Figure 8(a), slots 3, 4, 5, and 6 are in a split-schedule block involving jobs $j^{(1)}$ and $j^{(2)}$. Instead, in Figure 8(b), such slots are in a feasible schedule block involving the same two jobs.

Proposition 2. *Let $\mathcal{I}$ be an instance of the BPMSTP. Then, for each split-schedule $\mathcal{S}$ of $\mathcal{I}$, there is a feasible schedule $\mathcal{S}'$ equivalent to $\mathcal{S}$.*

Proof. Let us consider the schedule $\mathcal{S}_h \subseteq \mathcal{S}$ on each machine $h \in \mathcal{H}$. A split-schedule $\mathcal{S}_h$ on a machine can be partitioned into $b \geq 0$ split-schedule blocks $\mathcal{B}_{h,i}$, $i = 1, \ldots, b$. For each $h \in \mathcal{H}$, neither the makespan or the TEC of $\mathcal{S}_h$ is affected by different processing sequences of the jobs within each block $\mathcal{B}_{h,i}$, since there are no free slots in any of them. Indeed, let us observe that each split-schedule block is a solution of an instance $\tilde{\mathcal{I}}$ of $1 \,|\, prmp \,|\, C^{\max}, TEC$. Moreover, let us recall the fact that for each such preemptive solution, say, $\mathcal{Z}$ of $\tilde{\mathcal{I}}$, there is a non-preemptive solution $\mathcal{Z}'$ of $\tilde{\mathcal{I}}$ with the same makespan of $\mathcal{Z}$. Then, for each split-schedule block, there is an equivalent feasible schedule block with a different jobs processing sequence. Observing that $C^{\max}(\mathcal{S}) = \max_{h \in \mathcal{H}} C^{\max}(\mathcal{S}_h)$ and $E(\mathcal{S}) = \sum_{h \in \mathcal{H}} E(\mathcal{S}_h)$ concludes the proof. □

Proposition 2 ensures the possibility of converting any split-schedule into an equivalent feasible schedule. Algorithm 2 performs this task. Specifically, among the alternative equivalent feasible schedules that can be obtained from a split-schedule $\mathcal{S}$, Algorithm 2 generates the feasible schedule $\mathcal{S}$ such that, for each j and j' in $\mathcal{J}$ scheduled on the same machine, the inequality relation between the start times s_j and s'_j of j and j' in $\mathcal{S}$ holds in $\mathcal{S}'$ as well. The efficiency of Algorithm 2 lies in the fact that it considers each job, and possibly changes its assignment, at most once.

Towards the end of describing Algorithm 2, as well as the algorithms in the rest of the chapter, let

$$\mathcal{D}(\hat{K}) = \{(\mathcal{J}, \{p_j, j \in \mathcal{J}\}, \mathcal{H}, \{u_h, h \in \mathcal{H}\}, \hat{\mathcal{T}} = \{1, 2, \ldots, \hat{K}\} \subseteq \mathcal{T}, \{c_t, t \in \hat{\mathcal{T}}\})\} \quad (3.1)$$

be an instance of the BPMSTP for a given positive integer $\hat{K} \leq K$. Intuitively, instance (3.1) disregards all the time slots t such that $\hat{K} < t \leq K$. Algorithm 2 takes a solution $\mathcal{S}$ for an instance $\mathcal{D}(\hat{K})$ as in (3.1) as an input and returns a feasible schedule $\mathcal{S}'$ equivalent to $\mathcal{S}$. For each machine $h \in \mathcal{H}$, Algorithm 2 iterates over the time slots in non-decreasing order to determine the start time s_j of each job j scheduled on machine h (lines 4–9) in the split-schedule $\mathcal{S}$, and as soon as it identifies the starting time of a job j, it assigns j to p_j consecutive time slots, starting from the first free slot greater than or equal to s_j (lines 5–7).

For the sake of computational efficiency, Left-Leaning Red-Black (LLRB) trees [93] are used to represent schedules in the implementation of Algorithm 2, as well as in the

Algorithm 2 Convert-Schedule

Input A (possibly infeasible) schedule $\mathcal{S}$ for a BPMSTP instance $\mathcal{D}(\hat{K})$ as in (3.1).
Output A feasible schedule $\mathcal{S}'$ equivalent to $\mathcal{S}$.

1: $\mathcal{S}' \leftarrow \{\mathcal{S}'_h,\ h \in \mathcal{H}\}$, a collection of empty sets $\mathcal{S}'_h = \emptyset, h \in \mathcal{H}$
2: **for each** $h \in H$ **do**
3: $s \leftarrow 0,\ k \leftarrow 1$
4: **while** it exists a job j such that $\hat{s}_j = \min\{s_i : i \in \mathcal{J}, (i, h, \mathcal{T}_i) \in \mathcal{S}_h, s_i > s\}$ **do**
5: $s'_j \leftarrow \max\{k, \hat{s}_j\}$ *// The actual start time of j in the converted schedule*
6: $l' \leftarrow \{h, \{s'_j, s'_j + 1, ..., s'_j + p_j - 1\}\}$
7: Reassign j to l' by adding $(j, h, \{s'_j, s'_j + 1, ..., s'_j + p_j - 1\})$ to $\mathcal{S}'_h$
8: $s \leftarrow \hat{s}_j,\ k \leftarrow s'_j + p_j$
9: **end while**
10: **end for**
11: **return** $\mathcal{S}'$

implementations of the other algorithms presented in this section. The computational complexity of such algorithms is analysed accordingly. LLRBs are based on Red-Black trees [30], which are a type of self-balancing Binary Search Tree (BST). BSTs are typically used to represent ordered sets so as to enable logarithmic-time worst-case computational complexity for retrieval, insertion, and deletion operations. With such data structures, a schedule is stored as a set of M LLRBs, each of them associated with a distinct machine in $\mathcal{H}$. In more detail, each LLRB stores the start times of the jobs as the keys of the tree, augmenting each node with the location of the related job as additional information. For each $h \in \mathcal{H}$, lines $4 - 9$ are executed $O(\Omega)$ times, where

$$\Omega = \min\{N, \hat{K}\},$$

since the maximum number of jobs scheduled on a machine is N, and it is upper bounded by the number of time slots $\hat{K}$ in $\mathcal{D}(\hat{K})$. Lines 3 and 5 both take $O(1)$. The extraction of the entry with the lowest key among the ones with key greater than a given one from a LLRB (line 4) consisting of n elements is $O(log_2\, n)$. The insertion operation on a LLRB (line 7) has the same computational complexity. Therefore, lines 4 and 7 have a $O(log_2\, \Omega)$ complexity as considered in an iteration of lines 4–9. In the pseudo-code, line 6 is stated consistently with the notation of locations introduced earlier in this section. However, since the slots $s'_j, s'_j + 1, ..., s'_j + p_j - 1$ are consecutive, line 6 can be easily implemented to store only the starting time s'_j and the end time $s'_j + p_j - 1$ of each job $j \in \mathcal{J}$. Therefore, the complexity of line 6 is $O(1)$. Since lines 2–10 are executed M times, the worst-case

computational complexity of Algorithm 2 is $O(M\Omega \log_2 \Omega)$.

Finally, let us proceed to describe SGH in detail. SGH constructively computes a heuristic schedule for a BPMSTP instance by minimizing the TEC while disregarding the makespan, similarly to step 3) of CH. However, as opposed to it, SGH:

(a) returns no feasible solution if $K^{\max}$ is too tight for SGH to schedule all the jobs in $\mathcal{J}$;

(b) it temporarily allows the generation of split-schedule blocks while scheduling the jobs in $\mathcal{J}$;

(c) it finally converts any split-schedule block obtained with previous computations into an equivalent feasible one by means of Algorithm 2.

The pseudo-code of SGH is reported in Algorithm 3. Let us first describe the local variables used by Algorithm 3 for its computations. The set $\mathcal{S}$ is a schedule that is partitioned into M schedules $\mathcal{S}_h$, $h \in \mathcal{H}$, which are empty at line 1, and they are progressively filled as jobs are assigned to locations. Instead, $P_{\mathcal{J}}$ is a list that, after its initialization at line 2, contains the distinct processing times of the jobs in $\mathcal{J}$, i.e., the elements of $\mathcal{P}_{\mathcal{J}}$, in decreasing order. The set $\mathcal{J}_d$, $d \in \mathcal{P}_{\mathcal{J}}$, initialized at line 6, is a subset of $\mathcal{J}$ that contains jobs with processing time equal to d, according to (2.12). Finally, $L_{d,h}$, $d \in \mathcal{P}_{\mathcal{J}}$, $h \in \mathcal{H}$, initialized at line 6, is a list of the free locations for a job with processing time d on machine h, which is progressively updated as jobs are assigned to the locations.

Algorithm 3 initializes $\mathcal{S}$ as a set of empty sets at line 1. Then, at line 2, it computes a list $P_{\mathcal{J}}$ out of $\mathcal{P}_{\mathcal{J}}$, and then it sorts $P_{\mathcal{J}}$ in non-decreasing order. At line 3, it initializes $\mathcal{J}_d$ by using $\mathcal{J}$ for each $d \in \mathcal{P}_{\mathcal{J}}$ according to (2.13). At lines 4–15, Algorithm 3 computes a (possibly split-) schedule by assigning the jobs in $\mathcal{J}_d$ for each $d \in \mathcal{P}_{\mathcal{J}}$. Specifically, for each $d \in \mathcal{P}_{\mathcal{J}}$, it first builds, for each $h \in \mathcal{H}$, the list $L_{d,h}$ of all the available, free (possibly split-) locations with the smallest cost for a job with processing time d on h (lines 5–7). Then, it assigns the jobs in $\mathcal{J}_d$ at lines 8–14 as follows. For each $j \in \mathcal{J}_d$, it checks if there is at least a free location left for j in $\bigcup_{h \in H}\{l : l \in L_{d,h}\}$ at line 9. If such a location does not exist, it returns an empty schedule. Otherwise, it performs a random selection among the possible locations with the smallest cost for j. The motivation behind this specific tie-breaking strategy lies in experimental observations. In fact, such random selection enabled to achieve better solutions with respect to the deterministic choice that instead favors the earliest starting location (implemented in CH). At line 11, Algorithm 3 updates

$\mathcal{S}_{\hat{h}}$ by adding $(j, \hat{h}, \hat{\mathcal{A}})$ as a result of assigning j to the location $(\hat{h}, \hat{\mathcal{A}})$. As a consequence, previously free locations are affected, and new ones may be generated. Specifically, the locations that contain at least a slot $t \in \hat{\mathcal{A}}$ should be removed from the list, since they are not free after the assignment of j. Intuitively, such assignment may also result in new free split-locations containing the slots that delimit $(\hat{h}, \hat{\mathcal{A}})$. At lines 12–13, Algorithm 3 updates $L_{d,\hat{h}}$ accordingly. Finally, if the resulting schedule $\mathcal{S}$ is a split-schedule, Algorithm 3 converts it into a feasible one in line 17 with Algorithm 2.

Let us now determine the computational complexity of Algorithm 3. The computational complexity of lines 1–3 is $O(M + |\mathcal{P}_{\mathcal{J}}|\, log_2|\mathcal{P}_{\mathcal{J}}| + N)$, since line 1 initializes M empty schedules as LLRBs, line 2 requires to sort the list $P_{\mathcal{J}}$, and line 3 iterates over the jobs in $\mathcal{J}$ in order to build $\mathcal{F}_d$ for each $d \in \mathcal{P}_{\mathcal{J}}$. Then, let us consider the computational complexity of an iteration of lines 5–14. Let us focus on lines 5–7. For each $h \in \mathcal{H}$, in order to identify all the free locations for each $j \in \mathcal{J}_d$ on h, line 6 iterates over the time slots in $\mathcal{T}$ by advancing a pointer until $\hat{K}$ is reached. A queue with maximum capacity d is employed to store the free slots reached by the pointer. As soon as a free slot t is found, t is enqueued until the maximum capacity is reached. When this happens, a new free (possibly split-) location is identified. The start and the end time of such location are respectively equal to the oldest and the most recent elements in the queue. Then, whenever a new free slot t is found, the oldest element in the queue is dequeued and t is enqueued, progressively identifying new locations. Querying the job scheduled in each slot, which consists in retrieving an element in a $\mathcal{S}_h$, $h \in \mathcal{H}$, requires $O(\log_2 \Omega)$ as the maximum number of jobs scheduled on each machine is Ω. Hence, lines 5–7 take $O(M\hat{K} \log_2 \Omega)$. Such complexity can be improved to $O(M\Omega \log_2 \Omega)$ by observing that, for each slot $t \in \mathcal{T}$, the smallest key larger than t in $\mathcal{S}_h$, $h \in \mathcal{H}$, i.e., the smallest start time of the jobs in $\mathcal{S}_h$ among the ones greater than t, can be retrieved in $O(\log_2 \Omega)$. In this way, $\mathcal{S}_h$ has to be queried at most Ω times, since the job with the smallest start time greater than t can be determined in logarithmic time. Consider the loop at lines 8–14. Line 9 takes $O(1)$. Line 10 performs the random removal of a location from a set of locations with the same cost on $O(M)$ machines. Hence, line 10 takes $O(M)$. Line 11 updates $\mathcal{S}_{\hat{h}}$ with the result of the new assignment, and since the number of elements in $\mathcal{S}_{\hat{h}}$ is $O(\Omega)$, line 11 takes $O(\log_2 \Omega)$. The update of $L_{d,\hat{h}}$ at lines 12–13 takes $O(\Omega \log_2 \Omega)$ (see the analysis of lines 5–7). Hence, all the iterations performed by lines 4–15 take

$$O(|\mathcal{P}_{\mathcal{J}}|M\Omega \log_2 \Omega + N(M + \Omega \log_2 \Omega))$$

Algorithm 3 Split-Greedy Heuristic (SGH)

Input A BPMSTP instance $\mathcal{D}(\hat{K})$ as in (3.1).
Output A feasible schedule for $\mathcal{D}(\hat{K})$, if it exists; otherwise, an empty schedule.

1: Let $\mathcal{S} \leftarrow \{\mathcal{S}_h,\, h \in \mathcal{H}\}$ be a collection of the empty sets $\mathcal{S}_h = \emptyset, h \in \mathcal{H}$
2: Let $P_{\mathcal{J}}$ be a list of the elements in $\mathcal{P}_{\mathcal{J}}$ in decreasing order
3: Let $\mathcal{J}_d \leftarrow \{j \in \mathcal{J} : p_j = d\}$, $d \in \mathcal{P}_{\mathcal{J}}$
4: **for** $d \in \mathcal{P}_{\mathcal{J}}$ **do**
5: **for** $h \in \mathcal{H}$ **do**
6: Let $L_{d,h}$ be a list of the free locations with smallest cost on h for any j, $p_j = d$
7: **end for**
8: **for** $j \in \mathcal{J}_d$ **do**
9: **if** $L_{d,h} = \emptyset, \forall h \in \mathcal{H}$ **then return** $\emptyset$ // No free location for j
10: Remove a smallest cost location $\hat{l} = (\hat{h}, \hat{\mathcal{A}})$ from $\bigcup_{h\in\mathcal{H}}\{l : l \in L_{d,h}\}$
11: Assign job j to $\hat{l}$ by adding $(j, \hat{h}, \hat{\mathcal{A}})$ to $\mathcal{S}_{\hat{h}}$
12: Remove any location or split-location $(\hat{h}, \mathcal{A}) : \mathcal{A} \cap \hat{\mathcal{A}} \neq \emptyset$ from $L_{d,\hat{h}}$
13: Add any split-location $(\hat{h}, \mathcal{A}')$ with $|\mathcal{A}'| = d$ to $L_{d,\hat{h}}$
14: **end for**
15: **end for**
16: **if** $\mathcal{S}$ is a split-schedule **then**
17: $\mathcal{S} \leftarrow$ Convert-Schedule($\mathcal{S}$) // Convert $\mathcal{S}$ with Algorithm 2
18: **end if**
19: **return** $\mathcal{S}$

Finally, lines 16–18 take $O(M\,\Omega \log_2\,\Omega)$, that is the complexity of Algorithm 2. Therefore, the complexity of Algorithm 3 is

$$\begin{aligned} &O(M + |\mathcal{P}_{\mathcal{J}}| \log_2 |\mathcal{P}_{\mathcal{J}}| + N) + O(|\mathcal{P}_{\mathcal{J}}| M \Omega \log_2 \Omega) + O(NM) \\ &+ O(N\Omega \log_2 \Omega) + O(M\Omega \log_2 \Omega) \end{aligned} \tag{3.2}$$

The first and the last term of the sum are the computational complexity of lines 1–3 and 16–18, respectively. Instead, the second, the third, and the fourth term account for all the iterations of lines 5–7, 9–10, 11, 12–14, respectively. Finally, (3.2) can be more compactly expressed as

$$O(\Omega(M|\mathcal{P}_{\mathcal{J}}| + N) \log_2 \Omega + NM + |\mathcal{P}_{\mathcal{J}}| \log_2 \mathcal{P}_{\mathcal{J}}). \tag{3.3}$$

In order to show that (3.3) is the most compact expression for the complexity of SGH,

let us rewrite it as

$$O(\Omega M|\mathcal{P}_{\mathcal{J}}|\log_2\Omega + \Omega N\log_2\Omega + NM + |\mathcal{P}_{\mathcal{J}}|\log_2\mathcal{P}_{\mathcal{J}}). \tag{3.4}$$

Let us consider the first and the second term of the sum in (3.4), i.e., $\Omega M|\mathcal{P}_{\mathcal{J}}|\log_2\Omega$ and $\Omega N\log_2\Omega$, respectively. Generally, $|P_{\mathcal{J}}| \leq N$, but $M \gtrless N$. Hence, the first and the second term are asymptotically equivalent, i.e., $\Theta(\Omega M|\mathcal{P}_{\mathcal{J}}|\log_2\Omega) = \Theta(\Omega N\log_2\Omega)$. Let us now consider the second and the third term of the sum, i.e., $\Omega N\log_2\Omega$ and NM, respectively. Since $\Omega\log_2\Omega \gtrless M$, then indeed $\Theta(\Omega N\log_2\Omega) = \Theta(NM)$. Finally, as regards the third and the fourth term of the sum, since $NM \gtrless |\mathcal{P}_{\mathcal{J}}|\log_2|\mathcal{P}_{\mathcal{J}}|$, then $\Theta(NM) = \Theta(|\mathcal{P}_{\mathcal{J}}|\log_2|\mathcal{P}_{\mathcal{J}}|)$ as well. Then, the four terms in the sum are asymptotically equivalent due to the transitive property of the Big Theta notation.

3.2 Exchange Search

This section introduces ES. First, it gives the intuition underlying ES. Then, it states the definitions needed to formally describe ES. Finally, the section presents ES by providing its pseudo-code and an expression of its computational complexity.

The purpose of ES is to improve the TEC of a feasible schedule $\mathcal{S}$. It strives to achieve so by changing the assignment of subsets of scheduled jobs to the machines in order to improve TEC while preserving the feasibility, and without worsening the makespan. In some cases, ES may however improve the makespan as a byproduct of TEC minimization.

Definition 8. (Exchangeable Period Sequence) *Let $\mathcal{S}$ be a schedule as in* (2.1) *for a BPMSTP instance $\mathcal{I}$. An* **Exchangeable Period Sequence** *(EPS) is a subset $\mathcal{E} \subseteq \mathcal{T}$ of adjacent time slots on a machine $h \in \mathcal{H}$ in $\mathcal{S}$ such that, if h processes some job $j \in \mathcal{J}$ during a time slot $t \in \mathcal{E}$ in $\mathcal{S}$, then $\mathcal{T}_j \subseteq \mathcal{E}$.*

Figure 9 provides an example of EPS. The subset of time slots $\mathcal{E}_1$ in Figure 9(a) is an EPS with cardinality equal to 5, since it contains two jobs with processing time equal to 2 entirely scheduled in $\mathcal{E}_1$, and an idle time slot. Differently, Figure 9(b) shows a subset of slots $\mathcal{E}_2$ on h that is not an EPS, since job $j^{(0)}$ is not completely scheduled within $\mathcal{E}_2$.

Definition 9. (EPS subschedule) *Let $\mathcal{S}$ be a schedule as in* (2.1) *for a BPMSTP instance $\mathcal{I}$. For a given EPS $\mathcal{E}$ on machine $h \in \mathcal{H}$ in $\mathcal{S}$, an* **EPS subschedule** *$\mathcal{S}_{\mathcal{E},h} \subseteq \mathcal{S}$ is a single-machine schedule $\{(j, h, \mathcal{T}_j), \forall\, j \in \mathcal{J} : \mathcal{T}_j \cap \mathcal{E} \neq \emptyset\}$.*

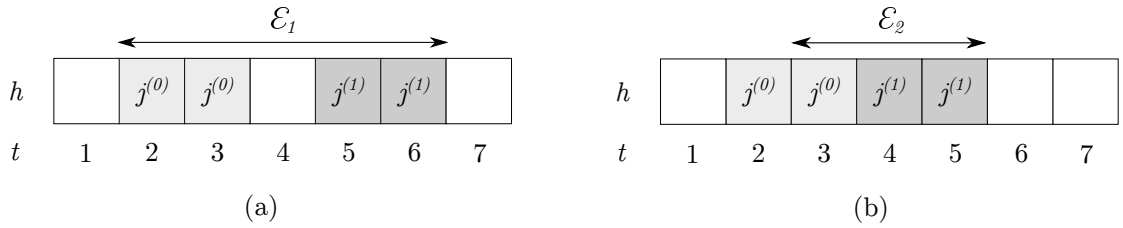

Figure 9: An example of two subsets of slots $\mathcal{E}_1 = \{2, 3, 4, 5, 6\}$ and $\mathcal{E}_2 = \{3, 4, 5\}$, of which only $\mathcal{E}_1$ is an EPS. In Figure 9(a), $\mathcal{E}_1$ is an EPS since $j^{(0)}$ and $j^{(1)}$ are entirely scheduled in $\mathcal{E}_1$. On the contrary, in Figure 9(b), $\mathcal{E}_2$ is not an EPS since $j^{(0)}$ is not entirely scheduled in $\mathcal{E}_2$.

For simplicity, we drop the second index from $\mathcal{S}_{\mathcal{E},h}$ when clear from the context. For a given EPS $\mathcal{E}$, there always exists a (possibly empty) subschedule $\mathcal{S}_{\mathcal{E}}$. In Figure 9(a), $\mathcal{S}_{\mathcal{E}_1}$ is the subschedule $\{(j^{(0)}, h, \{2, 3\}), (j^{(1)}, h, \{5, 6\})\}$. As a remark, observe that for a given free or assigned location $l = (h, \mathcal{A})$ of a job $j \in \mathcal{J}$, where $h \in \mathcal{H}$, $\mathcal{A} \subseteq \mathcal{T}$ is a subset of adjacent slots, and $p_j = |\mathcal{A}|$, $\mathcal{A}$ is an EPS by definition. For instance, in Figure 9(b), $(h, \{4, 5\})$ is the assigned location of $j^{(1)}$, and $(h, \{6, 7\})$ is a free location for a job with processing time equal to $p_{j^{(1)}} = 2$. As a matter of fact, the two sets $\{4, 5\}$ and $\{6, 7\}$ are EPS's.

For a given schedule $\mathcal{S}$, let $\mathcal{J}(\mathcal{S}_{\mathcal{E}}) \subseteq \mathcal{J}$ be the set of the jobs scheduled in $\mathcal{S}_{\mathcal{E}}$.

Definition 10. (EPS swap) *Let $\mathcal{S}$ be a feasible schedule for an instance $\mathcal{I}$ of the BPM-STP. Moreover, let $\mathcal{E} \subseteq \mathcal{T}$ and $\mathcal{E}' \subseteq \mathcal{T}$ be two EPS's on $h \in \mathcal{H}$ and $h' \in \mathcal{H}$, respectively, such that $|\mathcal{E}| = |\mathcal{E}'|$, and $\mathcal{E} \cap \mathcal{E}' = \emptyset$ if $h = h'$. Then, an* **EPS swap** *of $\mathcal{E}$ and $\mathcal{E}'$ on h in $\mathcal{S}$ is an algorithm that schedules each $j \in \mathcal{J}(\mathcal{S}_{\mathcal{E}})$ in $\mathcal{E}'$ on h', and each $j' \in \mathcal{J}(\mathcal{S}_{\mathcal{E}'})$ in $\mathcal{E}$ on h, by generating a new schedule $\mathcal{S}'$ without changing the relative assignments of the jobs in $\mathcal{E}$ and $\mathcal{E}'$, i.e., such that*

$$C_j(\mathcal{S}') = C_j(\mathcal{S}) - \max_{t \in \mathcal{E}} t + \max_{t \in \mathcal{E}'} t, \quad j \in \mathcal{J}(\mathcal{S}_{\mathcal{E}}),$$
$$C_{j'}(\mathcal{S}') = C_{j'}(\mathcal{S}) - \max_{t \in \mathcal{E}'} t + \max_{t \in \mathcal{E}} t, \quad j' \in \mathcal{J}(\mathcal{S}_{\mathcal{E}'}).$$

Figure 10 shows an example of an EPS swap of the two EPS's $\mathcal{E}_1$ and $\mathcal{E}_2$. In Figure 10(a), the EPS $\mathcal{E}_1$ has an associated subschedule $\mathcal{S}_{\mathcal{E}_1}$ that involves job $j^{(0)}$ assigned to the slots in $\mathcal{E}_1$ on machine h, while $\mathcal{E}_2$ has an associated subschedule $\mathcal{S}_{\mathcal{E}_2}$ that involves jobs $j^{(1)}$ and $j^{(2)}$ assigned to a subset of the slots in $\mathcal{E}_2$ on machine h'. Figure 10(b) shows the result of an EPS swap of the two EPS's. Observe that an EPS swap does not affect the feasibility of the involved subschedules, as it shifts the start times of the jobs in each subschedule by the same integer.

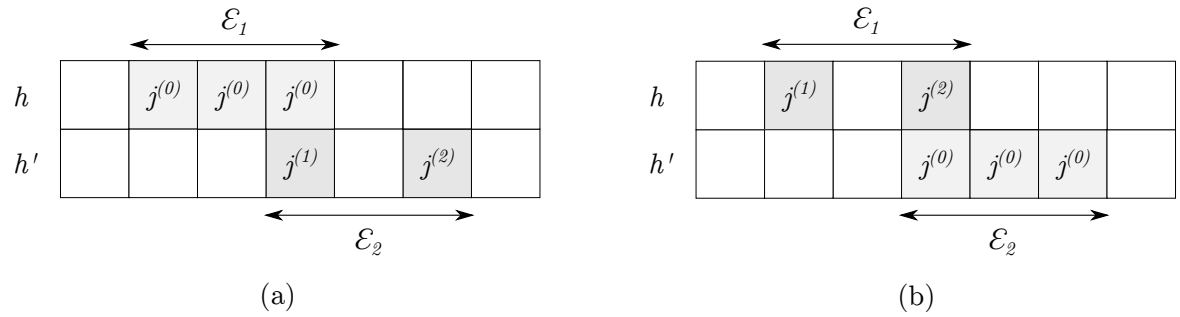

Figure 10: An example of an EPS swap of the EPS-J $\mathcal{E}_1$ and the EPS-I $\mathcal{E}_2$ on the machines h and h', respectively.

Definition 11. (EPS rearrangement) *Let $\mathcal{S}$ be a feasible schedule for a BPMSTP instance $\mathcal{I}$, and $\mathcal{E} \subseteq \mathcal{T}$ be an EPS on $h \in \mathcal{H}$ in $\mathcal{S}$. An* **EPS rearrangement** *of $\mathcal{E}$ on h in $\mathcal{S}$ is a procedure that reassigns each $j \in \mathcal{J}(\mathcal{S}_\mathcal{E})$ to locations of the form $(h, \mathcal{A})$, where $\mathcal{A} \subseteq \mathcal{E}$ is a subset of adjacent slots.*

Figure 11 shows an example of an EPS rearrangement applied to the EPS $\mathcal{E}_1$ in the schedule at the left of the arrow, which results in the EPS $\mathcal{E}_2$ in the schedule at the right.

Let us now discuss the idea at the core of ES. Towards this end, let us consider a schedule $\mathcal{S}$ for a BPMSTP instance $\mathcal{I}$. First, let us denote an EPS that contains no idle slots, and with a related subschedule with only a single job, as an *EPS-J*. Let us also refer to an EPS that contains at least an idle slot as an *EPS-I*. Then, let $\mathcal{E}$ and $\mathcal{E}'$ be any two EPS's in $\mathcal{S}$ on machines h and h', respectively, such that if $h = h'$, then $\mathcal{E} \cap \mathcal{E}' = \emptyset$. Hereinafter, we refer to an EPS swap of $\mathcal{E}$ and $\mathcal{E}'$, followed by a rearrangement of $\mathcal{E}$ and $\mathcal{E}'$, as an *EPS move* involving $\mathcal{E}$ and $\mathcal{E}'$. Suppose that such an EPS move results in a schedule $\mathcal{S}'$. Generally, $E(\mathcal{S}) \gtrless E(\mathcal{S}')$. ES searches for each move in the subset of EPS moves involving an EPS-J and an EPS-I, and that results in a schedule $\mathcal{S}'$ with TEC better than $\mathcal{S}$, i.e., $E(\mathcal{S}') < E(\mathcal{S})$. Each EPS move in such subset involves two EPS's

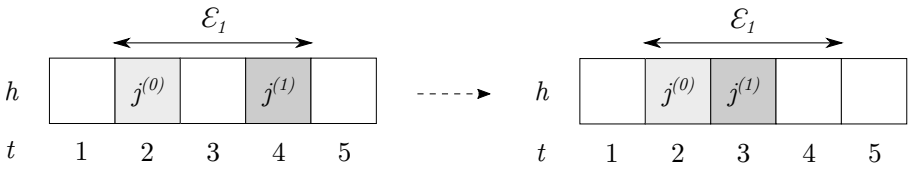

Figure 11: An example of an EPS rearrangement of the EPS-I $\mathcal{E}_1$.

with cardinality no greater than the maximum processing time of the jobs in $\mathcal{J}$, i.e.,

$$p_{\max} = \max_{p \in \mathcal{P}_{\mathcal{J}}} p,$$

by construction. Starting from a feasible schedule $\mathcal{S}$, ES explores such neighborhood of EPS moves, and it modifies $\mathcal{S}$ as soon as it finds an improving EPS move. Observe that such an EPS move cannot worsen that makespan by definition. Towards the description of the pseudo-code of ES, let us first define the cumulative energy price

$$\mu_{t,h} \coloneqq u_h \sum_{i=1}^{t} c_i, \quad h \in \mathcal{H}, t = 1, \ldots, K,$$

for a given instance $\mathcal{I}$, and $\mu_{0,h} \coloneqq 0, h \in \mathcal{H}$. Hence, the energy consumption cost of any location

$$l = (h, \{t, t+1, \ldots, t+p-1\}), \quad h \in \mathcal{H}, p \in \mathcal{P}, t = 1, 2, \ldots, K-p+1,$$

can be expressed as $\mu_{t+p-1,h} - \mu_{t-1,h}$. As a computational remark, the use of cumulative energy prices allows us to compute any location cost in $O(1)$, at the expense of precomputing $\mu_{t,h}, t = 1, \ldots, K, h \in \mathcal{H}$ in $O(MK)$.

Secondly, for a given feasible schedule $\mathcal{S}$, let us define $\mathcal{L}_p^{\mathrm{J}}(\mathcal{S})$ and $\mathcal{L}_p^{\mathrm{I}}(\mathcal{S})$ as the sets of subschedules of EPS-J's and EPS-I's in $\mathcal{S}$ of cardinality $p \in \mathcal{P}_{\mathcal{J}}$, respectively. Let us also introduce two sets of functions that associate a time slot t on a machine h with a subschedule $\mathcal{S}_{\mathcal{E}}$ associated with the EPS $\mathcal{E}$ on the same machine, and such that the smallest slot of $\mathcal{E}$ is t. Such functions are very useful from a computational standpoint, as they allow, if implemented as (direct addressing) tables or hash-maps, to verify the existence of an EPS on a machine and retrieve it in $O(1)$. With intent of recalling the data structures that enables this computational benefit, we hereinafter refer to such functions as "maps". Formally, let us denote as

$$\mathcal{M}_p^{\mathrm{J}}(\mathcal{S}) : \mathcal{H} \times \mathcal{T} \to \mathcal{L}_p^{\mathrm{J}}(\mathcal{S}) \cup \emptyset, \quad p \in \mathcal{P}_{\mathcal{J}},$$

the maps that, for each EPS-J $\{t, t+1, \ldots, t+p-1\} = \mathcal{E} : S_{\mathcal{E}} \in \mathcal{L}_p^{\mathrm{J}}(\mathcal{S})$, on a machine $h \in \mathcal{H}$, with $t \in \mathcal{T}$ and $p \in \mathcal{P}_{\mathcal{J}}$, associates (h, t) with $\mathcal{E}$. If $\mathcal{E}$ is not an EPS-J on h in schedule $\mathcal{S}$, then $\mathcal{M}_p^{\mathrm{J}}(\mathcal{S})$ associates (h, t) with the special value $\emptyset$. We denote the analogous maps for the EPS-I's as $\mathcal{M}_p^{\mathrm{I}}(\mathcal{S}) : \mathcal{H} \times \mathcal{T} \to \mathcal{L}_p^{\mathrm{I}}(\mathcal{S}) \cup \emptyset$. Finally, with a slight abuse of notation,

we refer to the two collections of sets $\{\mathcal{L}^{\mathrm{J}}_p(\mathcal{S}), p \in \mathcal{P}_{\mathcal{J}}\}$ and $\{\mathcal{L}^{\mathrm{I}}_p(\mathcal{S}), p \in \mathcal{P}_{\mathcal{J}}\}$ as $\mathcal{L}^{\mathrm{J}}(\mathcal{S})$ and $\mathcal{L}^{\mathrm{I}}(\mathcal{S})$, respectively. Similarly, we denote $\{\mathcal{M}^{\mathrm{J}}_p(\mathcal{S}), p \in \mathcal{P}_{\mathcal{J}}\}$ and $\{\mathcal{M}^{\mathrm{I}}_p(\mathcal{S}), p \in \mathcal{P}_{\mathcal{J}}\}$ as $\mathcal{M}^{\mathrm{J}}(\mathcal{S})$ and $\mathcal{M}^{\mathrm{I}}(\mathcal{S})$, respectively. From now on, the dependence on $\mathcal{S}$ is omitted whenever clear from the context.

Algorithm 4 reports the pseudo-code of ES. Algorithm 4 takes a feasible schedule $\mathcal{S}$ for a BPMSTP instance $\mathcal{D}(\hat{K})$ as in (3.1) as input, and it returns a feasible schedule $\mathcal{S}'$ obtained by performing all the improving EPS moves for $\mathcal{S}$. Algorithm 4 accomplishes this task by using the subroutines *FindEPS*, *EvaluateEPSMove*, and *UpdateEPS*, which are not reported in pseudo-code, and they are instead only described for the sake of compactness. In this way, the description omits the implementation details that are neither relevant to the analysis of the computational complexity, nor essential to the completeness of the presentation. First, we present an overview of such subroutines. FindEPS identifies all the EPS-J's and the EPS-I's in a feasible schedule $\mathcal{S}$. Algorithm 4 uses FindEPS to compute the maps $\mathcal{M}^{\mathrm{J}}$ and $\mathcal{M}^{\mathrm{I}}$, so as to iterate over distinct EPS moves in the main loop of the algorithm. EvaluateEPSMove efficiently evaluates if an EPS move entails an improvement in the TEC by disregarding some of the moves that cannot lead to improvement by exploiting a bounding condition. If the considered EPS move improves the TEC, it is performed. However, the maps $\mathcal{M}^{\mathrm{J}}$ and $\mathcal{M}^{\mathrm{I}}$ may become inconsistent after such an EPS move. In this case, UpdateEPS updates $\mathcal{M}^{\mathrm{J}}$ and $\mathcal{M}^{\mathrm{I}}$ by calling FindEPS so as to identify the new EPS's in the schedule. Let us now delve into a formal explanation of each subroutine.

FindEPS takes a feasible schedule $\mathcal{S}$ for $\mathcal{D}(\hat{K})$ as in (3.1), the subsets $\hat{\mathcal{H}} \subseteq \mathcal{H}$ and $\hat{\mathcal{T}} \subseteq \mathcal{T}$, and $\mathcal{P}_{\mathcal{J}}$ as an input, and returns the two sets of subschedules $\hat{\mathcal{L}}^{\mathrm{J}}(\mathcal{S})$ and $\hat{\mathcal{L}}^{\mathrm{I}}(\mathcal{S})$ of the EPS-J's and the EPS-I's in $\mathcal{S}$, respectively.

The identification of the EPS's in a schedule is conceptually similar to the task of finding free locations performed in line 6 of Algorithm 3. Formally, for any $p \in \mathcal{P}_{\mathcal{J}}$ and $h \in \hat{\mathcal{H}}$, two variables s and $e \geq s$ are first set to their initial values $\min_{t \in \hat{\mathcal{T}}} t$ and $\min_{t \in \hat{\mathcal{T}}} t + p - 1$, respectively. If $e > \max_{t \in \hat{\mathcal{T}}} t$, there is no EPS in $\mathcal{T}$ on machine h in schedule $\mathcal{S}$. Otherwise, FindEPS starts a loop as follows. At each iteration, it considers the set of slots $\mathcal{E} = \{s, s+1, \ldots, e\}$, and it checks if it is an EPS. If that is the case, it adds $\mathcal{S}_{\mathcal{E}}$ to $\hat{\mathcal{L}}^{\mathrm{J}}(\mathcal{S})$ if $\mathcal{E}$ is an EPS-J, or to $\hat{\mathcal{L}}^{\mathrm{I}}(\mathcal{S})$ if $\mathcal{E}$ is an EPS-I. At the end of the iteration, s is incremented by 1, or by the processing time of the job starting at s, if there is one. Then, e is incremented by the same quantity, so as to preserve the necessary condition

$p = e - s + 1$ for $\mathcal{E}$ to be an EPS. If e reaches the value $\max_{t \in \hat{T}} t + 1$, i.e., e cannot be the largest slot in an EPS, and the loop ends. Otherwise, FindEPS proceeds with the next iteration.

In order for $\mathcal{E}$ to be an EPS-I, or an EPS-J, of cardinality p, $e - s + 1$ has to be equal to p. Furthermore, s has to be free, or be the start time of a job, and e must be free, or be the end time of a job. In particular, $\mathcal{E}$ is an EPS-J if the slots in $\mathcal{E}$ are assigned to a single job. Instead, if at least one of the slots in $\mathcal{E}$ if free, then $\mathcal{E}$ is an EPS-I. Otherwise, $\mathcal{E}$ is still an EPS, but it is neither an EPS-J, nor an EPS-I.
Checking these conditions requires multiple queries to the LLRB's used to represent schedules, which yield a worst-case $O(\log_2 \hat{K})$ complexity for each of such queries. Each of them is carried out at most twice for each $t \in \hat{T}$, when s or e reach t. As a side, but significant, implementation note, for a given EPS $\mathcal{E} = \{t, t+1, \ldots, t+p-1\} \subseteq \mathcal{T}$ on a machine $h \in \mathcal{H}$, the actual implementation of $\mathcal{M}_p^{\mathrm{J}}$ and $\mathcal{M}_p^{\mathrm{I}}$ associates the key $\hat{K}h + t_i$ to $\mathcal{S}_{\mathcal{E}}$. This allows us to uniquely address each EPS by using a single key.

FindEPS also stores two additional information for further use: the number of the assigned slots in $\mathcal{E}$ and the cumulative energy cost of the list of slots contained in $\mathcal{E}$, sorted by energy cost in non-decreasing order. This information is necessary for EvaluateEPSMove in the evaluation of its bounding condition.

Finally, the computational complexity of FindEPS is

$$O(|\mathcal{P}_{\mathcal{J}}||\hat{\mathcal{H}}||\hat{\mathcal{T}}| \max\{\log_2 \hat{K},\, p_{\max} \log_2 p_{\max}\}) \tag{3.5}$$

The term $\max\{\log_2 \hat{K},\, p_{\max} \log_2 p_{\max}\}$ is due to the two distinct operations carried out for each $p \in \mathcal{P}_{\mathcal{J}}$, $h \in \hat{\mathcal{H}}$, and $t \in \{\min_{k \in \hat{\mathcal{T}}} k, \min_{k \in \hat{\mathcal{T}}} k + 1, \ldots, \max_{k \in \hat{\mathcal{T}}} k - p + 1\} \subseteq \hat{\mathcal{T}}$: querying the LLRB to check whether t is free or not on h at most twice, and sorting the slot costs of $\{t, t+1, \ldots, t+p-1\}$, if it is an EPS.

EvaluateEPSMove takes a schedule $\mathcal{S}$ for $\mathcal{D}(\hat{K})$ as in (3.1), and an EPS-I $\mathcal{E}^{(0)}$ and an EPS-J $\mathcal{E}^{(1)}$ of cardinality $p \in \mathcal{P}_{\mathcal{J}}$ on machines $h^{(0)}$ and $h^{(1)}$, respectively, as input. As an output, it returns the schedule $\mathcal{S}'$ that results from the EPS move involving $\mathcal{E}^{(0)}$ and $\mathcal{E}^{(1)}$ on machines $h^{(0)}$ and $h^{(1)}$, respectively, as well as the difference $E(\mathcal{S}') - E(\mathcal{S})$. The purpose of EvaluateEPSMove is to evaluate the improvement in TEC entailed by the EPS move involving $\mathcal{E}^{(0)}$ and $\mathcal{E}^{(1)}$.

First, EvaluateEPSMove checks a necessary condition for TEC improvement, so as to

possibly avoid to unnecessarily evaluate the EPS move. EvaluateEPSMove achieves so by exploiting the additional information stored with each EPS by FindEPS. Towards the end of describing such condition, let us consider an EPS

$$\mathcal{E} = \{i, i+1, \ldots, i+p-1\}, \quad p \in \mathcal{P}_{\mathcal{J}}, i \in \{1, 2, \ldots, K-p+1\},$$

on machine $h \in \mathcal{H}$ in $\mathcal{S}$. Then, let us refer as $Q_{\mathcal{E}}$ to the list containing the time slots in $\mathcal{E}$, sorted in non-decreasing order of their cost, i.e.,

$$Q_{\mathcal{E}} = (t^{(i)}, t^{(i+1)}, \ldots, t^{(i+p-1)}), \quad c_{t^{(i+n-1)}} < c_{t^{(i+n)}}, n = 1, 2, \ldots, p-1,$$

and such that $Q_{\mathcal{E}}$ is a permutation of $(i, i+1, \ldots, i+p-1)$.

For n such that $0 \leq n \leq p-1$, let us denote as

$$\eta^{\text{lb}}_{h,n}(Q_{\mathcal{E}}) = u_h \sum_{r=0}^{n-1} c_{t^{(i+r)}}, \qquad \eta^{\text{ub}}_{h}(\mathcal{E}) = u_h \sum_{r=0}^{p-1} c_{i+r},$$

the cumulative cost of the first n slots in the list $Q_{\mathcal{E}}$, and the sum of the costs of the slots in $\mathcal{E}$, respectively. Then, the inequality

$$\eta^{\text{lb}}_{h,n}(E) \leq E(\mathcal{S}_{\mathcal{E}}) \leq \eta^{\text{ub}}_{h}(\mathcal{E})$$

holds for $0 \leq n \leq p-1$. Finally, let $\alpha(\mathcal{S}_{\mathcal{E}}) \coloneqq \sum_{j \in \mathcal{J}(\mathcal{S}_{\mathcal{E}})} p_j$ be the number of slots assigned in $\mathcal{S}_{\mathcal{E}}$.

Formally, the EPS move involving the EPS-I $\mathcal{E}^{(0)}$ and the EPS-J $\mathcal{E}^{(1)}$ results in two subschedules, say, $\mathcal{S}'_{\mathcal{E}^{(0)}}$ and $\mathcal{S}'_{\mathcal{E}^{(1)}}$. Then, $E(\mathcal{S}'_{\mathcal{E}^{(0)}}) + E(\mathcal{S}'_{\mathcal{E}^{(1)}})$ is lower bounded by $\eta^{\text{ub}}_{h}(\mathcal{E}^{(0)}) + \eta^{\text{lb}}_{h,\alpha(\mathcal{S}_{\mathcal{E}^{(0)}})}(Q_{\mathcal{E}^{(1)}})$, since, after the EPS move, the job initially in $\mathcal{E}^{(1)}$ is then scheduled entirely in $\mathcal{E}^{(0)}$ with cost $\eta^{\text{ub}}_{h}(Q_{\mathcal{E}^{(0)}})$, and the cost of the jobs initially in $\mathcal{E}^{(0)}$, and then scheduled in $\mathcal{E}^{(1)}$, cannot exceed the sum of the $\alpha(\mathcal{S}_{\mathcal{E}^0})$ smallest costs of the slots in $\mathcal{E}^{(1)}$, i.e., $\eta^{\text{lb}}_{h,\alpha(\mathcal{S}_{\mathcal{E}^{(0)}})}(Q_{\mathcal{E}^{(1)}})$. If

$$E(\mathcal{S}_{\mathcal{E}^{(0)}}) + E(\mathcal{S}_{\mathcal{E}^{(1)}}) \leq \eta^{\text{ub}}_{h}(\mathcal{E}^{(0)}) + \eta^{\text{lb}}_{h,\alpha(\mathcal{S}_{\mathcal{E}^{(0)}})}(Q_{\mathcal{E}^{(1)}}) \tag{3.6}$$

then the EPS move is disregarded, as it cannot entail an improvement in TEC, and EvaluateEPSMove returns $\mathcal{S}$ and 0. Otherwise, ES carries out the EPS move by first applying the EPS swap, and then performing the EPS rearrangement of $\mathcal{E}^{(0)}$ with SGH.

As a consequence, ES generates a new schedule $\mathcal{S}'$. Afterwards, EvaluateEPSMove sets an inner variable δ as

$$\delta \leftarrow E(\mathcal{S}'_{\mathcal{E}^{(0)}}) + E(\mathcal{S}'_{\mathcal{E}^{(1)}}) - E(\mathcal{S}_{\mathcal{E}^{(0)}}) - E(\mathcal{S}_{\mathcal{E}^{(1)}}),$$

that is negative if the move is improving, and non-negative otherwise. Finally, EvaluateEPSMove returns $\mathcal{S}'$ and δ.
The computational complexity of EvaluateEPSMove is dominated by the one of SGH, given in (3.3), which is called once for each of the two EPS's involved. Therefore, the complexity of EvaluateEPSMove is

$$O(p_{\max}(2p_{\max} + 2p_{\max}) \log_2 \Omega + 4p_{\max} + p_{\max} \log_2 p_{\max}),$$

which is equivalent to

$$O(p_{\max}^2 \log_2 \Omega). \tag{3.7}$$

UpdateEPS takes a schedule $\mathcal{S}$ for $\mathcal{D}(\hat{K})$ as in (3.1), the two sets $\mathcal{L}^{\mathrm{J}}$ and $\mathcal{L}^{\mathrm{I}}$, the two sets of maps $\mathcal{M}^{\mathrm{J}}$ and $\mathcal{M}^{\mathrm{I}}$, a machine $h \in \mathcal{H}$, a subset $\hat{\mathcal{T}} \subseteq \mathcal{T}$, and $\mathcal{P}_{\mathcal{J}}$ as input, and returns the updated sets $\mathcal{L}^{\mathrm{J}}$ and $\mathcal{L}^{\mathrm{I}}$ of EPS's in $\hat{\mathcal{T}}$ on h, as well as the related maps $\mathcal{M}^{\mathrm{J}}$ and $\mathcal{M}^{\mathrm{I}}$. The purpose of UpdateEPS is to update $\mathcal{L}^{\mathrm{J}}$, $\mathcal{L}^{\mathrm{I}}$, $\mathcal{M}^{\mathrm{J}}$, and $\mathcal{M}^{\mathrm{I}}$, that may be inconsistent with $\mathcal{S}$ after an improving EPS move performed by EvaluateEPSMove.

In order to achieve this, UpdateEPS first iterates over each $p \in \mathcal{P}_{\mathcal{J}}$, and for each t such that

$$\max\{0, \min_{t' \in \hat{\mathcal{T}}} t' - p + 1\} \leq t \leq \min\{\hat{K} - p + 1, \max_{t' \in \hat{\mathcal{T}}} t'\},$$

it removes the entry, if present, associated with (h, t) from both $\mathcal{L}_p^{\mathrm{J}}$ and $\mathcal{L}_p^{\mathrm{I}}$. The complexity of such removal operations is $O(|\mathcal{P}_{\mathcal{J}}||\hat{\mathcal{T}}|) = O(|\mathcal{P}_{\mathcal{J}}|p_{\max})$, as $|\hat{\mathcal{T}}| \leq 3p_{\max} - 2$. In order to show this, let us call the entry associated with (h, t) as $\mathcal{E}$. Then, observe that most the $p_{\max} - 1$ slots before $\mathcal{E}$, the $p_{\max}$ slots in $\mathcal{E}$, and the $p_{\max} - 1$ slots after $\mathcal{E}$ are affected by the EPS move prior to the call of UpdateEPS.

Afterwards, UpdateEPS calls FindEPS with parameters $\mathcal{S}$, $\{h\}$, $\hat{\mathcal{T}}$, and $\mathcal{P}_{\mathcal{J}}$. Let us

denote its return values as

$$\hat{\mathcal{L}}^{\mathrm{I}} = \{\hat{\mathcal{L}}^{\mathrm{I}}_p, p \in \mathcal{P}_{\mathcal{J}}\}, \quad \text{and } \hat{\mathcal{L}}^{\mathrm{J}} = \{\hat{\mathcal{L}}^{\mathrm{J}}_p, p \in \mathcal{P}_{\mathcal{J}}\}.$$

UpdateEPS updates $\mathcal{L}^{\mathrm{I}}$ and $\mathcal{L}^{\mathrm{J}}$ with the newfound EPS's by performing the assignments

$$\mathcal{L}^{\mathrm{I}} \leftarrow \mathcal{L}^{\mathrm{I}}_p \cup \hat{\mathcal{L}}^{\mathrm{I}}_p, \quad \text{and } \mathcal{L}^{\mathrm{J}} \leftarrow \mathcal{L}^{\mathrm{J}}_p \cup \hat{\mathcal{L}}^{\mathrm{J}}_p$$

for each $p \in \mathcal{P}_{\mathcal{J}}$ in $O(|\mathcal{P}_{\mathcal{J}}|p_{\max})$ time. The maps $\mathcal{M}^{\mathrm{J}}$ and $\mathcal{M}^{\mathrm{I}}$ are then updated accordingly, again in $O(|\mathcal{P}_{\mathcal{J}}|p_{\max})$ time. Therefore, the complexity of UpdateEPS is

$$O(|\mathcal{P}_{\mathcal{J}}|p_{\max} \max\{\log_2 \Omega,\, p_{\max} \log_2 p_{\max}\} + |\mathcal{P}_{\mathcal{J}}|p_{\max}),$$

that can be compacted as

$$O(|\mathcal{P}_{\mathcal{J}}|p_{\max} \max\{\log_2 \Omega,\, p_{\max} \log_2 p_{\max})\}. \tag{3.8}$$

Let us finally describe the pseudo-code of Algorithm 4. Line 1 uses FindEPS in order to identify the EPS's in $\mathcal{S}$. Then, lines 2–4, for each $p \in \mathcal{P}_{\mathcal{J}}$, compute the maps $\mathcal{M}^{\mathrm{J}}_p$ and $\mathcal{M}^{\mathrm{I}}_p$. Line 5 instead computes $\mathcal{M}^{\mathrm{J}}$ and $\mathcal{M}^{\mathrm{I}}$, and line 6 builds the list $\hat{P}$ from $\mathcal{P}_{\mathcal{J}}$ as in Algorithm 3. Afterwards, lines 8–22 are iterated as long as they result in at least an improving EPS move. Specifically, for each possible move involving an EPS-J and an EPS-I in $\mathcal{S}$ (lines 9–11), ES evaluates it with EvaluateEPSMove at line 12. If such move is improving, i.e., $\delta < 0$, line 14 assigns the resulting schedule $\mathcal{S}'$ to $\mathcal{S}$. Then, lines 15 and 16 update the lists of EPS's and the maps with UpdateEPS. Line 17 sets the variable Improvement to true. As a result, at the end of the current iteration of lines 8–22, they are executed again. Line 18 interrupts the innermost loop according to the first improvement strategy, so as to search for further improving EPS moves that involve other EPS-J's. If the last iteration of lines 8–22 is unsuccessful in improving the TEC of $\mathcal{S}$, the loop ends. Finally, line 25 returns a feasible schedule $\mathcal{S}'$ that improves, or does not worsen, the TEC and the makespan of $\mathcal{S}$.

Let us discuss the computational complexity of Algorithm 4. Since there is a total of $M\hat{K}$ keys, lines 2–4 take $O(|\mathcal{P}_{\mathcal{J}}|M\hat{K})$. Line 5 takes $O(\mathcal{P}_{\mathcal{J}})$, while line 6 takes $O(|\mathcal{P}_{\mathcal{J}}| \log_2 |\mathcal{P}_{\mathcal{J}}|)$. Let us consider lines 7–23. First, observe that an EPS-J of cardinality $p \in \mathcal{P}_{\mathcal{J}}$ can be involved in an EPS move with up to $M(\hat{K} - p + 1)$ EPS-I's. Therefore,

Algorithm 4 Exchange-Search (ES)

Input A feasible schedule $\mathcal{S}$ for a BPMSTP instance $\mathcal{D}(\hat{K})$ as in (3.1).

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: Let $\mathcal{L}^{\mathrm{J}}, \mathcal{L}^{\mathrm{I}} \leftarrow \mathrm{FindEPS}(\mathcal{S}, \mathcal{H}, \mathcal{T}, \mathcal{P}_{\mathcal{J}})$
2: **for** each $p \in \mathcal{P}_{\mathcal{J}}$ **do**
3: Build the maps $\mathcal{M}_p^{\mathrm{J}} : \mathcal{H} \times \mathcal{T} \to \mathcal{L}_p^{\mathrm{J}}$ and $\mathcal{M}_p^{\mathrm{I}} : \mathcal{H} \times \mathcal{T} \to \mathcal{L}_p^{\mathrm{I}}$
4: **end for**
5: Let $\mathcal{M}^{\mathrm{J}} \leftarrow \{\mathcal{M}_p^{\mathrm{J}}, p \in \mathcal{P}_{\mathcal{J}}\}$, and $\mathcal{M}^{\mathrm{I}} \leftarrow \{\mathcal{M}_p^{\mathrm{I}}, p \in \mathcal{P}_{\mathcal{J}}\}$
6: Let $\hat{P}$ be the list of the elements $\mathcal{P}_{\mathcal{J}}$ sorted in non-increasing order
7: **repeat**
8: Let Improvement $\leftarrow$ false
9: **for** $p \in \hat{P}$ **do**
10: **for** $\mathcal{E}^{\mathrm{J}} : ((h^{\mathrm{J}}, \min_{t \in \mathcal{E}^{\mathrm{J}}} t), \mathcal{S}_{\mathcal{E}^{\mathrm{J}}}) \in \mathcal{M}_p^{\mathrm{J}}$ **do**
11: **for** $\mathcal{E}^{\mathrm{I}} : ((h^{\mathrm{I}}, \min_{t \in \mathcal{E}^{\mathrm{I}}} t), \mathcal{S}_{\mathcal{E}^{\mathrm{I}}}) \in \mathcal{M}_p^{\mathrm{I}}$ **do**
12: $\mathcal{S}', \delta \leftarrow \mathrm{EvaluateEPSMove}(\mathcal{S}, \mathcal{E}^{\mathrm{J}}, \mathcal{E}^{\mathrm{I}}, h^{\mathrm{J}}, h^{\mathrm{I}})$
13: **if** $\delta < 0$ **then**
14: Let $\mathcal{S} \leftarrow \mathcal{S}'$
15: $\mathcal{L}^{\mathrm{J}}, \mathcal{L}^{\mathrm{I}}, \mathcal{M}^{\mathrm{J}}, \mathcal{M}^{\mathrm{I}} \leftarrow \mathrm{UpdateEPS}(\mathcal{S}, \mathcal{L}^{\mathrm{J}}, \mathcal{L}^{\mathrm{I}}, \mathcal{M}^{\mathrm{J}}, \mathcal{M}^{\mathrm{I}}, h^{\mathrm{J}}, \hat{\mathcal{T}}^{\mathrm{J}}, \mathcal{P}_{\mathcal{J}})$
16: $\mathcal{L}^{\mathrm{J}}, \mathcal{L}^{\mathrm{I}}, \mathcal{M}^{\mathrm{J}}, \mathcal{M}^{\mathrm{I}} \leftarrow \mathrm{UpdateEPS}(\mathcal{S}, \mathcal{L}^{\mathrm{J}}, \mathcal{L}^{\mathrm{I}}, \mathcal{M}^{\mathrm{J}}, \mathcal{M}^{\mathrm{I}}, h^{\mathrm{I}}, \hat{\mathcal{T}}^{\mathrm{I}}, \mathcal{P}_{\mathcal{J}})$
17: Improvement $\leftarrow$ true
18: **break**
19: **end if**
20: **end for**
21: **end for**
22: **end for**
23: **until not** Improvement
24: $\mathcal{S}' \leftarrow \mathcal{S}$
25: **return** $\mathcal{S}'$

since there are N distinct EPS-J's, one for each job, the maximum number of EPS moves to be evaluated is $O(NM\hat{K})$, that is, lines 12–19 are executed $O(NM\hat{K})$ times. However, observe that, generally the higher N and $\sum_{p \in \mathcal{P}_{\mathcal{J}}} p$, the lower the number of idle slots, hence the smaller the set of EPS-I's. Moreover, for each EPS-J $\mathcal{E}$, ES interrupts the innermost loop (lines 11–20) as soon as an improving EPS move for an EPS-J $\mathcal{E}$, and then does not generally consider each EPS move involving $\mathcal{E}$. Hence, $O(NM\hat{K})$ is not the tightest bound for the number of EPS-I's, and the actual performances may be better than the computational complexity suggests.

The number of iterations of lines 7–23 depends on the structure of the solution $\mathcal{S}$, and

on the instance $\mathcal{D}(\hat{K})$ as well. In order to simplify the analysis without losing generality, we suppose that lines 7–23 can be executed at most $R > 0$ times. The value of R depends both on the instance of the problem, and on the schedule $\mathcal{S}$ given as input.
The determination of R is out of the scope of this thesis. However, R has a very small upper bound in practice. In fact, experimental observations related to the numerical results presented in Chapter 4 suggest that R is generally two orders of magnitude lower than $\hat{K}$ on the considered benchmark instances.
Finally, in light of these observations, the computational complexity of ES is

$$\begin{aligned} &O(|\mathcal{P}_{\mathcal{J}}||M\hat{K}\max\{\log_2\Omega, p_{\max}\log_2 p_{\max}\} + O(|\mathcal{P}_{\mathcal{J}}|M\hat{K}) + O(|\mathcal{P}_{\mathcal{J}}|\log_2|\mathcal{P}_{\mathcal{J}}|) + \\ &O(RNM\hat{K}(p_{\max}^2\log_2\Omega + |\mathcal{P}_{\mathcal{J}}|p_{\max}\max\{\log_2\Omega, p_{\max}\log_2 p_{\max}\})). \end{aligned} \tag{3.9}$$

The terms in the sum correspond to the complexity of line 1, as in (3.5), lines 2–4, lines 5–6, and to the main loop at lines 7–23, respectively. Specifically, as regards the complexity of lines 7–23, the first summand is due to line 12, and given by (3.7), while the second summand is due to lines 15 and 16, and given by (3.8). Expression (3.9) can be equivalently and more compactly expressed as

$$O(RNM\hat{K}p_{\max}^2(\log_2\Omega + p_{\max}\log_2 p_{\max}))$$

As a computational remark, let us observe that if the schedules were implemented as ordered lists instead of ordered sets, the computational complexity for scheduling each of the $O(p_{\max})$ jobs in an EPS would be $O(\Omega)$ instead of $O(\log_2\Omega)$. In fact, in this case, each job would require $O(\log_2\Omega)$ for binary search to identify the insertion point, and then take $O(\Omega)$ for shifting the elements after it.

An example may be useful to illustrate the update operations performed by ES.

Example 2. Let us consider an example of the application of ES on a schedule, say, $\mathcal{S}$, on a single machine h with $u_h = 1$, depicted in Figure 12(a). Figure 12(b) shows the

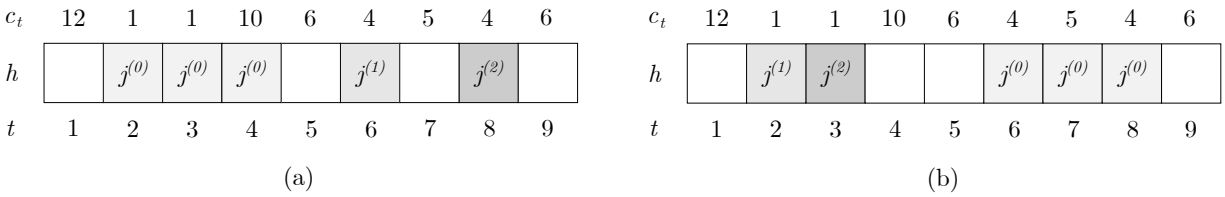

Figure 12: An example of an EPS move performed by ES. Such move involves the EPS-J {2, 3, 4} and the EPS-I {6, 7, 8} in the schedule in Figure 12(a). The resulting schedule is shown in Figure 12(b).

resulting schedule $\mathcal{S}'$. ES performs only one improving EPS move, that leads to a TEC improvement from $E(\mathcal{S}) = 20$ to $E(\mathcal{S}') = 15$.

Let us describe the distinct, possibly improving, EPS moves for $\mathcal{S}$. First, the EPS-J's in $\mathcal{S}$ are the ones in $\mathcal{L}_3^{\mathrm{J}}(\mathcal{S}) = \{\{2,3,4\}\}$ and $\mathcal{L}_1^{\mathrm{J}}(\mathcal{S}) = \{\{6\},\{8\}\}$. Similarly,

$$\mathcal{L}_1^{\mathrm{I}}(\mathcal{S}) = \{\{1\},\{5\},\{7\},\{9\}\}, \quad \mathcal{L}_2^{\mathrm{I}}(\mathcal{S}) = \{\{5,6\},\{6,7\},\{7,8\},\{8,9\}\},$$
$$\mathcal{L}_3^{\mathrm{I}}(\mathcal{S}) = \{\{5,6,7\},\{6,7,8\},\{7,8,9\}\}.$$

Let $\mathcal{E}^{(0)}$ be the EPS-J $\{2,3,4\}$ in $\mathcal{L}_3^{\mathrm{J}}(\mathcal{S})$, and let also

$$\mathcal{E}^{(1)} = \{5,6,7\}, \quad \mathcal{E}^{(2)} = \{6,7,8\}, \quad \mathcal{E}^{(3)} = \{7,8,9\}$$

be the EPS-I's in $\mathcal{L}_3^{\mathrm{I}}(\mathcal{S})$.

Let us consider the possible EPS moves involving $\mathcal{E}^{(0)}$. First,

$$E(\mathcal{S}_{\mathcal{E}^{(0)}}) = 12, \quad E(\mathcal{S}_{\mathcal{E}^{(1)}}) = 4, \quad E(\mathcal{S}_{\mathcal{E}^{(2)}}) = 8, \quad E(\mathcal{S}_{\mathcal{E}^{(3)}}) = 4.$$

Moreover,

$$\eta_h^{\mathrm{ub}}(\mathcal{E}^{(1)}) = 15, \quad \eta_h^{\mathrm{ub}}(\mathcal{E}^{(2)}) = 13, \quad \eta_h^{\mathrm{ub}}(\mathcal{E}^{(3)}) = 15,$$

and the cumulative costs of the slots in $\mathcal{E}^{(0)}$ in non-decreasing order of their costs are

$$\eta_{h,1}^{\mathrm{lb}}(Q_{\mathcal{E}^{(0)}}) = 1, \quad \eta_{h,2}^{\mathrm{lb}}(Q_{\mathcal{E}^{(0)}}) = 2, \quad \eta_{h,3}^{\mathrm{lb}}(Q_{\mathcal{E}^{(0)}}) = 12.$$

The necessary condition (3.6) for an improving move with $\mathcal{E}^{(0)}$ is only satisfied by $\mathcal{E}^{(2)}$ among the EPS's in $\mathcal{L}_3^{\mathrm{I}}(\mathcal{S})$, since

$$\eta_h^{\mathrm{ub}}(\mathcal{E}^{(2)}) + \eta_{h,2}^{\mathrm{lb}}(Q_{\mathcal{E}^{(0)}}) = 15 < E(\mathcal{S}_{\mathcal{E}^{(2)}}) + E(\mathcal{S}_{\mathcal{E}^{(0)}}) = 20,$$

but

$$\eta_h^{\mathrm{ub}}(\mathcal{E}^{(1)}) + \eta_{h,1}^{\mathrm{lb}}(Q_{\mathcal{E}^{(0)}}) = 16, \quad \eta_h^{\mathrm{ub}}(\mathcal{E}^{(3)}) + \eta_{h,1}^{\mathrm{lb}}(Q_{\mathcal{E}^{(0)}}) = 16$$

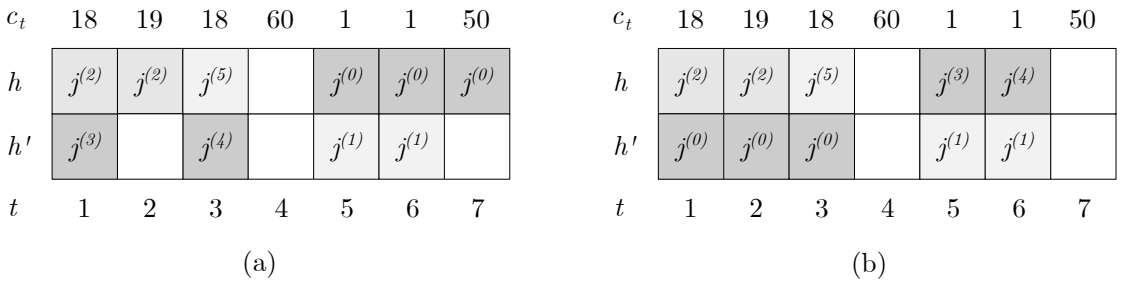

Figure 13: An example of the application of ES. There is only one improving EPS move, that involves the EPS-J $\{5, 6, 7\}$ on machine h and the EPS-I $\{1, 2, 3\}$ on h' in Figure 13(a). Figure 13(b) shows the resulting schedule.

are no greater than

$$E(\mathcal{S}_{\mathcal{E}^{(1)}}) + E(\mathcal{S}_{\mathcal{E}^{(0)}}) = 16, \quad E(\mathcal{S}_{\mathcal{E}^{(3)}}) + E(\mathcal{S}_{\mathcal{E}^{(0)}}) = 16,$$

respectively. The EPS move involving $\mathcal{E}^{(0)}$ and $\mathcal{E}^{(2)}$ is indeed the only one that entails an improvement in TEC.

Another example may be helpful in highlighting why ES can be effective in improving solutions.

Example 3. Figure 13(a) shows a feasible schedule, say, $\mathcal{S}$, possibly generated by SGH, for two machines h and h', with $u_h = u_{h'} = 1$, and six jobs $j^{(0)}$, $j^{(1)}$, $j^{(2)}$, $j^{(3)}$, $j^{(4)}$ and $j^{(5)}$, with $p_{j^{(0)}} = 3$, $p_{j^{(1)}} = p_{j^{(2)}} = 2$, and $p_{j^{(3)}} = p_{j^{(4)}} = p_{j^{(5)}} = 1$. The makespan and the TEC of $\mathcal{S}$ are $C^{\max}(\mathcal{S}) = 7$ and $E(\mathcal{S}) = 145$, respectively. Figure 13(b) shows the result of the application of ES to $\mathcal{S}$. ES performs only one improving EPS move, i.e., the one that involves the EPS-J $\mathcal{E}^{\mathrm{J}} = \{5, 6, 7\}$ on machine h, and the EPS-I $\mathcal{E}^{\mathrm{I}} = \{1, 2, 3\}$ on machine h'. After the EPS swap that is part of such move, ES performs the EPS rearrangement of $\mathcal{E}^{\mathrm{J}}$ by scheduling $j^{(3)}$, $j^{(4)}$ with SGH. The resulting schedule $\mathcal{S}'$ improves the TEC, as $E(\mathcal{S}') = 114 < E(\mathcal{S})$. Incidentally, $\mathcal{S}$ also improves the makespan, as $C^{\max}(\mathcal{S}') = 6 < C^{\max}(\mathcal{S})$.

3.3 Split-Greedy Scheduler

This section describes SGS, the first complete heuristic scheme for the BPMSTP presented in this thesis.

SGS exploits the ϵ-constraint paradigm [56] for multi-objective optimization, similarly to CH. First, let us observe that, for a given BPMSTP instance $\mathcal{I}$, each Pareto-optimal solution $\mathcal{S}^\star$ to $\mathcal{I}$ corresponds to a non-dominated point $(C^{\max}(\mathcal{S}^\star), E(\mathcal{S}^\star))$ in the optimal Pareto front. In particular, there exist at most $K - \underline{K}(\mathcal{I}) + 1$ different Pareto-optimal solutions, where

$$\underline{K}(\mathcal{I}) = \max \left\{ \lfloor \sum_{j \in \mathcal{J}} p_j / M \rfloor, \max_{j \in \mathcal{J}} \{p_j\} \right\}. \tag{3.10}$$

Indeed, since the processing times p_j, $j \in \mathcal{J}$, are integer numbers, $C^{\max}(\mathcal{S}^\star)$ ranges between the lower bound $\underline{K}(\mathcal{I})$ and the upper bound K. Observe that the lower bound (3.10) used by SGS is stronger than the bound $K^{min} = \sum_{j \in J} p_j / M$ in CH.

SGS computes a set of non-dominated, heuristic solutions by solving, for each $\hat{K}$ such that $\underline{K}(\mathcal{I}) \leq \hat{K} \leq K$, the BPMSTP instance $\mathcal{D}(\hat{K})$ as in (3.1) with SGH. The pseudo-code of SGS is reported in Algorithm 5. At line 1, Algorithm 5 initializes $\underline{K}(\mathcal{I})$ as in (3.10). Then, at line 2, it initializes the set of solutions $\mathcal{F}$ as an empty set. It also initializes the iteration variable $\hat{K}$, used as the upper bound for the makespan in the loop at lines 3–11, as K. At lines 3–11, Algorithm 5 computes the set of solutions $\mathcal{F}$. Algorithm 5 first evaluates the condition $\hat{K} \geq \underline{K}$ at line 3. If it does not hold, i.e., the current maximum makespan $\hat{K}$ is not greater or equal than the lower bound $\underline{K}$, then there is no feasible solution for $\hat{K}$. Afterwards, at line 5, it solves an instance $\mathcal{D}(\hat{K})$ as in (3.1) for the BPMSTP by means of SGH. If $\mathcal{S}$ is infeasible, then the loop ends at line 7. Observe that, as opposed to CH, Algorithm 5 verifies the feasibility of the computed solution before proceeding. If $\mathcal{S}$ is feasible, then $\mathcal{S}$ is added to $\mathcal{F}$ at line 9, and $\hat{K}$ is updated at line 10 for the next iteration. At line 12, Algorithm 5 finally returns the set of non-dominated heuristic solutions in $\mathcal{F}$.

The computational complexity of Algorithm 5 is dominated by SGH at line 5, which occurs $O(K)$ times during the execution of the algorithm. Hence, the complexity of Algorithm 5 is

$$O(K(\Omega M |\mathcal{P}_\mathcal{J}| \log_2 \Omega + \Omega N \log_2 \Omega + NM + |\mathcal{P}_\mathcal{J}| \log_2 \mathcal{P}_\mathcal{J})).$$

As a final remark, observe that the lower bound $\underline{K}(\mathcal{I})$ given by (3.10) is not tight for all the instances of the BPMSTP, and generally for the problem $Pm||C^{\max}$ as well.

Example 4. Let us consider a BPMSTP instance with a set of $N = 4$ jobs with processing

Algorithm 5 Split-Greedy Scheduler

Input An instance $\mathcal{I}$ of the BPMSTP.
Output A set of non-dominated heuristic solutions for $\mathcal{I}$.

1: $\underline{K} \leftarrow max\{\lfloor \sum_{j \in J} \ p_j/M \rfloor, max_{j \in J}\{p_j\}\}$
2: Let $\mathcal{F} \leftarrow \emptyset$, and $\hat{K} \leftarrow K$
3: **while** $\hat{K} \geq \underline{K}$ **do**
4: Let $\mathcal{D}(\hat{K})$ be an instance as in (3.1)
5: $\mathcal{S} \leftarrow SGH(\mathcal{D}(\hat{K}))$
6: **if** $\mathcal{S}$ is an empty schedule **then** // Checks if $\mathcal{S}$ is infeasible
7: **break**
8: **end if**
9: Set $\mathcal{F} \leftarrow \mathcal{F} \cup \{\mathcal{S}\}$
10: $\hat{K} \leftarrow \hat{K} - 1$
11: **end while**
12: **return** the set of non-dominated solutions in $\mathcal{F}$

times $p_1 = 2$, $p_2 = p_3 = 9$, and $p_4 = 10$, to be scheduled on $M = 3$ machines with $u_1 = u_2 = u_3 = 1$, and a number $K = 10$ of time slots. The lower bound for $C^{\max}$ given by (3.10) is equal to 10, but there is no feasible solution with such makespan. In fact, jobs 2, 3, and 4 have to be scheduled on three different machines, without leaving two adjacent time slots for job 1.

SGS can be enhanced by improving the solution $\mathcal{S}$, computed at line 5 with SGH, by means of ES before line 9. Let us refer to such algorithm as Split-Greedy Scheduler with Exchange Search (SGS-ES).

3.4 Exact Algorithm

The exact algorithm heavily relies on the mathematical models described in Chapter 2, and it exploits the ϵ-constraint paradigm similarly to SGS (Section 3.3). For the sake of compactness, this section describes the exact algorithm with Formulation 2. The algorithm can be modified to work with Formulation 1 with minor changes.

First, let us define the *reduced formulation* of the BPMSTP as the optimization of (2.15) subject to constraints (2.16)-(2.18) and (2.21). In other words, the reduced formulation only requires the minimization of the TEC (the makespan is discarded) without considering constraints (2.19) and (2.20) that are related to the makespan.

Algorithm 6 reports the pseudo-code for an exact algorithm for the BPMSTP. Algo-

Algorithm 6 Exact algorithm for the BPMSTP

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

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

rithm 6 takes a BPMSTP instance $\mathcal{I}$ as input, and returns the set of the optimal solutions for $\mathcal{I}$. The algorithm first initializes the solution set $\mathcal{O}$ at line 1 and the parameter $\hat{K}$ at line 2. The latter is used in the downsized instances within the subsequent loop. Then, it repeats lines 3–11 until either $\hat{K}$ is lower than the lower bound $\underline{K}(\mathcal{I})$ or an infeasible solution is obtained before reaching $\underline{K}(\mathcal{I})$. In more detail, Algorithm 6 solves the reduced formulation associated with the downsized instance $\mathcal{D}(\mathcal{I}, \hat{K})$ with MILP at line 4. Then, if no feasible solution exists, the loop is stopped at line 6. Otherwise, Algorithm 6 calls Algorithm 1 to obtain a representation of the optimal solution of $\mathcal{D}(\mathcal{I}, \hat{K})$ as a feasible schedule at line 8. Afterwards, it adds the new solution $\mathcal{S}^\star$ to the set of solutions $\mathcal{O}$ at line 9. Observe that, after line 9, any solution $\mathcal{S}'$ to $\mathcal{D}(\mathcal{I}, \hat{K})$ with makespan $C^{\max}(\mathcal{S}')$ such that $C^{\max}(\mathcal{S}^\star) \leq C^{\max}(\mathcal{S}') \leq \hat{K}$ is either equivalent to or dominated by $\mathcal{S}^\star$. Hence, at line 10, Algorithm 6 updates the number of slots $\hat{K}$ for the next iteration as $C^{\max}(\mathcal{S}^\star) - 1$. Finally, it returns the set $\mathcal{O}$ of Pareto optimal solutions at the end of the loop.

The computational efficiency of step 4 in Algorithm 6 can be enhanced by providing an initial feasible solution for the MILP solver computed by an ad-hoc heuristic, such as SGS or SGS-ES, described in the previous section. The advantages of such choice will be investigated in Chapter 4 in comparison to the general-purpose heuristics for initialization used in commercial MILP solvers.

Chapter 4

Numerical results

This chapter reports the results of the tests aimed at experimentally evaluating the algorithms described in Chapter 3. The tests were motivated by the goals of assessing:

1. the performances of SGH with respect to CH;
2. the impact of ES on the quality of the solutions computed by SGH;
3. the performances of SGS-ES with respect to other state-of-the-art algorithms;
4. the relative performances of the exact algorithm with the two mathematical models;
5. the impact of the initial solution provided by SGH on the computational efficiency of the exact algorithm;
6. the performances of SGS-ES with respect to the exact algorithm with Formulation 2.

The first goal is essential in experimentally evaluate SGH with respect to the state-of-the-art for the BPMSTP. The second goal is important to check whether the improvement in solutions quality enabled by the application of ES justifies its computational times. The third goal is a fundamental step towards establishing, from an experimental standpoint, SGS-ES as a state-of-the-art heuristic scheme for the BPMSTP. The fourth goal, instead, deals with the the exact algorithm. Specifically, it is focused on highlighting the higher computational efficiency of Formulation 2 with respect to Formulation 1 as a part of the MILP solving step. The fifth goal again focuses again on the exact algorithm, by further enhancing its computational efficiency with the use of SGH to find the initial

feasible solution. Finally, the sixth goal is fundamental in evaluating the optimality gap of the solutions provided by SGS-ES. Moreover, a comparison of the computational times achieved by SGS-ES and the exact algorithm allows us to investigate the impact of the trade-off between the solutions quality and the computational efficiency.

This chapter is organized as follows. Section 4.1 describes the benchmark instances. Then, Section 4.2 provides details on the algorithms implementation, while Section 4.3 describes the performance metrics used to evaluate the algorithms performances. Section 4.4 experimentally evaluates the impact of the shortcomings of CH, described in Section 3.1 of Chapter 3. Section 4.5 compares SGS with SGS-ES in order to evaluate the introduction of ES in the heuristic scheme. This section concludes that the quality–computational time trade-off of introducing ES in the heuristic scheme is very favorable for all the instances. Then, Section 4.6 compares the performances of SGS-ES with the ones achieved by NSGA-III, MOEA/D, and CH. Section 4.7 compares the two mathematical models described in Chapter 2. Afterwards, Section 4.8 evaluates the impact of the heuristic scheme in speeding up the computational process the exact algorithm by providing an initial feasible solution. Finally, Section 4.9 shows the relative performances of SGS-ES and the exact algorithm.

The experimental tests in Sections 4.4, 4.5, and 4.6 were carried out on a Windows 10 system equipped with a Intel(R) Core i7-8750H CPU @ 2.20GHz, 6 cores processor, and 16 gigabytes of RAM. Due to machine unavailability, the tests in Section 4.7, Section 4.8, and Section 4.9 were performed on another machine with a Windows 10 system, equipped with a Intel Core i9-9900K Octa-core 3.6 GHz processor as well as 16 GB of RAM.

4.1 Instances

The tests were performed on a benchmark that consists of a set of 60 instances proposed by Wang et al. [108], and another set of 30 instances proposed in our work [9].

The first set of instances consists of 30 small-scale instances (numbered from 1 to 30), and 30 medium-scale and large-scale instances (numbered from 31 to 60). The latter set is hereinafter referred to MLS instances for short.

The second set of instances contains 30 very large-scale instances, or VLS instances for short (numbered from 61 to 90). The value of the number of machines M, the number of jobs N, and the number of time slots K in each instance of the VLS set is given by an element (M, N, K) of the cartesian product between $\{25, 30, 40\}$, $\{250, 300, 350, 400, 500\}$,

Instance	Problem data				Instance	Problem data				Instance	Problem data			
	N	M	K	$\lvert\mathcal{P}_\mathcal{J}\rvert$		N	M	K	$\lvert\mathcal{P}_\mathcal{J}\rvert$		N	M	K	$\lvert\mathcal{P}_\mathcal{J}\rvert$
1	6	3	50	4	31	30	8	100	3	61	250	25	350	12
2	6	3	80	4	32	60	8	100	3	62	250	25	500	12
3	6	5	50	4	33	100	8	100	3	63	300	25	350	12
4	6	5	80	3	34	150	8	100	3	64	300	25	500	12
5	6	7	50	5	35	200	8	100	4	65	350	25	350	12
6	6	7	80	3	36	30	16	100	3	66	350	25	500	12
7	10	3	50	5	37	60	16	100	3	67	400	25	350	12
8	10	3	80	5	38	100	16	100	3	68	400	25	500	12
9	10	5	50	4	39	150	16	100	3	69	500	25	350	12
10	10	5	80	5	40	200	16	100	4	70	500	25	500	12
11	10	7	50	5	41	30	20	100	3	71	250	30	350	12
12	10	7	80	4	42	60	20	100	3	72	250	30	500	12
13	15	3	50	5	43	100	20	100	3	73	300	30	350	12
14	15	3	80	5	44	150	20	100	3	74	300	30	500	12
15	15	5	50	4	45	200	25	100	4	75	350	30	350	12
16	15	5	80	5	46	30	8	300	4	76	350	30	500	12
17	15	7	50	5	47	60	8	300	4	77	400	30	350	12
18	15	7	80	5	48	100	8	300	4	78	400	30	500	12
19	20	3	50	5	49	150	8	300	4	79	500	30	350	12
20	20	3	80	5	50	200	8	300	4	80	500	30	500	12
21	20	5	50	5	51	30	16	300	4	81	250	40	350	12
22	20	5	80	5	52	60	16	300	4	82	250	40	500	12
23	20	7	50	5	53	100	16	300	4	83	300	40	350	12
24	20	7	80	5	54	150	16	300	4	84	300	40	500	12
25	25	3	50	5	55	200	16	300	4	85	350	40	350	12
26	25	3	80	5	56	30	25	300	4	86	350	40	500	12
27	25	5	50	5	57	60	25	300	4	87	400	40	350	12
28	25	5	80	5	58	100	25	300	4	88	400	40	500	12
29	25	7	50	5	59	150	25	300	4	89	500	40	350	12
30	25	7	80	5	60	200	25	300	4	90	500	40	500	12

Table 4: For each instance, this table reports the number of jobs N, the number of machines M, the number of time slots K, and the number of distinct processing times $|\mathcal{P}_\mathcal{J}|$. The maximum processing time $p_{\max}$ is equal to $|\mathcal{P}_\mathcal{J}|$ for all the instances except for instances 1, 3, 4, 6, 9, 12 and 15, where $p_{\max}$ is equal to 5.

and $\{350, 500\}$. The processing times p_j, $j \in \mathcal{J}$, were randomly drawn from the uniform distribution $U[1, 12]$. In order to reflect the tendencies of a highly volatile electricity market, the values of the consumption rate u_h, $h \in \mathcal{H}$, and the values of the time slot costs c_k, $k \in \mathcal{T}$, were randomly drawn from the uniform distributions $U[1, 6]$ and $U[1, 8]$, respectively. Such choice fostered the generation of smaller time intervals with respect to the first 60 instances.

Table 4 shows the parameters N, M, K, and $|\mathcal{P}_\mathcal{J}|$ for each of the 90 instances in the benchmark. For compactness, $p_{\max}$ is not shown since it is equal to $|\mathcal{P}_\mathcal{J}|$ for all the instances, except for instances 1, 3, 4, 6, 9, 12 and 15, where $p_{\max}$ is equal to 5. The whole benchmark is available at *https://github.com/ORresearcher/PhD-thesis*.

4.2 Implementation

All the algorithms were implemented in Java 16. In particular, the implementation of the exact algorithm also employs the Java CPLEX 20.1.0 API. Furthermore, the exact algorithm accepts solutions with an optimal gap that does not exceed 10^{-4}.

SGS-ES was compared with three distinct state-of-the-art solution approaches: CH, NSGA-III [34], and MOEA/D [123]. CH was first implemented by strictly following the description of Wang et al. [108]. Such implementation is referred to as *CH-M*, since it is the Java version of the MATLAB implementation kindly shared by the authors. Then, CH was reimplemented by building upon CH-M, and correcting the identified shortcomings. Such second implementation, called *CH-J*, is the reference CH implementation for the comparisons in this thesis.

NSGA-III is a genetic algorithm that builds upon NSGA-II [35]. NSGA-II was proposed by Deb et al. [35] as a further step towards a computationally efficient genetic algorithm for multi-objective optimization. It was used by Wang et al. [108] as a reference state-of-the-art algorithm as the benchmark for CH performances. NSGA-III differs from NSGA-II in the operator used to select the solutions to be included in the population. In fact, in order to ensure the diversity of the solutions, NSGA-III adopts a predefined set of reference points instead of the niche-preservation operator based on the crowding distance exploited by NSGA-II. The Java implementation of NSGA-III is based on the MATLAB implementation of NSGA-II of Wang et al. [108]. The design of NSGA-II was preserved, i.e., the Java implementation of NSGA-III used the same solution representation, initialization procedure, crossover and mutation operators, and the same parameters setting as well.

MOEA/D is an evolutionary multi-objective algorithm based on a similar idea. In fact, it defines a set of weight vectors a priori, so that each generated solution is uniquely associated with a single vector to guarantee solutions diversity. The MOEA/D algorithm presented by Zhang and Li [123] was implemented for the BPMSTP by exploiting some parts of NSGA-III design, i.e., the same solution representation, initialization procedure, crossover operator, and termination condition. In addition, at each iteration, MOEA/D generates the candidate solutions to possibly update the ones currently associated with each weight vector by means of the crossover operator. This operator is applied to a pair of parent solutions randomly drawn from the set including the CW solutions associated with the closest weight vectors, where the parameter CW is fixed to 10 as in [123].

4.3 Performance metrics

This chapter uses two different classes of state-of-the-art performance metrics to evaluate the performances of multi-objective algorithms. The metrics in the first class evaluate the quality of the Pareto front, while the ones in the second class provide a measure of the distribution uniformity of the non-dominated points in the front [12].

Let $\mathcal{F} \subseteq \mathbb{R}^n$ be an n-dimensional Pareto front, and $\mathcal{F}^\star$ be its reference optimal Pareto front. In practice, if $\mathcal{F}^\star$ is not known, then it is approximated as the set of the non-dominated solutions in the union of the fronts computed by the algorithms under comparison. The first class of metrics is constituted by the three following ones:

- D_R [60], given by
$$D_R(\mathcal{F}) = \frac{1}{|\mathcal{F}^\star|} \sum_{y \in \mathcal{F}^\star} \min\{d_{x,y} : x \in \mathcal{F}\}$$
where $\mathcal{F}^\star$ is the optimal Pareto front, and $d_{x,y}$ is the Euclidean distance between the two points x and y in $\mathcal{F}$ and $\mathcal{F}^\star$, respectively. Smaller values of D_R denote higher quality fronts;

- *Purity* [12], given by
$$P(\mathcal{F}) = \frac{N_d(\mathcal{F}^\star \cap \mathcal{F})}{N_d(\mathcal{F})},$$
where $N_d(\mathcal{F})$ is the number of efficient solutions in $\mathcal{F}$. The value of $P(\mathcal{F})$ is the ratio between the number of non-dominated points in the intersection of the reference Pareto front $\mathcal{F}^\star$ and the front $\mathcal{F}$, and the number of non-dominated points in $\mathcal{F}$. As a consequence, larger values of Purity denote higher quality fronts.

- *Hypervolume* [53, 129], that measures the hypervolume covered by $\mathcal{F}$ with respect to a reference point in the objectives space. 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 value of $\mathcal{F}$ is the measure of the region weakly dominated by $\mathcal{F}$ and bounded above by r, i.e.,
$$H(\mathcal{F}) = \Lambda(\{q \in \mathbb{R}^n : \exists p \in \mathcal{F} : p \leq q \leq r\}),$$
where $\Lambda(\cdot)$ is the Lebesgue measure. Larger Hypervolume values denote a better approximation of the optimal Pareto front.

Let also $f_k(x)$, $k = 1, \ldots, n$, be the the value of the k-th objective function in $\mathcal{F}$ computed for some $x \in \mathcal{F}$. The second class of metrics is constituted by the two following ones:

- *Spacing* [12], given by

$$SP(\mathcal{F}) = \sqrt{\frac{1}{N_d(\mathcal{F})} \sum_{i=1}^{N_d(\mathcal{F})} (\delta_i - \overline{\delta})^2}$$

  where

$$\delta_i = \min_{j \neq i} \sum_{k=1}^{n} |f_k(x_i) - f_k(x_j)|,$$

  and

$$\overline{\delta} = \frac{1}{N_d(\mathcal{F}) - 1} \sum_{i=1}^{N_d(\mathcal{F})-1} \delta_i,$$

  The smaller the values of Spacing, the better the uniformity of the distribution of the solutions in $\mathcal{F}$.

- *Spread* [35], also denoted as D_2 [108], given by

$$D_2(\mathcal{F}) = \frac{d_f + d_l + \sum_{i=1}^{N_d(\mathcal{F})-1} |d_i - \overline{d}|}{d_f + d_l + (N_d(\mathcal{F}) - 1)\overline{d}},$$

  where

$$d_i = \sqrt{\sum_{k=1}^{n} \Big(f_k(x_i) - f_k(x_{i+1})\Big)^2},$$

  being x_i and x_{i+1} two neighboring points in $\mathcal{F}$, and

$$\overline{d} = \frac{1}{N_d(\mathcal{F}) - 1} \sum_{k=1}^{n} d_i.$$

  Finally, d_f and d_l are the Euclidean distances between the extreme solutions and the boundary solutions of the Pareto-optimal solutions in $\mathcal{F}$ [35]. The smaller the values of Spread, the better the uniformity of the distribution of the solutions in $\mathcal{F}$.

Observe that distribution metrics are secondary to quality metrics. In fact, distribution metrics should be used as a complementary evaluation metrics for fronts with a comparable degree of quality. As an example, let us consider a well distributed front $\mathcal{F}$ that is totally

dominated by another front $\mathcal{F}'$ which is instead poorly distributed, but qualitatively higher than $\mathcal{F}$. In such a case, distribution indices allow one to perform a more accurate comparison, thanks to the better insight on the structure of the Pareto fronts.

When the quality metrics allow one to draw clear conclusions, distribution metrics have limited use in enhancing the comparisons. In fact, the distribution of the Pareto-optimal points in a front depend on the related problem, and problem instance as well. The structure of such a specific optimal Pareto front cannot be captured by a general-purpose metric that aims at measuring the distance between distinct non-dominated points, regardless of the nature of the problem.

This thesis also considers the *Empirical Attainment Function* (EAF) [33] as a further metric to evaluate performances. EAF provides a graphical representation of the probability of an algorithm of generating solutions that dominate a given point of the objective space in a single run [80]. In particular, the *Differential Empirical Attainment Function* (Diff-EAF) proposed in [75] visually shows the differences between a pair of EAFs, by highlighting the regions of the objective space in which an algorithm outperforms another in terms of generated solutions. However, differently from the other metrics, EAF and Diff-EAF are only a visual tools for comparison, as they do not provide an overall score.

Finally, the thesis uses two further metrics, FM_1 and FM_2, in order to evaluate the feasibility of a front generated by an algorithm. Such metrics are specifically introduced to measure the number of infeasible solutions in the Pareto fronts generated by CH.

For a given Pareto front $\mathcal{F}$, $FM_1(\mathcal{F})$ is the percentage of infeasible points in $\mathcal{F}$:

$$FM_1(\mathcal{F}) = \frac{|\mathcal{U}(\mathcal{F})|}{|\mathcal{F}|}, \tag{4.1}$$

where $\mathcal{U}(\mathcal{F})$ denotes the set of infeasible points of $\mathcal{F}$. Instead, FM_2 is the average percentage of non-scheduled jobs:

$$FM_2(\mathcal{F}) = \begin{cases} \frac{|\mathcal{J}^{\text{ns}}|}{|\mathcal{U}(\mathcal{F})|N} & \text{if } FM_1(\mathcal{F}) \neq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{4.2}$$

where $\mathcal{J}^{\text{ns}}$ is the set of non-scheduled jobs in the infeasible solutions in $\mathcal{F}$.

4.4 Evaluating the limits of CH

This section performs an experimental evaluation of the limits of the implementation of the original version of CH, i.e., CH-M. Indeed, as discussed in Chapter 3, CH [108] may mistakenly add infeasible solutions in the computed Pareto front.

Table 5 reports the feasibility metrics FM_1 and FM_2 for each of the fronts computed by CH-M for each the last 60 of instances. Let us observe that 13.33% of the computed Pareto front for the MLS instances includes infeasible solutions. Moreover, an average of 1% of the solutions are infeasible, since an average of 2.58% of the jobs are not scheduled. Let us now consider the results obtained on the set of VLS instances. For 90% of the instances, the computed Pareto fronts include solutions with unscheduled jobs, even though the average numbers of solutions and unscheduled jobs are close to the relative ones for the MLS instances. In light of these results, we conclude that CH-J is indeed the best implementation of CH to ensure fair comparisons.

4.5 Evaluating Exchange Search

This section compares SGS and SGS-ES so as to evaluate the computational efficiency of ES with respect to the quality of the computed solutions.

Table 6 reports the Hypervolume, Purity, and Spread metrics, as well as the computational times (in seconds), achieved by SGS and SGS-ES on the last 60 instances, grouped by the MLS and the VLS sets. In addition, for each metric and each of the two compared algorithms, the third-to-last row shows the average performance, while the second-to-last row reports the number of times the considered algorithm achieved the best performance over the instances in the set. For each instance, the best values of each metric are highlighted in bold. The last row reports the probability value obtained with the Friedman non-parametric test. If p is smaller than 1%, then we can reject the null hypothesis that SGS and SGS-ES generate results that do not differ significantly from a statistical standpoint.

The values reported in Table 6 were obtained by averaging the results of the comparisons between each of the 10 runs of SGS with each of the 10 runs of SGS-ES, due to the stochasticity of SGH in both SGS and SGS-ES. Table 6 shows that SGS-ES prevailed in the Hypervolume and Purity metrics on both the sets of instances, whereas SGS got slightly better average values for Spread. The statistical tests indeed reveal that SGS-

Instance	FM1	FM2	Instance	FM1	FM2
31			61	0.81%	2.00%
32			62	0.58%	2.80%
33			63	0.83%	1.50%
34			64	0.85%	3.00%
35			65	0.41%	0.57%
36			66		
37			67	1.81%	3.50%
38			68	1.17%	3.06%
39			69	0.51%	1.60%
40			70	1.23%	2.40%
41			71	1.28%	2.00%
42			72	0.78%	1.40%
43	1.54%	3.00%	73	1.72%	3.42%
44			74		
45			75	1.73%	3.93%
46			76	1.44%	4.14%
47			77	0.83%	3.25%
48			78	0.55%	4.25%
49	0.95%	2.67%	79	1.30%	2.47%
50			80	1.37%	2.00%
51			81	1.72%	3.40%
52			82	0.31%	0.80%
53			83	0.43%	1.00%
54	0.79%	2.67%	84	0.30%	0.33%
55			85	1.69%	3.43%
56			86		
57			87	0.45%	0.50%
58			88	0.61%	3.75%
59	0.72%	2.00%	89	0.43%	3.40%
60			90	0.60%	1.80%
Average	1.00%	2.58%		0.95%	2.43%
Non-feasible	13.33%			90.00%	

Table 5: The two feasibility metrics FM_1 and FM_2 applied to results achieved by CH-M on instances 31–90.

ES is significantly better than SGS for Hypervolume and Purity, whereas the results for Spread are statistically comparable. Furthermore, the comparison of the computational times reveals that SGS-ES is unavoidably more computationally demanding than SGS, since the former algorithm uses ES to improve the quality of the solutions. However, the average and the maximum computational time required by SGS-ES are acceptable for the MLS instances. As regards the VLS instances, the average time is about 67 seconds, while the maximum time is about 161 seconds. Moreover, the computational times exceeds 100 seconds in only 6 of the VLS instances. Such computational times can be considered acceptable as well, especially in view of the achieved improvement in the performances.

Table 6: Comparison of the results achieved by SGS and SGS-ES on the MLS and the VLS instances based on the Hypervolume, Purity, Spread, and CPU time (s) metrics.

Instance	Hypervolume		Purity		Spread		CPU		Instance	Hypervolume		Purity		Spread		CPU	
	SGS-ES	SGS	SGS-ES	SGS	SGS-ES	SGS	SGS-ES	SGS		SGS-ES	SGS	SGS-ES	SGS	SGS-ES	SGS	SGS-ES	SGS
31	**0.8813**	0.8790	**0.9232**	0.4475	**0.9173**	0.9292	0.0518	**0.0104**	61	**0.8198**	0.8130	**0.9994**	0.0068	1.0602	**1.0461**	21.7450	0.8068
32	**0.8090**	0.8082	**0.9742**	0.8478	0.9482	**0.9458**	0.0738	**0.0181**	62	**0.8295**	0.8220	**1.0000**	0.0032	**1.0119**	1.0160	62.3640	1.7699
33	**0.7791**	0.7777	**0.9876**	0.5517	**0.8728**	0.8882	0.0937	**0.0254**	63	**0.7660**	0.7599	**1.0000**	0.0078	0.8898	**0.8866**	25.8207	0.8211
34	**0.6960**	0.6901	**0.9510**	0.2159	0.7688	**0.7570**	0.1220	**0.0299**	64	**0.7862**	0.7805	**0.9996**	0.0033	0.9226	**0.9076**	57.0123	1.8385
35	**0.6011**	0.6009	**1.0000**	0.9541	0.5066	**0.5027**	0.0586	**0.0251**	65	**0.7245**	0.7116	**1.0000**	0.0044	0.8744	**0.8567**	30.4711	0.8903
36	**0.8542**	0.8518	**0.9496**	0.6613	0.7243	**0.7106**	0.0810	**0.0171**	66	**0.8014**	0.7910	**1.0000**	0.0075	**0.9823**	0.9981	75.7831	2.0618
37	**0.8709**	0.8695	**0.9463**	0.4795	0.8880	**0.8757**	0.1202	**0.0214**	67	**0.7581**	0.7518	**1.0000**	0.0060	0.8073	**0.7831**	25.9146	0.8793
38	**0.8409**	0.8399	**0.8638**	0.4692	**1.1245**	1.1427	0.1838	**0.0317**	68	**0.7391**	0.7266	**0.9998**	0.0046	0.8607	**0.8409**	84.8752	2.1309
39	**0.7830**	0.7814	**0.9895**	0.6211	**0.8684**	0.8748	0.2712	**0.0430**	69	**0.7368**	0.7289	**1.0000**	0.0188	0.8101	**0.8012**	25.9769	0.9363
40	**0.7827**	0.7730	**0.9553**	0.2247	**0.8447**	0.8456	0.3673	**0.0548**	70	**0.7508**	0.7378	**0.9999**	0.0064	0.8576	**0.8521**	90.3612	2.4123
41	**0.9282**	0.9282	**1.0000**	0.9938	**0.8120**	0.8125	0.0815	**0.0161**	71	**0.8057**	0.7987	**1.0000**	0.0000	**0.9612**	0.9786	25.2129	0.8829
42	**0.8529**	0.8523	**0.9943**	0.8179	**0.9458**	0.9489	0.1289	**0.0252**	72	**0.8735**	0.8691	**1.0000**	0.0000	1.0414	**1.0401**	65.3719	2.0116
43	**0.8613**	0.8599	**0.8563**	0.3881	**1.0217**	1.0239	0.1995	**0.0338**	73	**0.7955**	0.7878	**1.0000**	0.0041	0.9906	**0.9669**	30.7839	0.9326
44	**0.8312**	0.8301	**0.7902**	0.4311	1.0712	**1.0695**	0.3085	**0.0481**	74	**0.8508**	0.8456	**0.9992**	0.0048	1.0319	**1.0230**	77.8728	2.1324
45	**0.7900**	0.7871	**0.9491**	0.3861	0.8651	**0.8599**	0.5005	**0.0622**	75	**0.7888**	0.7825	**1.0000**	0.0043	0.9003	**0.8973**	34.7806	0.9713
46	**0.8242**	0.8194	**0.9145**	0.4845	**0.5920**	0.6045	0.4204	**0.1037**	76	**0.8508**	0.8438	**1.0000**	0.0059	0.9747	**0.9655**	92.2981	2.2562
47	**0.8832**	0.8811	**0.8538**	0.4783	**0.7928**	0.7968	0.7459	**0.1384**	77	**0.7478**	0.7404	**0.9999**	0.0050	0.8656	**0.8538**	37.2449	1.0383
48	**0.8724**	0.8676	**0.9573**	0.1964	0.7753	**0.7696**	0.9804	**0.1999**	78	**0.7890**	0.7781	**1.0000**	0.0058	**0.9621**	0.9639	107.1587	2.4133
49	**0.8025**	0.7972	**0.9686**	0.1526	0.8234	**0.8118**	1.4556	**0.2815**	79	**0.7464**	0.7371	**1.0000**	0.0128	0.8274	**0.7982**	45.8857	1.1171
50	**0.8275**	0.8226	**0.9864**	0.3545	**0.6961**	0.7183	1.8296	**0.3628**	80	**0.8158**	0.8099	**1.0000**	0.0035	0.9324	**0.9251**	101.0368	2.6189
51	**0.8457**	0.8427	**0.9286**	0.6523	0.8498	**0.8278**	0.7754	**0.1494**	81	**0.8201**	0.8147	**0.9987**	0.0062	**0.9468**	0.9510	37.5365	1.0809
52	**0.8657**	0.8642	**0.8922**	0.5680	**0.7515**	0.7907	1.1850	**0.1941**	82	**0.8646**	0.8596	**0.9970**	0.0066	**1.1053**	1.1441	92.0232	2.4982
53	**0.8875**	0.8864	**0.9411**	0.4613	0.8235	**0.8185**	1.6191	**0.2666**	83	**0.8199**	0.8135	**0.9992**	0.0024	**1.0663**	1.0665	42.3040	1.1566
54	**0.8836**	0.8825	**0.9019**	0.4204	**0.6813**	0.7081	2.1862	**0.3516**	84	**0.8282**	0.8230	**0.9998**	0.0041	**0.9831**	0.9930	113.6161	2.5701
55	**0.8445**	0.8417	**0.9874**	0.4762	0.8538	**0.8484**	3.3929	**0.4477**	85	**0.8237**	0.8175	**1.0000**	0.0030	**1.0316**	1.0450	49.3493	1.2331
56	**0.8867**	0.8852	**0.9792**	0.7035	0.8028	**0.7363**	1.1292	**0.1762**	86	**0.8308**	0.8248	**0.9999**	0.0007	**0.9742**	0.9781	129.1932	2.7451
57	**0.7435**	0.7377	**0.9392**	0.4743	0.6317	**0.6155**	1.7694	**0.2853**	87	**0.8091**	0.8034	**1.0000**	0.0003	0.9481	**0.9412**	52.3694	1.2788
58	**0.9057**	0.9047	**0.9960**	0.5480	**0.6884**	0.7099	2.4266	**0.3276**	88	**0.8372**	0.8308	**1.0000**	0.0033	**1.0088**	1.0124	149.7154	2.9976
59	**0.9012**	0.9002	**0.8873**	0.4814	0.9665	**0.9515**	4.6183	**0.4209**	89	**0.7629**	0.7485	**1.0000**	0.0004	0.9555	**0.9426**	72.6986	1.3990
60	**0.8169**	0.8150	**0.9478**	0.5063	0.7862	**0.7788**	4.2915	**0.5277**	90	**0.8093**	0.8004	**0.9998**	0.0066	1.0377	**1.0320**	161.0501	3.2595
Avg	**0.8318**	0.8292	**0.9404**	0.5149	0.8231	**0.8225**	1.0489	**0.1565**		**0.7994**	0.7918	**0.9997**	0.0050	0.9541	**0.9502**	67.2609	**1.7047**
N. best	**30**	0	**30**	0	14	16	0	**30**		**30**	0	**30**	0	11	19	0	30
p	7.24E-04		4.32E-04		0.715		4.32E-04			4.32E-04		4.32E-04		0.1441		4.32E-04	

4.6 Comparing SGS-ES with other state-of-the-art heuristics

Table 7, 8, 9 and 10 reports the results of the comparisons between SGS-ES, CH-J, NSGA-III and MOEA/D on the MLS and the VLS instances. The computational times (CPU) are expressed in seconds. Furthermore, the last two rows of the tables allow us to determine, for each metric, the ranks of the four algorithms under comparison. Specifically, for each metric, the second-to-last row reports the mean ranks achieved by the four algorithms, according to the Friedman test based multiple comparison test. For each metric, the last row shows the relative ranking of the four algorithms with a 99% confidence interval. A lower rank denotes a better algorithm, and algorithms that did not generate statistically different results have the same rank.

Table 7 and 8 show the results achieved by the four algorithms on the MLS set. In more detail, Table 7 reports the algorithms performances according to the three quality metrics. SGS-ES achieves the best value for Hypervolume on each instance. The only instance where SGS-ES achieves an inferior score, according to the Purity and D_R metrics, is instance 41. Instead, Table 8 reports the algorithms performances according to the two distribution metrics, as well as the computational times. As regards the values of the distribution metrics, Table 8 shows that SGS-ES attained the best average results for Spacing, whereas MOEA/D has the best average Spread. The p values denote that the results are statistically equivalent. In addition, the algorithms have the same rank on Spacing, whereas only NSGA-III has a better rank than SGS-ES on Spread. SGS-ES and CH-J are also the best algorithms from a computational efficiency standpoint. Indeed, even though SGS-ES needed on average less than half the computational time of CH-J, they share the same rank.

Table 9 and 10 show the results achieved by the four algorithms on the VLS set. In particular, Table 9 reports the values of the quality metrics attained by the four algorithms. SGS-ES significantly outperforms the other algorithms, since it achieved the best values on all the metrics, and it is always ranked first. In particular, it is worth observing that, except for instance 74, SGS-ES achieved the best score for Purity, while CH, NSGA-III, and MOEA/D achieved the worst. SGS-ES also achieved the best score for D_R on the whole VLS set. Table 10 reports the values of the distribution metrics, and the computational times as well. CH-J achieved the best result for Spread and Spacing. However, the p values reveal the statistical equivalence of the four algorithms. Moreover,

it is important to observe that SGS-ES attained significantly better computational times with respect to the other algorithms.

Finally, Figure 14 reports the Diff-EAF's plots of SGS-ES with respect to the other three algorithms, i.e., CH-J, NSGA-III and MOEA/D for instance 49. Such plots were obtained by using the visualization tool proposed by López-Ibánez et al. [76].

The comparisons of the three stochastic algorithms under comparison, that is, SGS-ES, NSGA-III and MOEA/D, were carried by considering the results achieved in 10 distinct runs. Each diagram plots a lower and an upper line that connect the best and worst points attained by the compared methods in all the runs, respectively. The plots also show a dashed line corresponding to the median attained by each algorithm. The areas highlighted with different shades of grey show the points where the EAF of an algorithm is larger than the one of the compared algorithm of at least 20% (the darker is the area, the larger is the difference). Observe that, according to all the pairs of diagrams, SGS-ES computed better results than CH-J, NSGA-III and MOEA/D.

As pointed out in Section 4.3, a thorough analysis using Diff-EAF's is not practical. However, the results obtained by considering similar diagrams for several instances were similar to the ones depicted in Figure 14.

Table 7: Comparison of the results achieved by SGS-ES, CH-J, NSGA-III and MOEA/D on the MLS instances based on the Hypervolume, Purity, and D_R metrics.

Instance	**Hypervolume**				**Purity**				D_R			
	SGS-ES	**CH-J**	**NSGA-III**	**MOEA/D**	**SGS-ES**	**CH-J**	**NSGA-III**	**MOEA/D**	**SGS-ES**	**CH-J**	**NSGA-III**	**MOEA/D**
31	**0.8816**	0.8717	0.8717	0.8717	**0.99556**	0.0540	0.0540	0.0540	**0.0000**	0.0117	0.0116	0.0117
32	**0.8135**	0.8006	0.8006	0.8006	**1.00000**	0.0981	0.0983	0.0981	**0.0000**	0.0103	0.0102	0.0103
33	**0.7772**	0.7736	0.7736	0.7736	**1.00000**	0.3702	0.3702	0.3702	**0.0000**	0.0032	0.0032	0.0032
34	**0.6962**	0.6838	0.6839	0.6839	**0.99349**	0.1082	0.1082	0.1082	**0.0000**	0.0120	0.0119	0.0119
35	**0.6011**	0.5875	0.5875	0.5875	**1.00000**	0.3889	0.3889	0.3889	**0.0000**	0.0115	0.0115	0.0115
36	**0.8868**	0.8771	0.8773	0.8771	**1.00000**	0.5043	0.5304	0.5043	**0.0000**	0.0079	0.0076	0.0079
37	**0.8709**	0.8658	0.8658	0.8658	**0.98307**	0.2972	0.2992	0.2972	**0.0001**	0.0069	0.0068	0.0069
38	**0.8405**	0.8380	0.8380	0.8380	**0.84029**	0.4113	0.4113	0.4113	**0.0006**	0.0039	0.0039	0.0039
39	**0.7793**	0.7742	0.7742	0.7742	**1.00000**	0.3333	0.3333	0.3333	**0.0000**	0.0052	0.0052	0.0052
40	**0.7827**	0.7687	0.7687	0.7687	**0.98297**	0.1188	0.1188	0.1188	**0.0001**	0.0109	0.0109	0.0109
41	**0.9282**	0.9276	0.9280	0.9280	0.93750	0.9267	**0.9933**	**0.9933**	0.0016	0.0016	**0.0000**	**0.0000**
42	**0.8569**	0.8541	0.8541	0.8541	**1.00000**	0.7143	0.7143	0.7143	**0.0000**	0.0031	0.0031	0.0031
43	**0.8612**	0.8573	0.8573	0.8573	**0.96216**	0.1918	0.1918	0.1918	**0.0002**	0.0045	0.0045	0.0045
44	**0.8312**	0.8285	0.8285	0.8285	**0.80172**	0.3508	0.3508	0.3508	**0.0004**	0.0029	0.0029	0.0029
45	**0.7900**	0.7800	0.7800	0.7800	**0.86024**	0.2411	0.2411	0.2411	**0.0006**	0.0084	0.0084	0.0084
46	**0.8173**	0.8096	0.8114	0.8096	**0.89985**	0.3596	0.4207	0.3596	**0.0006**	0.0072	0.0068	0.0072
47	**0.8949**	0.8903	0.8903	0.8903	**0.94731**	0.3377	0.3377	0.3377	**0.0001**	0.0134	0.0134	0.0134
48	**0.8724**	0.8654	0.8654	0.8654	**0.87835**	0.2500	0.2500	0.2500	**0.0004**	0.0076	0.0076	0.0076
49	**0.8059**	0.7865	0.7865	0.7865	**1.00000**	0.0000	0.0000	0.0000	**0.0000**	0.0158	0.0158	0.0159
50	**0.8275**	0.8119	0.8120	0.8119	**0.99789**	0.3222	0.3222	0.3222	**0.0000**	0.0134	0.0134	0.0134
51	**0.8492**	0.8397	0.8420	0.8420	**0.81529**	0.4771	0.5600	0.5486	**0.0051**	0.0107	0.0056	0.0058
52	**0.8681**	0.8657	0.8658	0.8658	**0.79728**	0.5429	0.5429	0.5429	**0.0022**	0.0101	0.0101	0.0101
53	**0.8875**	0.8847	0.8847	0.8847	**0.93060**	0.3286	0.3286	0.3286	**0.0002**	0.0058	0.0058	0.0058
54	**0.8836**	0.8803	0.8803	0.8803	**0.97846**	0.1765	0.1765	0.1765	**0.0001**	0.0056	0.0056	0.0056
55	**0.8445**	0.8411	0.8411	0.8411	**0.97501**	0.4133	0.4133	0.4133	**0.0000**	0.0026	0.0026	0.0026
56	**0.8867**	0.8833	0.8841	0.8841	**0.87550**	0.4033	0.5278	0.5144	**0.0016**	0.0082	0.0054	0.0055
57	**0.7435**	0.7353	0.7363	0.7363	**0.85679**	0.5326	0.6011	0.5853	**0.0009**	0.0040	0.0031	0.0031
58	**0.9057**	0.9046	0.9046	0.9046	**1.00000**	0.4483	0.4483	0.4483	**0.0000**	0.0039	0.0039	0.0039
59	**0.9017**	0.8995	0.8995	0.8995	**0.96119**	0.3170	0.3170	0.3170	**0.0000**	0.0040	0.0040	0.0040
60	**0.8184**	0.8162	0.8162	0.8162	**0.93429**	0.3607	0.3607	0.3607	**0.0003**	0.0037	0.0037	0.0037
Avg	**0.8335**	0.8267	0.8270	0.8269	**0.94016**	0.3460	0.3604	0.3560	**0.0005**	0.0073	0.0070	0.0070
N. best	30	0	0	0	29	0	1	1	29	0	1	1
p			1.87E-17				4.49E-16				2.05E-15	
Mean rank	4	1.75	2.2	2.05	3.9333	1.8	2.2833	1.9833	1.0833	3.2167	2.6667	3.0333
Rank position	1	2	2	2	1	2	2	2	1	2	2	2

Table 8: Comparison of the results achieved by SGS-ES, CH-J, NSGA-III and MOEA/D on the MLS instances based on the Spread, Spacing, and CPU time (s) metrics.

Instance	**Spread**				**Spacing**				**CPU**			
	SGS-ES	**CH-J**	**NSGA-III**	**MOEA/D**	**SGS-ES**	**CH-J**	**NSGA-III**	**MOEA/D**	**SGS-ES**	**CH-J**	**NSGA-III**	**MOEA/D**
31	**0.9173**	0.9638	0.9662	0.9638	0.0453	**0.0443**	0.0445	**0.0443**	**0.0518**	0.2630	14.0060	2.8346
32	0.9482	0.8792	**0.8782**	0.8792	0.0304	**0.0294**	0.0295	**0.0294**	**0.0738**	0.1037	14.4933	3.3755
33	0.8728	**0.8562**	**0.8562**	**0.8562**	**0.0232**	0.0238	0.0238	0.0238	**0.0937**	0.1145	15.1787	4.9112
34	**0.7687**	0.7887	0.7887	0.7887	**0.0169**	0.0171	0.0170	0.0170	0.1220	**0.0928**	16.0313	6.4969
35	**0.5066**	0.6053	0.6053	0.6053	**0.0205**	0.0232	0.0232	0.0232	**0.0586**	0.0918	17.3539	8.2203
36	0.7243	**0.6877**	0.6927	**0.6877**	0.0770	**0.0694**	0.0696	**0.0694**	**0.0810**	0.2132	15.1890	3.0904
37	**0.8880**	0.9034	0.9006	0.9034	0.0515	**0.0509**	0.0509	**0.0509**	**0.1202**	0.1646	15.3972	4.3526
38	1.1245	1.0910	**1.0910**	1.0910	0.0303	**0.0298**	0.0298	**0.0298**	**0.1838**	0.2798	16.1686	5.9278
39	**0.8684**	0.8764	0.8764	0.8764	0.0244	**0.0243**	**0.0243**	**0.0243**	0.2712	0.1295	17.0252	8.3602
40	0.8447	0.8119	**0.8118**	0.8119	**0.0250**	0.0258	0.0258	0.0258	0.3673	**0.1424**	17.6690	10.3599
41	0.8159	0.7885	**0.7667**	**0.7667**	0.1247	0.1259	**0.1234**	**0.1234**	**0.0815**	0.1231	14.7031	3.4379
42	**0.9458**	0.9669	0.9668	0.9669	**0.0682**	0.0713	0.0712	0.0713	0.1289	**0.1181**	15.3369	4.8556
43	1.0217	**0.9456**	**0.9456**	**0.9456**	**0.0376**	0.0389	0.0389	0.0389	0.1995	**0.1376**	15.7598	6.4588
44	**1.0712**	1.0759	1.0759	1.0759	0.0261	**0.0261**	0.0261	0.0261	0.3085	**0.1456**	16.7421	8.9836
45	0.8651	**0.8328**	**0.8328**	**0.8328**	**0.0237**	0.0237	0.0237	0.0237	0.5005	**0.1736**	18.4610	12.3467
46	**0.5925**	0.6452	0.6491	0.6452	**0.0487**	0.0510	0.0509	0.0510	**0.4204**	2.7928	16.9153	5.2404
47	0.7928	**0.7753**	0.7753	**0.7753**	0.0282	**0.0264**	0.0264	**0.0264**	**0.7459**	2.8224	17.0613	6.1811
48	0.7753	**0.7632**	0.7636	**0.7632**	0.0223	**0.0219**	0.0220	**0.0219**	**0.9804**	2.9008	17.7563	7.5760
49	**0.8234**	0.8301	0.8435	0.8508	**0.0088**	0.0089	0.0091	0.0093	**1.4556**	5.8274	21.1909	11.8035
50	0.6961	**0.6925**	0.6925	**0.6925**	0.0172	0.0155	**0.0155**	0.0155	**1.8296**	2.8851	19.4386	10.7617
51	0.8559	0.8635	**0.8297**	0.8297	**0.1124**	0.1550	0.1522	0.1524	**0.7754**	5.3334	19.7377	8.4114
52	0.7515	0.6922	**0.6921**	0.6922	**0.0848**	0.0903	0.0892	0.0892	**1.1850**	5.3997	20.4408	9.6556
53	0.8235	0.7922	**0.7922**	0.7922	0.0388	**0.0379**	**0.0379**	**0.0379**	**1.6191**	5.4256	20.8525	11.3704
54	0.6813	0.6765	**0.6764**	0.6765	**0.0338**	0.0341	0.0341	0.0341	**2.1862**	5.4775	22.1008	12.8782
55	0.8538	**0.8308**	**0.8308**	**0.8308**	**0.0144**	0.0148	0.0148	0.0148	**3.3929**	5.7765	22.7618	15.9638
56	0.8049	**0.7689**	0.7824	0.7821	**0.1013**	0.1122	0.1116	0.1115	**1.1292**	8.5075	23.0410	12.5976
57	0.6317	0.6023	**0.6018**	0.6023	**0.0835**	0.0993	0.0996	0.0996	**1.7694**	8.4910	23.5395	13.0080
58	0.6884	**0.6690**	**0.6690**	**0.6690**	0.0558	**0.0557**	**0.0557**	**0.0557**	**2.4266**	8.4105	24.5284	15.3027
59	0.9664	**0.8668**	0.8668	**0.8668**	0.0232	**0.0203**	0.0203	**0.0203**	**4.6183**	8.5497	25.4301	18.2997
60	0.7862	**0.7681**	**0.7681**	**0.7681**	**0.0228**	0.0238	0.0238	0.0238	**4.2915**	8.5050	26.9437	20.8700
Avg	0.8236	0.8103	0.8096	**0.8096**	**0.0440**	0.0464	0.0462	0.0462	**1.0489**	2.9799	18.7085	9.1311
N. best	9	13	15	13	16	12	5	12	23	7	0	0
p			0.0138				0.8082				5.28E-18	
Mean rank	3.0667	2.3833	2.2	2.35	2.3667	2.6333	2.5333	2.4667	1.2333	1.7667	4	3
Rank position	2	1, 2	1	1, 2	1	1	1	1	1	1	3	2

Table 9: Comparison of the results achieved by SGS-ES, CH-J, NSGA-III and MOEA/D on the VLS instances based on the Hypervolume, Purity, and D_R metrics.

Instance	**Hypervolume**				**Purity**				D_R			
	SGS-ES	**CH-J**	**NSGA-III**	**MOEA/D**	**SGS-ES**	**CH-J**	**NSGA-III**	**MOEA/D**	**SGS-ES**	**CH-J**	**NSGA-III**	**MOEA/D**
61	**0.8197**	0.8047	0.8047	0.8047	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0118	0.0117	0.0118
62	**0.8295**	0.8172	0.8172	0.8172	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0097	0.0096	0.0097
63	**0.7660**	0.7498	0.7498	0.7498	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0113	0.0113	0.0113
64	**0.7862**	0.7695	0.7695	0.7695	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0110	0.0110	0.0110
65	**0.7245**	0.7015	0.7015	0.7015	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0171	0.0171	0.0171
66	**0.8015**	0.7835	0.7835	0.7835	**1.0000**	0.0058	0.0058	0.0058	**0.0000**	0.0153	0.0153	0.0153
67	**0.7581**	0.7414	0.7416	0.7415	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0105	0.0106	0.0105
68	**0.7390**	0.7136	0.7138	0.7137	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0176	0.0177	0.0177
69	**0.7368**	0.7177	0.7177	0.7177	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0131	0.0131	0.0131
70	**0.7525**	0.7282	0.7282	0.7282	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0172	0.0173	0.0172
71	**0.8098**	0.7951	0.7952	0.7951	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0106	0.0107	0.0107
72	**0.8734**	0.8629	0.8630	0.8630	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0075	0.0075	0.0075
73	**0.7979**	0.7810	0.7812	0.7811	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0111	0.0112	0.0112
74	**0.8509**	0.8415	0.8416	0.8416	**0.9964**	0.0038	0.0038	0.0038	**0.0000**	0.0067	0.0067	0.0067
75	**0.7879**	0.7710	0.7712	0.7711	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0114	0.0115	0.0115
76	**0.8505**	0.8394	0.8396	0.8395	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0072	0.0073	0.0072
77	**0.7478**	0.7223	0.7223	0.7223	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0176	0.0176	0.0176
78	**0.7890**	0.7716	0.7716	0.7716	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0136	0.0136	0.0136
79	**0.7465**	0.7253	0.7254	0.7253	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0141	0.0142	0.0141
80	**0.8157**	0.8011	0.8012	0.8011	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0096	0.0097	0.0096
81	**0.8205**	0.8066	0.8069	0.8069	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0090	0.0089	0.0089
82	**0.8650**	0.8562	0.8563	0.8563	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0084	0.0084	0.0084
83	**0.8200**	0.8034	0.8036	0.8036	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0111	0.0109	0.0109
84	**0.8292**	0.8169	0.8169	0.8169	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0090	0.0089	0.0089
85	**0.8239**	0.8096	0.8097	0.8096	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0102	0.0102	0.0102
86	**0.8317**	0.8202	0.8202	0.8202	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0082	0.0082	0.0082
87	**0.8090**	0.7974	0.7974	0.7974	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0081	0.0081	0.0081
88	**0.8372**	0.8251	0.8251	0.8251	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0089	0.0089	0.0089
89	**0.7629**	0.7428	0.7429	0.7428	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0147	0.0146	0.0147
90	**0.8093**	0.7948	0.7949	0.7949	**1.0000**	0.0000	0.0000	0.0000	**0.0000**	0.0102	0.0101	0.0102
Avg	**0.7997**	0.7837	0.7838	0.7838	**0.9999**	0.0003	0.0003	0.0003	**0.0000**	0.0114	0.0114	0.0114
N. best	30	0	0	0	30	0	0	0	30	0	0	0
p			1.66E-17				2.19E-19				2.39E-15	
Mean rank	4	1.5323	2.4335	2.0323	4	2	2	2	1	2.95	3.05	3
Rank position	1	3	2	2, 3	1	2	2	2	1	3	3	3

Table 10: Comparison of the results achieved by SGS-ES, CH-J, NSGA-III and MOEA/D on the MLS instances based on the Spread, Spacing, and CPU time (s) metrics.

Instance	**Spread**				**Spacing**				**CPU**			
	SGS-ES	**CH-J**	**NSGA-III**	**MOEA/D**	**SGS-ES**	**CH-J**	**NSGA-III**	**MOEA/D**	**SGS-ES**	**CH-J**	**NSGA-III**	**MOEA/D**
61	1.0602	1.0473	**1.0459**	1.0471	0.0077	0.0073	0.0073	**0.0073**	**21.7450**	50.1929	58.8327	66.7168
62	**1.0119**	1.0496	1.0464	1.0496	0.0055	**0.0053**	0.0054	0.0053	**62.3640**	472.9695	327.7931	488.7253
63	0.8898	0.8261	0.8256	**0.8255**	**0.0055**	0.0056	0.0056	0.0056	**25.8207**	38.8928	51.8122	58.2162
64	0.9226	**0.8744**	0.9106	0.8880	0.0051	**0.0043**	0.0059	0.0051	**57.0123**	457.8403	312.8328	476.2181
65	**0.8744**	0.9017	0.9005	0.9017	0.0059	**0.0056**	0.0056	**0.0056**	**30.4711**	46.5792	61.1699	67.6016
66	**0.9823**	1.0137	1.0132	1.0137	0.0056	**0.0053**	0.0053	**0.0053**	**75.7831**	294.5378	215.1681	314.6727
67	0.8073	**0.7758**	0.7919	0.7892	**0.0053**	0.0054	0.0062	0.0062	**25.9146**	61.7673	73.8247	86.6843
68	0.8607	**0.8158**	0.8333	0.8314	0.0039	**0.0037**	0.0041	0.0043	**84.8752**	321.0555	244.1829	344.4683
69	0.8101	0.7642	**0.7625**	0.7626	**0.0054**	0.0056	0.0055	0.0055	**25.9769**	69.9960	81.8046	99.3700
70	0.8576	**0.8543**	0.8780	0.8794	0.0043	**0.0040**	0.0049	0.0051	**90.3612**	333.5820	254.6311	362.0777
71	0.9612	**0.9179**	0.9734	0.9788	0.0076	**0.0071**	0.0097	0.0101	**25.2129**	77.2715	77.8492	94.0222
72	1.0414	1.0363	**1.0323**	1.0335	0.0084	0.0084	0.0084	**0.0084**	**65.3719**	129.7969	108.7024	147.3442
73	0.9906	**0.9181**	0.9536	0.9519	**0.0068**	0.0072	0.0083	0.0084	**30.7839**	55.2623	64.4688	75.9908
74	1.0319	**1.0087**	1.0087	1.0087	0.0079	**0.0078**	0.0078	0.0078	**77.8728**	186.2354	145.0289	206.7596
75	0.9003	**0.8709**	0.8948	0.8977	0.0064	**0.0061**	0.0070	0.0072	**34.7806**	53.0671	65.9283	77.1399
76	**0.9747**	0.9768	0.9945	0.9922	**0.0074**	0.0075	0.0081	0.0081	**92.2981**	100.0133	94.9817	124.1663
77	0.8656	0.8515	**0.8510**	0.8511	0.0057	0.0055	0.0055	**0.0055**	**37.2449**	79.5971	86.5105	105.8310
78	0.9621	**0.9338**	0.9338	**0.9338**	0.0052	**0.0047**	0.0047	**0.0047**	**107.1587**	378.6946	267.8694	405.3375
79	0.8274	**0.7902**	0.8018	0.7966	0.0053	**0.0051**	0.0058	0.0055	**45.8857**	52.1700	71.6422	83.8278
80	0.9324	**0.9018**	0.9218	0.9127	**0.0054**	0.0055	0.0062	0.0059	**101.0368**	228.2797	187.8734	261.1875
81	0.9468	**0.9066**	0.9523	0.9528	0.0096	**0.0084**	0.0115	0.0116	**37.5365**	54.1488	60.3279	75.1196
82	**1.1053**	1.1475	1.1439	1.1441	0.0081	0.0077	**0.0077**	0.0077	**92.0232**	369.0146	254.4615	389.2578
83	1.0663	1.0155	**1.0145**	1.0146	**0.0077**	0.0078	0.0079	0.0079	**42.3040**	74.4537	77.4552	98.9680
84	0.9831	**0.9649**	0.9666	0.9669	0.0061	**0.0058**	0.0058	0.0058	**113.6161**	351.1298	248.0501	375.9355
85	1.0316	**1.0017**	1.0243	1.0186	0.0071	**0.0069**	0.0079	0.0077	**49.3493**	62.4446	72.6152	89.6216
86	**0.9742**	0.9830	0.9828	0.9830	0.0058	**0.0055**	0.0055	0.0055	**129.1932**	180.4264	145.6146	207.6757
87	0.9481	**0.9275**	0.9276	**0.9275**	0.0072	0.0068	**0.0068**	0.0068	**52.3694**	126.2987	119.3945	156.4167
88	1.0088	1.0092	**1.0047**	1.0055	0.0061	0.0059	0.0059	**0.0059**	**149.7154**	300.2133	229.4651	330.7565
89	0.9555	0.9294	**0.9275**	0.9293	0.0070	0.0066	0.0066	**0.0066**	72.6986	**46.6116**	68.8165	83.1229
90	1.0377	1.0098	**1.0089**	1.0098	**0.0052**	0.0054	0.0054	0.0054	**161.0501**	285.6970	220.1420	323.1561
Avg	0.9541	**0.9341**	0.9442	0.9432	0.0063	**0.0061**	0.0066	0.0066	**67.2609**	177.9413	144.9750	202.5463
N. best	6	15	8	3	8	15	2	8	29	1	0	0
p			0.0056				0.0049				1.02E-16	
Mean rank	3.1667	2.1667	2.1667	2.5	2.6333	1.8667	2.75	2.75	1.0667	2.5	2.4333	4
Rank position	3	2	1	1	2	1	2	2	1	2	2	3

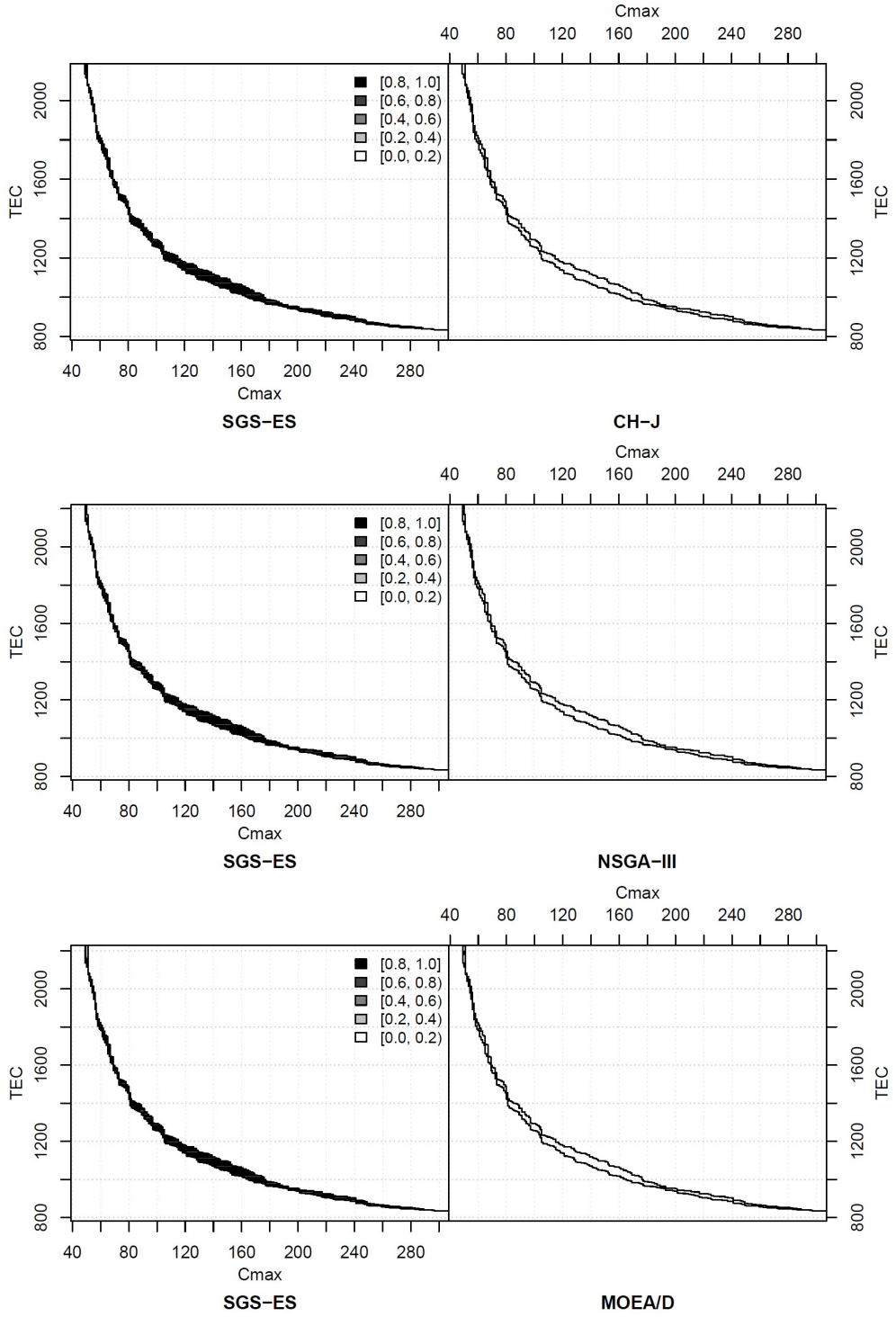

Figure 14: The Diff-EAF plots that compare SGS-ES with CH-J, NSGA-III and MOEA/D for instance 49.

4.7 Comparing the mathematical models

This section compares the two mathematical models introduced in Chapter 2 from an experimental standpoint, so as to evaluate the impact of the reduction of the solution space introduced by Formulation 2 with respect to Formulation 1. In order to achieve this, the MILP solving step of the exact algorithm was first carried out with Formulation 1, and then with Formulation 2.

Table 11 reports the results of the comparison for the first 60 instances. The exact algorithm was not able to solve the last 30 instances with Formulation 1 due to the RAM constraints of the machine used for the experimental evaluations. For each instance, the value under the column $\Delta_{F1,F2}$ is a measure of the improvement introduced by Formulation 2 in the computational times. Formally, let us denote the time taken for the exact algorithm with Formulation 1 and Formulation 2 to solve instance i as $\delta_{\mathrm{F1},i}$ and $\delta_{\mathrm{F2},i}$, respectively. Then, the value of $\Delta_{F1,F2}$ in row i is equal to $1 - \delta_{\mathrm{F2},i}/\delta_{\mathrm{F1},i}$.

Formulation 2 outperforms Formulation 1 in all the 60 instances, except for instance 4. In particular, the average computational time of Formulation 2 on instances 31–60 is remarkably two order of magnitudes lower than the average time achieved by Formulation 1 on the same instances. Overall, Formulation 2 achieved a 96.68% average improvement on such instances.

4.8 Evaluating the impact of SGH as the initial feasible solution heuristic for the exact algorithm

The purpose of this section is to evaluate the increase in computational efficiency enabled by the use of SGH as a means to provide an initial feasible solution for the mathematical programming step of the exact algorithm.

Table 4.8 shows the performances achieved by the exact algorithm that uses Formulation 2, with (F2-ws) or without (F2) the initial solution provided by SGH. For each instance, the time limit for computations was set to 7200 seconds. The values under the column $\Delta_{F2,F2ws}$ are defined similarly to the values of $\Delta_{F1,F2}$ in Section 4.7. The computational advantage enabled by the initial solution provided by SGH is noteworthy, especially for instances 31–60, where it amounts to a 43.13% average improvement. Instances 20 and 24 are the only instances where the exact algorithm without the SGH initial solution achieves superior performances. For such instances, SGH is revealed as a burden for the

already fast mathematical programming step of the exact algorithm. Finally, it is interesting to observe that the initial solution provided by SGH is still insufficient to enable the completion of the computations on both instance 62 and 83 within the time limit.

Instance	**CPU**			**Instance**	**CPU**		
	F1	**F2**	$\Delta_{F1,F2}$		**F1**	**F2**	$\Delta_{F1,F2}$
1	0.2853	**0.2236**	21.63	31	7.2461	**0.9331**	87.12
2	0.6087	**0.4214**	30.77	32	21.5001	**1.1392**	94.70
3	0.3469	**0.2920**	15.82	33	67.4728	**1.5184**	97.75
4	**0.3941**	0.5131	-30.21	34	175.1797	**2.2396**	98.72
5	0.4617	**0.4139**	10.36	35	264.9398	**2.5727**	99.03
6	0.7080	**0.4404**	37.80	36	10.4937	**0.8970**	91.45
7	0.7839	**0.5416**	30.91	37	29.6982	**1.3301**	95.52
8	0.9900	**0.5786**	41.55	38	130.3063	**2.5860**	98.02
9	0.5386	**0.2888**	46.39	39	219.4452	**2.8135**	98.72
10	0.9039	**0.7345**	18.74	40	417.0358	**4.1034**	99.02
11	0.6921	**0.4311**	37.71	41	16.1518	**0.9381**	94.19
12	1.2225	**0.8308**	32.04	42	37.9442	**1.7523**	95.38
13	0.8033	**0.4120**	48.71	43	124.6872	**2.6392**	97.88
14	1.2348	**0.5765**	53.32	44	334.8600	**3.8372**	98.85
15	0.7733	**0.2753**	64.40	45	652.2999	**5.7957**	99.11
16	1.7666	**0.7318**	58.58	46	45.5896	**3.4578**	92.42
17	1.2358	**0.5949**	51.86	47	98.2345	**4.2304**	95.69
18	2.4041	**1.0459**	56.49	48	296.7794	**7.3264**	97.53
19	1.2549	**0.5384**	57.10	49	1247.3653	**17.4488**	98.60
20	1.8719	**0.6493**	65.31	50	1298.3045	**11.4346**	99.12
21	2.1630	**0.6916**	68.02	51	134.1745	**7.1962**	94.64
22	2.8599	**0.9937**	65.25	52	102.4868	**5.2770**	94.85
23	1.6491	**0.4706**	71.46	53	512.3011	**10.1268**	98.02
24	4.1166	**1.2622**	69.34	54	824.7366	**12.4966**	98.48
25	1.2025	**0.3022**	74.87	55	2242.6907	**17.9711**	99.20
26	3.1990	**0.8453**	73.58	56	94.1080	**4.1502**	95.59
27	1.5952	**0.4340**	72.79	57	232.9828	**11.0508**	95.26
28	4.6481	**1.1727**	74.77	58	547.1832	**12.9963**	97.62
29	3.7137	**0.9684**	73.92	59	1729.1312	**21.2720**	98.77
30	4.2313	**0.9548**	77.43	60	2359.3696	**20.5363**	99.13
Avg	1.6220	**0.6210**	49.02	**Avg**	475.8233	**6.7356**	96.68
N. best	1	**29**		**N. best**	0	**30**	

Table 11: Comparison of the first mathematical model with the second one, as a part of the mathematical programming step in the exact algorithm.

Table 12: Comparison of the ad-hoc initial solution provided by SGH with the initial solution heuristics used by CPLEX for the second mathematical model, as part of the mathematical programming step of the exact algorithm.

Instance	**CPU**			**Instance**	**CPU**			**Instance**	**CPU**		
	F2-ws	**F2**	$\Delta_{F2,F2ws}$		**F2-ws**	**F2**	$\Delta_{F2,F2ws}$		**F2-ws**	**F2**	$\Delta_{F2,F2ws}$
1	**0.1008**	0.2236	54.93	31	**0.8512**	0.9331	8.78	61	**666.9969**	824.1859	19.07
2	**0.1153**	0.4214	72.63	32	**0.9265**	1.1392	18.68	62	**7200.0000**	**7200.0000**	0.00
3	**0.0727**	0.2920	75.12	33	**1.1595**	1.5184	23.64	63	**1108.9581**	1255.9919	11.71
4	**0.0871**	0.5131	83.03	34	**1.8419**	2.2396	17.76	64	**1659.7399**	1936.1128	14.27
5	**0.0633**	0.4139	84.69	35	**1.4991**	2.5727	41.73	65	**768.7349**	905.3403	15.09
6	**0.0863**	0.4404	80.39	36	**0.5590**	0.8970	37.68	66	**1734.9434**	2009.4580	13.66
7	**0.4766**	0.5416	12.00	37	**0.9004**	1.3301	32.31	67	**1087.8168**	1354.7480	19.70
8	**0.4386**	0.5786	24.21	38	**2.1188**	2.5860	18.07	68	**2049.5867**	2338.4407	12.35
9	**0.2030**	0.2888	29.71	39	**1.8326**	2.8135	34.86	69	**1530.0836**	1905.9716	19.72
10	**0.2011**	0.7345	72.62	40	**3.1554**	4.1034	23.10	70	**2169.0729**	2469.0323	12.15
11	**0.1715**	0.4311	60.22	41	**0.3282**	0.9381	65.01	71	**852.8703**	983.5572	13.29
12	**0.3166**	0.8308	61.89	42	**0.6280**	1.7523	64.16	72	**722.9097**	884.6691	18.28
13	**0.2909**	0.4120	29.40	43	**1.9229**	2.6392	27.14	73	**989.8646**	1147.7326	13.75
14	**0.4330**	0.5765	24.88	44	**2.8586**	3.8372	25.50	74	**1049.9244**	1268.2838	17.22
15	**0.2621**	0.2753	4.78	45	**4.0976**	5.7957	29.30	75	**1026.3015**	1202.6681	14.66
16	**0.3315**	0.7318	54.70	46	**0.9616**	3.4578	72.19	76	**1079.7493**	1294.5702	16.59
17	**0.3683**	0.5949	38.09	47	**2.8655**	4.2304	32.26	77	**1360.9421**	1509.7140	9.85
18	**0.6615**	1.0459	36.76	48	**5.2100**	7.3264	28.89	78	**2367.6941**	2675.9165	11.52
19	**0.2933**	0.5384	45.53	49	**13.6404**	17.4488	21.83	79	**1569.5841**	1787.6726	12.20
20	0.6525	**0.6493**	-0.50	50	**8.5745**	11.4346	25.01	80	**1988.7290**	2338.7394	14.97
21	**0.6331**	0.6916	8.46	51	**0.7239**	7.1962	89.94	81	**655.7005**	776.1643	15.52
22	**0.9480**	0.9937	4.60	52	**1.4469**	5.2770	72.58	82	**1407.1212**	2595.9413	45.80
23	**0.2800**	0.4706	40.50	53	**3.9724**	10.1268	60.77	83	**7200.0000**	**7200.0000**	0.00
24	1.2983	**1.2622**	-2.85	54	**4.7098**	12.4966	62.31	84	**1730.3946**	2075.4408	16.63
25	**0.2478**	0.3022	17.99	55	**11.6508**	17.9711	35.17	85	**1121.9333**	1344.9506	16.58
26	**0.4783**	0.8453	43.42	56	**1.4056**	4.1502	66.13	86	**1635.3323**	1953.5618	16.29
27	**0.2995**	0.4340	30.99	57	**1.6399**	11.0508	85.16	87	**1100.0952**	1294.0546	14.99
28	**0.8081**	1.1727	31.09	58	**2.7634**	12.9963	78.74	88	**1804.7383**	2141.9116	15.74
29	**0.6944**	0.9684	28.29	59	**14.8052**	21.2720	30.40	89	**1325.5667**	1553.8070	14.69
30	**0.8524**	0.9548	10.72	60	**7.2250**	20.5363	64.82	90	**2634.9724**	3066.1234	14.06
Avg	0.4055	0.6210	38.61	**Avg**	3.5425	6.7356	43.13	**Avg**	**1786.6786**	2043.1587	15.01
N. best	28	2		**N. best**	30	0		**N. best**	30	2	

4.9 Comparing the exact algorithm with SGS-ES

This section concludes the presentation of the numerical results. The purpose of this section is to compare the most computationally efficient exact solution approach with the whole heuristic schema presented in this thesis to evaluate the trade-off between solutions quality and required computational times. This section achieves so by comparing the performances of the exact algorithm that exploits Formulation 2 for the mathematical programming step with SGH as a heuristic for the initial feasible solution, and SGS-ES. The optimality of the solutions obtained by the exact algorithm allows us to disregard the distribution metrics. Indeed, the purpose of the comparison is to evaluate the quality of the Pareto front computed by SGS-ES with respect to the reference optimal Pareto front computed by the exact algorithm, regardless of its structure, i.e., the uniformity of the distribution of the Pareto-optimal solutions.

Table 13 shows the results of the comparisons. For each instance, the time limit for computations was again set to 7200 seconds. The values under the column $\Delta_{F2ws,SE}$ are defined similarly to the values of $\Delta_{F1,F2}$ in Section 4.7. Formally, let $\delta_{SE,i}$ and $\delta_{F2ws,i}$ be the time taken for SGS-ES and the exact algorithm, with Formulation 2 for the MILP step and SGH as the initial feasible solution heuristic, to solve instance i, respectively. Then, the value of $\Delta_{F2ws,SE}$ for instance i is $1 - \delta_{SE,i}/\delta_{F2ws,i}$. SGS-ES achieves significantly better computational times on the whole benchmark. In particular, the values under the column $\Delta_{F2ws,SE}$ suggest an average improvement of 73.56% for the MLS instances, which is as high as 96.08% and 96.55% for the VLS and the small-scale instances, respectively.

Hypervolume is the only quality metric reported in the table, as it is sufficient to highlight the difference in solutions quality attained by the two algorithms. Although the exact algorithm considerably outperforms SGS-ES on the MLS and the VLS instances, except for instances 62 and 83, SGS-ES succeeds in computing the optimal solutions for ten of the small-scale instances (1, 11, 13, 20, 25, 26, 27, 28, 29, and 30). The performances of SGS-ES may be improved by considering a larger set of EPS moves, at the expense of a higher computational burden.

Table 13: Comparison of the results achieved by the (warmstarted) exact algorithm with Formulation 2 and SGS-ES on each benchmark instance based on the Hypervolume and the CPU time (s) metrics.

Instance	Hypervolume		CPU			Instance	Hypervolume		CPU			Instance	Hypervolume		CPU		
	F2-ws	**SGS-ES**	**F2-ws**	**SGS-ES**	$\Delta_{F2ws,SE}$		**F2-ws**	**SGS-ES**	**F2-ws**	**SGS-ES**	$\Delta_{F2ws,SE}$		**F2-ws**	**SGS-ES**	**F2-ws**	**SGS-ES**	$\Delta_{F2ws,SE}$
1	**0.7460**	**0.7460**	0.4766	**0.0059**	98.77	31	**0.8829**	0.8813	0.8512	**0.0384**	95.48	61	**0.8223**	0.8162	666.9969	**18.6766**	97.20
2	**0.7756**	0.7745	0.4386	**0.0098**	97.76	32	**0.8122**	0.8109	0.9265	**0.0551**	94.06	62	0.2722	**0.8262**	7200.0000	**53.8304**	99.25
3	**0.7603**	0.7408	0.2030	**0.0046**	97.72	33	**0.7783**	0.7769	1.1595	**0.0767**	93.38	63	**0.7704**	0.7645	1108.9581	**22.1941**	98.00
4	**0.7566**	0.7563	0.2011	**0.0137**	93.18	34	**0.7021**	0.6916	1.8419	**0.0946**	94.86	64	**0.7886**	0.7827	1659.7399	**49.0354**	97.05
5	**0.8019**	0.8017	0.1715	**0.0077**	95.52	35	**0.6012**	0.6011	1.4991	**0.0483**	96.78	65	**0.7298**	0.7205	768.7349	**25.6876**	96.66
6	**0.8247**	0.8234	0.3166	**0.0148**	95.34	36	**0.8563**	0.8541	0.5590	**0.0641**	88.54	66	**0.8064**	0.7984	1734.9434	**65.3891**	96.23
7	**0.7259**	0.7256	0.2909	**0.0031**	98.92	37	**0.8731**	0.8710	0.9004	**0.0914**	89.85	67	**0.7590**	0.7549	1087.8168	**22.0816**	97.97
8	**0.8153**	0.8137	0.4330	**0.0104**	97.59	38	**0.8473**	0.8407	2.1188	**0.1351**	93.63	68	**0.7446**	0.7372	2049.5867	**72.8456**	96.45
9	**0.7687**	0.7559	0.2621	**0.0054**	97.94	39	**0.7804**	0.7771	1.8326	**0.2147**	88.29	69	**0.7418**	0.7356	1530.0836	**22.0728**	98.56
10	**0.7732**	0.7722	0.3315	**0.0153**	95.39	40	**0.7823**	0.7730	3.1554	**0.2913**	90.77	70	**0.7563**	0.7450	2169.0729	**77.6561**	96.42
11	**0.8077**	**0.8077**	0.3683	**0.0079**	97.85	41	**0.9294**	0.9282	0.3282	**0.0651**	80.17	71	**0.8055**	0.7983	852.8703	**21.7973**	97.44
12	**0.8506**	0.8484	0.6615	**0.0193**	97.09	42	**0.8535**	0.8529	0.6280	**0.1035**	83.52	72	**0.8763**	0.8704	722.9097	**56.3769**	92.20
13	**0.6900**	**0.6900**	0.2933	**0.0038**	98.72	43	**0.8630**	0.8614	1.9229	**0.1566**	91.86	73	**0.7987**	0.7918	989.8646	**26.4831**	97.32
14	**0.7548**	0.7533	0.6525	**0.0104**	98.40	44	**0.8279**	0.8223	2.8586	**0.2399**	91.61	74	**0.8522**	0.8471	1049.9244	**67.1380**	93.61
15	**0.7520**	0.7509	0.6331	**0.0060**	99.06	45	**0.7936**	0.7899	4.0976	**0.4059**	90.09	75	**0.7890**	0.7835	1026.3015	**30.1399**	97.06
16	**0.7993**	0.7987	0.9480	**0.0196**	97.94	46	**0.8266**	0.8162	0.9616	**0.3468**	63.94	76	**0.8550**	0.8488	1079.7493	**79.8189**	92.61
17	**0.7086**	0.7082	0.2800	**0.0082**	97.07	47	**0.8642**	0.8612	2.8655	**0.6161**	78.50	77	**0.7514**	0.7442	1360.9421	**31.9174**	97.65
18	**0.8156**	0.8106	1.2983	**0.0185**	98.58	48	**0.8760**	0.8725	5.2100	**0.8131**	84.39	78	**0.7935**	0.7866	2367.6941	**92.9936**	96.07
19	**0.5794**	0.5785	0.2478	**0.0042**	98.29	49	**0.8064**	0.8020	13.6404	**1.1943**	91.24	79	**0.7519**	0.7451	1569.5841	**40.2177**	97.44
20	**0.7294**	**0.7294**	0.4783	**0.0084**	98.24	50	**0.8324**	0.8162	8.5745	**1.5257**	82.21	80	**0.8171**	0.8118	1988.7290	**86.7287**	95.64
21	**0.7458**	0.7455	0.2995	**0.0070**	97.66	51	**0.8557**	0.8451	0.7239	**0.6305**	12.90	81	**0.8230**	0.8152	655.7005	**32.5687**	95.03
22	**0.7970**	0.7968	0.8081	**0.0143**	98.23	52	**0.8681**	0.8656	1.4469	**0.9671**	33.16	82	**0.8684**	0.8616	1407.1212	**79.0559**	94.38
23	**0.7743**	0.7703	0.6944	**0.0075**	98.93	53	**0.8903**	0.8875	3.9724	**1.3249**	66.65	83	0.2324	**0.8159**	7200.0000	**36.6585**	99.49
24	**0.8413**	0.8403	0.8524	**0.0217**	97.45	54	**0.8848**	0.8836	4.7098	**1.8315**	61.11	84	**0.8314**	0.8261	1730.3946	**97.6430**	94.36
25	**0.6976**	**0.6976**	0.1008	**0.0024**	97.59	55	**0.8454**	0.8445	11.6508	**2.8734**	75.34	85	**0.8251**	0.8201	1121.9333	**42.2191**	96.24
26	**0.7683**	**0.7683**	0.1153	**0.0059**	94.86	56	**0.8899**	0.8868	1.4056	**0.9294**	33.88	86	**0.8337**	0.8276	1635.3323	**111.6060**	93.18
27	**0.7425**	**0.7425**	0.0727	**0.0037**	94.88	57	**0.7517**	0.7432	1.6399	**1.4716**	10.26	87	**0.8107**	0.8041	1100.0952	**45.2810**	95.88
28	**0.7438**	**0.7438**	0.0871	**0.0091**	89.56	58	**0.9060**	0.9057	2.7634	**2.0238**	26.76	88	**0.8402**	0.8351	1804.7383	**128.1692**	92.90
29	**0.4531**	**0.4531**	0.0633	**0.0054**	91.42	59	**0.8994**	0.8950	14.8052	**3.8813**	73.78	89	**0.7685**	0.7590	1325.5667	**63.2728**	95.23
30	**0.7532**	**0.7532**	0.0863	**0.0116**	86.58	60	**0.8198**	0.8169	7.2250	**3.6342**	49.70	90	**0.8131**	0.8068	2634.9724	**136.8648**	94.81
Avg	0.7517	0.7499	0.4055	0.0095	96.55	**Avg**	0.8333	0.8292	3.5425	0.8715	73.56	**Avg**	0.7643	0.7960	1786.7791	57.8807	96.08
N. best	30	10	0	90		**N. best**	30	0	0	90		**N. best**	28	2	0	90	

Chapter 5

Conclusions

The achievement of energy efficiency in manufacturing production has become a compelling matter in the latest years, due to the pressing environmental issues and the consequent desire to shift towards a sustainable industry model. The pricing scheme induced by the TOU-based tariffs policy indeed allows to drive customers demand so as to relieve peak energy generation, and enabling financial benefits for the energy providers as well.

This thesis considered a representative, hard TOU scheduling problem, the BPMSTP. It provided two distinct mathematical models, as well as the two heuristics SGH and ES, based on several insights on the problem itself. The exact algorithm based on the models, and the whole heuristic scheme based on SGH and ES, achieved high performances on the wide test benchmark. The combinatorial properties of the BPMSTP, and the novel intuitions underlying the algorithms presented in this thesis, may also enable further research progress in problems that display a similar structure.

Among the immediate future developments of this thesis, it is important to discuss the scope for improvement of the exact algorithm, SGS, and SGS-ES. Currently, such algorithms tackle the BPMSTP by solving a sequence of distinct single-objective optimization problems that constrain the maximum makespan of the solution. Such problems are considered independently. In fact, the three algorithms do not exploit the information provided by the job assignments performed at the previous iterations, despite the makespan varies by little between them. Such a small variation intuitively suggests that neighboring solutions may share several job assignments. In the case of SGS and SGS-ES, such an intuition may lead to increase the computational efficiency. Instead, its application to the case of the exact algorithm may lead to a matheuristic where the previous assignments are treated as additional constraints in the mathematical program.

Furthermore, as specifically regards ES, the considered subset of EPS moves may be reduced by disregarding part of the EPS-I's only consisting of idle slots. As a matter of fact, many such EPS-I's are already implicitly disregarded by SGH while performing the assignments of the jobs. Such information could be exploited by ES so as to avoid considering EPS-I's that cannot lead to improvement. Another improvement of the heuristic scheme may rely on using the exact algorithm to improve the initial solution provided by SGH as much as possible within a strict time limit, before applying ES.

Finally, expanding the experimental benchmark by including instances with particular properties, such as non-increasing slot costs or distinct processing times, may be beneficial to gain further insight into the combinatorics of the BPMSTP. Furthermore, the BPMSTP itself could be extended to take into account different machines environments, such as uniform or unrelated machines. The jobs could be characterized by machine-dependent setup times, while the relationships between distinct jobs could be captured by introducing sequence-dependent setup times. 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 in modeling real manufacturing systems. It would be interesting to investigate how the existing mathematical formulations and heuristics for the BPMSTP may translate to new problems with the aforementioned characteristics.

In conclusion, the difficulty of the TOU scheduling problems, along with their practical nature, naturally drives research towards operational solutions for practitioners. Yet, several research efforts showed how observations on the solutions structure and properties often led to more efficient heuristics, in terms of both computational efficiency and solutions quality. Future research should address the difficulty of the problems by developing exact or fast heuristic algorithms resulting from a thorough investigation of the structure of the problem. Specifically, at the time of writing, while the literature in TOU scheduling offers several mathematical modeling efforts, it is lacking in matheuristics, and most of all exact algorithms that exploits mathematical programming.

Appendix A

Additional notation

This appendix presents a reference for some additional notation used in the thesis. In order to describe the considered scheduling problems, the thesis extensively uses the triplet notation $\alpha \,|\, \beta \,|\, \gamma$ [51], and adopts the naming conventions used by Pinedo [88]. For clarity, Table 14 reports it to the extent of the scope of the thesis. Moreover, Table 15 shows the acronyms used by the thesis for mathematical programming and algorithms.

Field	**Characteristic**	**Meaning**
α field	1	single machine
	Pm	m parallel identical machines
	Qm	m parallel machines with different speeds
	Rm	m unrelated parallel machines)
	Fm	flow shop on m machines
	Jm	job shop on m machines
	FFc	flexible flow shop on c machines
	FJc	flexible job shop on c machines
β field	r_j	release dates
	d_j	due dates
	w_j	weights
	$prmp$	preemption
	$batch(b)$	batch processing
	$prmu$	permutation
	nwt	no-wait)
	$rcrc$	recirculation
γ field	C_{max}	makespan
	L_{max}	maximum lateness
	T_{max}	maximum tardiness
	$\sum w_j C_j$	total weighted completion time
	$\sum w_j T_j$	total weighted tardiness
	$\sum w_j U_j$	weighted number of tardy jobs

Table 14: Three-field notation for classical scheduling problems.

Class	Algorithm	Acronym
Mathematical programming and approximation algorithms [64]	Linear Programming	LP
	Integer Programming	IP
	Mixed-Integer Programming	MIP
	Mixed-Integer Linear Programming	MILP
	Mixed-Integer Non-Linear Programming	MINLP
	Polynomial-Time Approximation Scheme	PTAS
Heuristics and metaheuristics [77]	Local Search	LS
	Iterated Local Search	ILS
	Genetic Algorithm	GA
	Evolution Algorithm	EA
	Multi-objective Evolutionary Algorithm based on Decomposition	MOEA/D
	Non-dominated Sorted Genetic Algorithm	NSGA
	Particle Swarm Optimization	PSO
	Tabu Search	TS
	Variable Neighborhood Search	VNS
	Ant Colony Optimization	ACO
	Single-Population Genetic Algorithm	SPGA
	Multi-Population Genetic Algorithm	MPGA
	Strength Pareto-archived Evolutionary Algorithm 2	SPEA

Table 15: Acronyms of some algorithms for discrete optimization.

Appendix B

Symbols

This appendix reports the main mathematical symbols used in this thesis as a reference. Table 16 reports the symbols related to the problem statement, and introduced in Chapter 2. Table 17 reports the symbols related to the algorithms described in the thesis, and introduced in Chapter 3.

Symbol	**Meaning**
$\mathcal{J} = \{1, 2, \ldots, N\}$	Set of jobs
p_j, $j \in \mathcal{J}$	Processing time of job j
$\mathcal{H} = \{1, 2, \ldots, M\}$	Set of machines
u_h, $h \in \mathcal{H}$	Energy consumption rate of machine h
$\mathcal{T} = \{1, 2, \ldots, K\}$	Set of time slots
c_t, $t \in \mathcal{T}$	Cost of slot t
$\mathcal{S}$	Schedule
$C_j(\mathcal{S})$, $j \in \mathcal{J}$	Completion time of job j in $\mathcal{S}$
$C^{\max}(\mathcal{S})$	Makespan of $\mathcal{S}$
$E(\mathcal{S})$	Total energy cost of $\mathcal{S}$
$\mathcal{I}$	Instance $(\mathcal{J}, \{p_j, j \in \mathcal{J}\}, \mathcal{H}, \{u_h, h \in \mathcal{H}\}, \mathcal{T}, \{c_t, t \in \mathcal{T}\})$ of the BPMSTP
$\mathcal{P}_{\mathcal{J}'}$	Distinct processing times of the jobs in $\mathcal{J}' \subseteq \mathcal{J}$
$\mathcal{J}_d$, $d \in \mathcal{P}_{\mathcal{J}}$	Set of jobs with processing time d
$b_{d,t}$, $d \in \mathcal{P}_{\mathcal{J}}$, $t = 1, 2, \ldots, K - d + 1$	Sum of the costs of the slots in $\{t, t+1, \ldots, t+d-1\}$

Table 16: Table of the main mathematical symbols introduced in Chapter 2, and related to the problem statement.

Symbol	Meaning
$(h, \mathcal{A})$	Location in slots $\mathcal{A} \subseteq \mathcal{T}$ on machine $h \in \mathcal{H}$ (for a job with processing time $\lvert\mathcal{A}\rvert$)
$\mathcal{B}$	Schedule block
$\mathcal{S}_h$	Schedule on machine $h \in \mathcal{H}$
$\mathcal{D}(\hat{K})$	Instance $(\mathcal{J}, \{p_j, j \in \mathcal{J}\}, \mathcal{H}, \{u_h, h \in \mathcal{H}\}, \hat{\mathcal{T}} = \{1, 2, \ldots, \hat{K}\} \subseteq \mathcal{T}, \{c_t, t \in \hat{\mathcal{T}}\})$ of the BPMSTP
Ω	Minimum between N and $\hat{K}$, $1 \leq \hat{K} \leq K$
$\mathcal{S}_\mathcal{E}$ $(\mathcal{S}_{\mathcal{E},h})$	EPS subschedule of $\mathcal{E}$ (on $h \in \mathcal{H}$ in $\mathcal{S}$)
$\mathcal{J}(\mathcal{S}_\mathcal{E})$	Jobs scheduled in the subschedule $\mathcal{S}_\mathcal{E}$
$p_{\max}$	Maximum processing time in $\mathcal{P}_\mathcal{J}$
$\mu_{t,h}$, $t \in \mathcal{T}$, $h \in \mathcal{H}$	Sum of the costs of the slots in $\{1, 2, \ldots, t\}$, multiplied by u_h
$\mathcal{L}_p^J(\mathcal{S})$, $\mathcal{L}_p^I(\mathcal{S})$	Sets of subschedules of EPS-J's and EPS-I's in $\mathcal{S}$ of cardinality $p \in \mathcal{P}_\mathcal{J}$, respectively
$\mathcal{M}_p^J(\mathcal{S}) : \mathcal{H} \times \mathcal{T} \to \mathcal{L}_p^J(\mathcal{S}) \cup \emptyset, \quad p \in \mathcal{P}_\mathcal{J}$	Function that associates a time slot t on a machine h with a subschedule $\mathcal{S}_\mathcal{E}$ associated to the EPS-J $\mathcal{E}$ on h, and such that the smallest slot of $\mathcal{E}$ is t.
$\mathcal{M}_p^I(\mathcal{S}) : \mathcal{H} \times \mathcal{T} \to \mathcal{L}_p^I(\mathcal{S}) \cup \emptyset, \quad p \in \mathcal{P}_\mathcal{J}$	Function that associates a time slot t on a machine h with a subschedule $\mathcal{S}_\mathcal{E}$ associated to the EPS-I $\mathcal{E}$ on h, and such that the smallest slot of $\mathcal{E}$ is t.
$Q_\mathcal{E}$	List of the slots in the EPS $\mathcal{E}$ in non-decreasing order of their costs
$\eta_{h,n}^{\text{lb}}(Q_\mathcal{E})$	Sum of the costs of the first n slots in $Q_\mathcal{E}$, multiplied by u_h
$\eta_h^{\text{ub}}(\mathcal{E})$	Sum of the costs of the slots in $\mathcal{E}$
$\alpha(\mathcal{S}_\mathcal{E})$	Number of assigned slots in the subschedule $\mathcal{S}_\mathcal{E}$
$\underline{K}(\mathcal{I})$	Lower bound $\max\left\{\lfloor \sum_{j\in\mathcal{J}} p_j/M \rfloor, \max_{j\in\mathcal{J}}\{p_j\}\right\}$ for the makespan in SGS and the exact algorithm

Table 17: Table of the main mathematical symbols introduced in Chapter 3, and used in the description of SGH, ES, SGS, and the exact algorithm.

Appendix C

Reference

This appendix provides a summary of the main definitions and notions introduced in the thesis, by means of Table 18 and Table 19.

Notion	**Meaning**
Adjacent slots	Two slots t and $t+1$, with $t \in \mathcal{T}$, $t \neq K$
Assigned location	Location that only contains the assigned slots of one job
Assigned-consecutive slots	Two slots t and $t+k$, $1 \leq t \leq t+2 \leq t+k \leq K$, on machine $h \in \mathcal{H}$, that are assigned slots of some job $j \in \mathcal{J}$, and such that $t+1, t+2, \ldots, t+k-1$ are assigned slots of a subset of jobs of $\mathcal{J} \setminus \{j\}$.
Exchange Search (ES)	The local search for the BPMSTP presented in this thesis
Exchangeable Period Sequence (EPS)	A subset $\mathcal{E} \subseteq \mathcal{T}$ of adjacent time slots on a machine $h \in \mathcal{H}$ such that if $h \in \mathcal{H}$ processes some job $j \in \mathcal{J}$ in a time slot $t \in \mathcal{E}$, then all the assigned slots of j are in $\mathcal{E}$
EPS move	An EPS swap of two EPS's $\mathcal{E}$ and $\mathcal{E}'$, followed by an EPS rearrangement of $\mathcal{E}$, and an EPS rearrangement of $\mathcal{E}'$
EPS rearrangement	A procedure that reassigns each job scheduled in an EPS $\mathcal{E}$ on machine $h \in \mathcal{H}$ to a subset of adjacent slots in $\mathcal{E}$ on h.
EPS subschedule	The single-machine schedule associated with an EPS
EPS swap	For two given EPS's $\mathcal{E}$ and $\mathcal{E}'$ on machines h and h', respectively, an algorithm that schedules, in $\mathcal{E}'$ on machine h', each job originally scheduled in $\mathcal{E}$ on machine h, and vice-versa, without changing the relative assignments of the jobs.

Table 18: A summary of the main definitions and notions introduced in the thesis.

Notion	**Meaning**
Free-consecutive slots	Two free slots t and $t+k$ in $\mathcal{T}$ on a machine $h \in \mathcal{T}$, such that $t+1, t+2, \ldots, t+k-1$ are assigned
Free location	Location that contains only free slots
Free (or idle) slot	A slot $t \in \mathcal{T}$ on a machine $h \in \mathcal{H}$, such that no job is assigned to t on h
Location	An ordered pair $(h, \mathcal{A})$, $h \in \mathcal{H}$, $\mathcal{A} \subseteq \mathcal{T}$
Schedule block	Single-machine schedule that involves only subsets of non-free slots delimited by two free slots
Split-Greedy Heuristic (SGH)	The constructive heuristic for the BPMSTP presented in this thesis
Split-Greedy Scheduler (SGS)	The algorithm presented in this thesis that iteratively exploits SGH to find a heuristic Pareto front for the BPMSTP
Split-Greedy Scheduler with Exchange Search (SGS-ES)	The algorithm presented in this thesis that iteratively exploits SGH and ES to find a heuristic Pareto front for the BPMSTP
Split-location	Location including at least two free-consecutive slots
Split-schedule	A schedule with at least a split-scheduled job
Split-scheduled job	A job that is assigned to a split-location

Table 19: A summary of the main definitions and notions introduced in the thesis (continued).

Bibliography

[1] Mohammad Mohsen Aghelinejad, Yassine Ouazene, and Alice Yalaoui. Machine and production scheduling under electricity time varying prices. In *2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 992–996. IEEE, 2016. ISBN 978-1-5090-3665-3.

[2] Mohammad Mohsen Aghelinejad, Yassine Ouazene, and Alice Yalaoui. Preemptive scheduling of a single machine with finite states to minimize energy costs. In Antonio Sforza and Claudio Sterle, editors, *Optimization and Decision Science: Methodologies and Applications*, volume 217, pages 591–599. Springer International Publishing, 2017. ISBN 978-3-319-67307-3 978-3-319-67308-0.

[3] Mohammad Mohsen Aghelinejad, Yassine Ouazene, and Alice Yalaoui. Production scheduling optimisation with machine state and time-dependent energy costs. *International Journal of Production Research*, 56(16):5558–5575, 2018. ISSN 0020-7543, 1366-588X.

[4] Mohammad Mohsen Aghelinejad, Yassine Ouazene, and Alice Yalaoui. Energy efficient scheduling problems under time-of-use tariffs with different energy consumption of the jobs. *IFAC-PapersOnLine*, 51(11):1053–1058, 2018. ISSN 24058963.

[5] Mohammad Mohsen Aghelinejad, Yassine Ouazene, and Alice Yalaoui. Single-machine scheduling problem with total energy consumption costs minimization. *IFAC-PapersOnLine*, 52(13):409–414, 2019. ISSN 24058963.

[6] Mohammad Mohsen Aghelinejad, Oussama Masmoudi, Yassine Ouazene, and Alice Yalaoui. Multi-state two-machine permutation flow shop scheduling optimisation with time-dependent energy costs. *IFAC-PapersOnLine*, 53(2):11156–11161, 2020. ISSN 24058963.

[7] Mohammad Mohsen Aghelinejad, Yassine Ouazene, and Alice Yalaoui. Energy-cost-aware flow-shop scheduling systems with state-dependent energy consumptions. *IOP Conference Series: Earth and Environmental Science*, 463:012163, 2020. ISSN 1755-1315.

[8] MohammadMohsen Aghelinejad, Yassine Ouazene, and Alice Yalaoui. Complexity analysis of energy-efficient single machine scheduling problems. *Operations Research Perspectives*, 6:100105, 2019. ISSN 22147160.

[9] Davide Anghinolfi, Massimo Paolucci, and Roberto Ronco. A bi-objective heuristic approach for green identical parallel machine scheduling. *European Journal of Operational Research*, 289:416–434, 2021.

[10] Sadiqi Assia, El Abbassi Ikram, El Barkany Abdellah, Darcherif Moumen, and El Biyaali Ahmed. Optimizing electricity costs during integrated scheduling of jobs and stochastic preventive maintenance under time-of-use electricity tariffs. 2019. Publisher: Production Engineering Committee of the Polish Academy of Sciences, Polish Association for Production Management.

[11] Seyed Amin Badri, Allahyar Daghbandan, Zahra Aghabeiginiyay Fatalaki, and Mohammad Mirzazadeh. Flow shop scheduling under time-of-use electricity tariffs using fuzzy multi-objective linear programming approach. *Journal of Mathematical Modeling*, 9(2), 2021.

[12] S. Bandyopadhyay, S.K. Pal, and B. Aruna. Quantitative indices, and pattern classification. *IEEE Transactions On Systems, Man, And Cybernetics*, 34(5):2088–2099, 2004.

[13] Jose Batista Abikarram, Katie McConky, and Ruben Proano. Energy cost minimization for unrelated parallel machine scheduling under real time and demand charge pricing. *Journal of Cleaner Production*, 208:232–242, 2019. ISSN 09596526.

[14] Peter Brucker, Andreas Drexl, Rolf Möhring, Klaus Neumann, and Erwin Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1):3–41, 1999. ISSN 0377-2217.

[15] Jianhua Cao, Ruilin Pan, Xue Xia, Xuemei Shao, and Xuemin Wang. An efficient scheduling approach for an iron-steel plant equipped with self-generation equipment under time-of-use electricity tariffs. *Swarm and Evolutionary Computation*, 60: 100764, 2021. ISSN 22106502.

[16] Shuchi Chawla, Nikhil R. Devanur, A. Holroyd, Anna R. Karlin, J. B. Martin, and Balasubramanian Sivan. Stability of service under time-of-use pricing. *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 2017.

[17] Ada Che, Yizeng Zeng, and Ke Lyu. An efficient greedy insertion heuristic for energy-conscious single machine scheduling problem under time-of-use electricity tariffs. *Journal of Cleaner Production*, 129:565–577, 2016. ISSN 09596526.

[18] Ada Che, Shibohua Zhang, and Xueqi Wu. Energy-conscious unrelated parallel machine scheduling under time-of-use electricity tariffs. *Journal of Cleaner Production*, 156:688–697, 2017. ISSN 09596526.

[19] Bo Chen and Xiandong Zhang. Scheduling with time-of-use costs. *European Journal of Operational Research*, 274(3):900–908, 2019. ISSN 03772217.

[20] Lin Chen, Nicole Megow, Roman Rischke, Leen Stougie, and José Verschae. Optimal algorithms for scheduling under time-of-use tariffs. *Annals of Operations Research*, 304:85–107, 2021. ISSN 0254-5330, 1572-9338.

[21] Shih-Hsin Chen, Yeong-Cheng Liou, Yi-Hui Chen, and Kun-Ching Wang. Order acceptance and scheduling problem with carbon emission reduction and electricity tariffs on a single machine. *Sustainability*, 11(19):5432, 2019. ISSN 2071-1050.

[22] Junheng Cheng, Feng Chu, Weili Xia, Jianxun Ding, and Xiang Ling. Bi-objective optimization for single-machine batch scheduling considering energy cost. In *2014 International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 236–241. IEEE, 2014. ISBN 978-1-4799-6773-5.

[23] Junheng Cheng, Feng Chu, Chengbin Chu, and Weili Xia. Bi-objective optimization of single-machine batch scheduling under time-of-use electricity prices. *RAIRO - Operations Research*, 50(4):715–732, 2016. ISSN 0399-0559, 1290-3868.

[24] Junheng Cheng, Feng Chu, Ming Liu, and Weili Xia. Single-machine batch scheduling under time-of-use tariffs: New mixed-integer programming approaches. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 003498–003503. IEEE, 2016. ISBN 978-1-5090-1897-0.

[25] Junheng Cheng, Feng Chu, Ming Liu, Peng Wu, and Weili Xia. Bi-criteria single-machine batch scheduling with machine on/off switching under time-of-use tariffs. *Computers & Industrial Engineering*, 112:721–734, 2017. ISSN 03608352.

[26] Junheng Cheng, Feng Chu, and Mengchu Zhou. An improved model for parallel machine scheduling under time-of-use electricity price. *IEEE Transactions on Automation Science and Engineering*, 15(2):896–899, 2018. ISSN 1545-5955, 1558-3783.

[27] Junheng Cheng, Peng Wu, and Feng Chu. 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)*, pages 1–5. IEEE, 2019. ISBN 978-1-72811-566-5.

[28] Xingrui Cheng, Feng Gao, Chaobo Yan, Xiaohong Guan, Kun Liu, Siyun Chen, Nana Yao, and Jing Cai. Permutation flow shop scheduling with delay time under time-of-use electricity tariffs. In *2016 12th World Congress on Intelligent Control and Automation (WCICA)*, pages 2743–2748. IEEE, 2016. ISBN 978-1-4673-8414-8.

[29] Tonmoy Toufic Choudhury, Sanjoy Kumar Paul, Humyun Fuad Rahman, Zhenguo Jia, and Nagesh Shukla. A systematic literature review on the service supply chain: Research agenda and future research directions. *Production Planning & Control*, 31 (16):1363–1384, 2020.

[30] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[31] Weiwei Cui and Biao Lu. Energy-aware operations management for flow shops under TOU electricity tariff. *Computers & Industrial Engineering*, 151:106942, 2021. ISSN 03608352.

[32] Weiwei Cui, Huali Sun, and Beixin Xia. Integrating production scheduling, maintenance planning and energy controlling for the sustainable manufacturing systems under TOU tariff. *Journal of the Operational Research Society*, 71(11):1760–1779, 2020. ISSN 0160-5682, 1476-9360.

[33] Viviane Grunert Da Fonseca, Carlos M Fonseca, and Andreia O Hall. Inferential performance assessment of stochastic optimisers and the attainment function. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 213–225. Springer, 2001.

[34] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transaction on Evolutionary Computation*, 18(4):577—-601, 2014.

[35] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-II. *IEEE Transaction on Evolutionary Computation*, 6(2): 182—197, 2002.

[36] M. Despeisse, P. Ball, S. Evans, and A. Levers. Industrial ecology at factory level – a conceptual model. *Journal of Cleaner Production*, 31:30–39, 2012.

[37] Jian-Ya Ding, Shiji Song, Rui Zhang, Raymond Chiong, and Cheng Wu. Parallel machine scheduling under time-of-use electricity prices: New models and optimization approaches. *IEEE Transactions on Automation Science and Engineering*, 13 (2):1138–1154, 2016. ISSN 1545-5955, 1558-3783.

[38] Junwen Ding, Sven Schulz, Liji Shen, Udo Buscher, and Zhipeng Lü. Energy aware scheduling in flexible flow shops with hybrid particle swarm optimization. *Computers & Operations Research*, 125:105088, 2021. ISSN 03050548.

[39] Baigang Du, Tian Tan, Jun Guo, Yibing Li, and Shunsheng Guo. Energy-cost-aware resource-constrained project scheduling for complex product system with activity splitting and recombining. *Expert Systems with Applications*, 173:114754, 2021. ISSN 0957-4174.

[40] U.S. Energy Information Administration (EIA). Manufacturing energy consumption survey. 2021. URL https://www.eia.gov/consumption/manufacturing/pdf/MECS%202018%20Results%20Flipbook.pdf.

[41] International Energy Agency. World Energy Outlook 2016. 2016. URL https://www.eia.gov/outlooks/ieo/pdf/0484(2016).pdf.

[42] International Energy Agency. World Energy Outlook 2019. 2019. URL https://iea.blob.core.windows.net/assets/98909c1b-aabc-4797-9926-35307b418cdb/WEO2019-free.pdf.

[43] Kan Fang, Nelson A. Uhan, Fu Zhao, and John W. Sutherland. Scheduling on a single machine under time-of-use electricity tariffs. *Annals of Operations Research*, 238(1):199–227, 2016. ISSN 0254-5330, 1572-9338.

[44] Christian Gahm, F. Denz, Martin Dirr, and A. Tuma. Energy-efficient scheduling in manufacturing companies: A review and research framework. *European Journal of Operational Research*, 248:744–757, 2016.

[45] K. Gao, Y. Huang, Ali Sadollah, and L. Wang. A review of energy-efficient scheduling in intelligent production systems. *Complex & Intelligent Systems*, 6:237–249, 2020.

[46] M. R. Garey and David S. Johnson. " strong " np-completeness results: Motivation, examples, and implications. *J. ACM*, 25:499–508, 1978.

[47] Kaifeng Geng, Chunming Ye, Zhen hua Dai, and Li Liu. Bi-objective re-entrant hybrid flow shop scheduling considering energy consumption cost under time-of-use electricity tariffs. *Complexity*, 2020:1–17, 2020. ISSN 1076-2787, 1099-0526.

[48] A. Giret, D. Trentesaux, and V. Prabhu. Sustainability in manufacturing operations scheduling: A state of the art review. *Journal of Manufacturing Systems*, 37:126–140, 2015.

[49] Xu Gong, Toon De Pessemier, Wout Joseph, and Luc Martens. A generic method for energy-efficient and energy-cost-effective production at the unit process level. *Journal of Cleaner Production*, 113:508–522, 2016. ISSN 09596526.

[50] Xu Gong, Toon De Pessemier, Luc Martens, and Wout Joseph. Energy- and labor-aware flexible job shop scheduling under dynamic electricity pricing: A many-objective optimization investigation. *Journal of Cleaner Production*, 209:1078–1094, 2019. ISSN 09596526.

[51] R. Graham, E. Lawler, J. Lenstra, and A. Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326, 1977.

[52] Stephen C. Graves, Harlan C. Meal, Daniel Stefek, and Abdel Hamid Zeghmi. Scheduling of re-entrant flow shops. *Journal of Operations Management*, 3(4):197–207, 1983. ISSN 0272-6963.

[53] Andreia P. Guerreiro, Carlos M. Fonseca, and Luís Paquete. The hypervolume indicator: Problems and algorithms. *ArXiv*, abs/2005.00515, 2020.

[54] Wang Guirong and Li Qiqiang. Solving the steelmaking-continuous casting production scheduling problem with uncertain processing time under the TOU electricity price. In *2017 Chinese Automation Congress (CAC)*, pages 1383–1388. IEEE, 2017. ISBN 978-1-5386-3524-7.

[55] Hubert Hadera, Iiro Harjunkoski, Guido Sand, Ignacio E. Grossmann, and Sebastian Engell. Optimization of steel production scheduling with complex time-sensitive electricity cost. *Computers & Chemical Engineering*, 76:117–136, 2015. ISSN 00981354.

[56] Y. Haimes, L. Lasdon, and D. Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 1:296–297, 1971.

[57] E. Heydarian-Forushani, M.E.H. Golshan, and M. Shafie-khah. Flexible security-constrained scheduling of wind power enabling time of use pricing scheme. *Energy*, 90:1887–1900, 2015. ISSN 03605442.

[58] Minh Hung Ho, Faicel Hnaien, and Frederic Dugardin. Electricity cost minimisation for optimal makespan solution in flow shop scheduling under time-of-use tariffs. *International Journal of Production Research*, 59(4):1041–1067, 2021. ISSN 0020-7543, 1366-588X.

[59] F. Hu, X. Feng, and Hui Cao. A short-term decision model for electricity retailers: Electricity procurement and time-of-use pricing. *Energies*, 11:3258, 2018.

[60] H. Ishibuchi, T. Yoshida, and T. Murata. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2):204–223, 2003.

[61] Shima Javanmard, Behrouz Afshar-Nadjafi, and Seyed Taghi Akhavan Niaki. A bi-objective model for scheduling of multiple projects under multi-skilled workforce for distributed load energy usage. *Operational Research*, 2021. ISSN 1109-2858, 1866-1505.

[62] En-da Jiang and Ling Wang. Multi-objective optimization based on decomposition for flexible job shop scheduling under time-of-use electricity prices. *Knowledge-Based Systems*, 204:106177, 2020. ISSN 09507051.

[63] Min Kong, Jin Xu, Tinglong Zhang, Shaojun Lu, Chang Fang, and Nenad Mladenovic. Energy-efficient rescheduling with time-of-use energy cost: Application of variable neighborhood search algorithm. *Computers & Industrial Engineering*, 156: 107286, 2021. ISSN 03608352.

[64] B. Korte and J. Vygen. *Combinatorial Optimization*. Springer-Verlag, Germany, 2018.

[65] Janardhan Kulkarni and Kamesh Munagala. Algorithms for cost-aware scheduling. In Thomas Erlebach and Giuseppe Persiano, editors, *Approximation and Online Algorithms*, pages 201–214, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-38016-7.

[66] B Kurniawan and S Fujimura. Optimization of bi-objective permutation flow shop scheduling with electricity cost consideration. *IOP Conference Series: Materials Science and Engineering*, 909:012045, 2020. ISSN 1757-899X.

[67] B. Kurniawan, A. A. Gozali, W. Weng, and S. Fujimura. A genetic algorithm for unrelated parallel machine scheduling minimizing makespan cost and electricity cost under time-of-use (TOU) tariffs with job delay mechanism. In *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 583–587, Singapore, 2017. IEEE. ISBN 978-1-5386-0948-4.

[68] B. Kurniawan, A. A. Gozali, W. Weng, and S. Fujimura. A mix integer programming model for bi-objective single machine with total weighted tardiness and electricity cost under time-of-use tariffs. In *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 137–141. IEEE, 2018. ISBN 978-1-5386-6786-6.

[69] Bobby Kurniawan, Widyaning Chandramitasari, Alfian Akbar Gozali, Wei Weng, and Shigeru Fujimura. Triple-chromosome genetic algorithm for unrelated parallel machine scheduling under time-of-use tariffs. *IEEJ Transactions on Electrical and Electronic Engineering*, 15(2):208–217, 2020. ISSN 1931-4973, 1931-4981.

[70] Bobby Kurniawan, Wen Song, Wei Weng, and Shigeru Fujimura. Distributed-elite local search based on a genetic algorithm for bi-objective job-shop scheduling under time-of-use tariffs. *Evolutionary Intelligence*, 2020. ISSN 1864-5909, 1864-5917.

[71] E. Lawler and J. Labetoulle. On preemptive scheduling of unrelated parallel processors by linear programming. *Journal of the Association for Computing Machinery*, 25:612–619, 1978.

[72] Seokgi Lee, Byung Do Chung, Hyun Woo Jeon, and Jaeyeon Chang. A dynamic control approach for energy-efficient production scheduling on a single machine under time-varying electricity pricing. *Journal of Cleaner Production*, 165:552–563, 2017. ISSN 09596526.

[73] Mengyun Li, Tao Li, Shitong Peng, and Yanchun Guo. Batch scheduling of remanufacturing flexible job shop for minimal electricity- and time-cost. In Shilong Wang, Mark Price, Ming K. Lim, Yan Jin, Yuanxin Luo, and Rui Chen, editors, *Recent Advances in Intelligent Manufacturing*, volume 923, pages 300–307. Springer Singapore, 2018. ISBN 9789811323959 9789811323966. Series Title: Communications in Computer and Information Science.

[74] D. Liu, Y. Gao, T. Zhang, Z. Dong, H. Huang, C. Wang, and S. Xi. Research on virtual power plant optimal scheduling based on tou electricity price. volume 257, 2021.

[75] Manuel López-Ibánez, Luis Paquete, and Thomas Stützle. Hybrid population-based algorithms for the bi-objective quadratic assignment problem. *Journal of Mathematical Modelling and Algorithms*, 5(1):111–137, 2006.

[76] Manuel López-Ibánez, Luis Paquete, and Thomas Stützle. Exploratory analysis of stochastic local search algorithms in biobjective optimization. In *Experimental methods for the analysis of optimization algorithms*, pages 209–222. Springer, 2010.

[77] S. Luke. *Essentials of Metaheuristics*. Lulu, second edition, 2013. URL `https://cs.gmu.edu/~sean/book/metaheuristics/Essentials.pdf`.

[78] Hao Luo, Bing Du, George Q. Huang, Huaping Chen, and Xiaolin Li. Hybrid flow shop scheduling considering machine electricity consumption cost. *International Journal of Production Economics*, 146(2):423–439, 2013. ISSN 09255273.

[79] Hamidreza Maghsoudlou, Behrouz Afshar-Nadjafi, and Seyed Taghi Akhavan Niaki. A framework for preemptive multi-skilled project scheduling problem with time-of-use energy tariffs. *Energy Systems*, 12(2):431–458, 2021. ISSN 1868-3967, 1868-3975.

[80] Gerardo Minella, Rubén Ruiz, and Michele Ciavotta. Restarted iterated pareto greedy algorithm for multi-objective flowshop scheduling problems. *Computers & Operations Research*, 38(11):1521–1533, 2011.

[81] Joon-Yung Moon, Kitae Shin, and Jinwoo Park. 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, 2013. ISSN 0268-3768, 1433-3015.

[82] Hamid Najafzad, Hamed Davari-Ardakani, and Reza Nemati-Lafmejani. Multi-skill project scheduling problem under time-of-use electricity tariffs and shift differential payments. *Energy*, 168:619–636, 2019. ISSN 03605442.

[83] Mehdi Nikzad and Abouzar Samimi. Integration of optimal time-of-use pricing in stochastic programming for energy and reserve management in smart micro-grids. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, pages 1–18, 2020.

[84] Ruilin Pan, Zhenghong Li, Jianhua Cao, Hongliang Zhang, and Xue Xia. Electrical load tracking scheduling of steel plants under time-of-use tariffs. *Computers & Industrial Engineering*, 137:106049, 2019. ISSN 03608352.

[85] Zhi Pei, Mingzhong Wan, Zhong-Zhong Jiang, Ziteng Wang, and Xu Dai. An approximation algorithm for unrelated parallel machine scheduling under TOU electricity tariffs. *IEEE Transactions on Automation Science and Engineering*, 18(2): 743–756, 2021. ISSN 1545-5955, 1558-3783.

[86] Lining Peng, Mingshun Yang, and Renhao Xiao. An integer programming model for flow shop scheduling under TOU and tiered electricity price. *IOP Conference Series: Earth and Environmental Science*, 692(2):022105, 2021. ISSN 1755-1307, 1755-1315.

[87] Michal Penn and Tal Raviv. Complexity and algorithms for min cost and max profit scheduling under time-of-use electricity tariffs. *Journal of Scheduling*, 24(1):83–102, 2021. ISSN 1094-6136, 1099-1425.

[88] M.L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 5th edition, 2016.

[89] Si-yuan Qian, Zhao-hong Jia, and Kai Li. A multi-objective evolutionary algorithm based on adaptive clustering for energy-aware batch scheduling problem. *Future Generation Computer Systems*, 113:441–453, 2020. ISSN 0167739X.

[90] Jens Rocholl, Lars Mönch, and John Fowler. Bi-criteria parallel batch machine scheduling to minimize total weighted tardiness and electricity cost. *Journal of Business Economics*, 90(9):1345–1381, 2020. ISSN 0044-2372, 1861-8928.

[91] Saeed Rubaiee and Mehmet Bayram Yildirim. An energy-aware multiobjective ant colony algorithm to minimize total completion time and energy cost on a single-machine preemptive scheduling. *Computers & Industrial Engineering*, 127:240–252, 2019. ISSN 03608352.

[92] Hossein Saberi-Aliabad, Mohammad Reisi-Nafchi, and Ghasem Moslehi. Energy-efficient scheduling in an unrelated parallel-machine environment under time-of-use electricity tariffs. *Journal of Cleaner Production*, 249:119393, 2020. ISSN 09596526.

[93] Robert Sedgewick and Kevin Wayne. *Algorithms, 4th Edition*. Addison-Wesley Professional, 2011.

[94] Abhay Sharma, Fu Zhao, and John W. Sutherland. Econological scheduling of a manufacturing enterprise operating under a time-of-use electricity tariff. *Journal of Cleaner Production*, 108:256–270, 2015. ISSN 09596526.

[95] Fadi Shrouf, Joaquin Ordieres-Meré, Alvaro García-Sánchez, and Miguel Ortega-Mier. Optimizing the production scheduling of a single machine to minimize total energy consumption costs. *Journal of Cleaner Production*, 67:197–207, 2014. ISSN 09596526.

[96] S. Sichilalu, X. Xia, and J. Zhang. Optimal scheduling strategy for a grid-connected photovoltaic system for heat pump water heaters. *Energy Procedia*, 61:1511–1514, 2014. ISSN 18766102.

[97] In Ho Sin and Byung Do Chung. Bi-objective optimization approach for energy aware scheduling considering electricity cost and preventive maintenance using genetic algorithm. *Journal of Cleaner Production*, 244:118869, 2020. ISSN 09596526.

[98] Inês Soares, Maria João Alves, and Carlos Henggeler Antunes. A deterministic bounding procedure for the global optimization of a bi-level mixed-integer problem. *European Journal of Operational Research*, 291(1):52–66, 2021. ISSN 03772217.

[99] Mao Tan, Hua-li Yang, Bin Duan, Yong-xin Su, and Feng He. Optimizing production scheduling of steel plate hot rolling for economic load dispatch under time-of-use electricity pricing. *Mathematical Problems in Engineering*, 2017:1–13, 2017. ISSN 1024-123X, 1563-5147.

[100] Mao Tan, Bin Duan, and Yongxin Su. Economic batch sizing and scheduling on parallel machines under time-of-use electricity pricing. *Operational Research*, 18(1): 105–122, 2018. ISSN 1109-2858, 1866-1505.

[101] Mao Tan, Hua-Li Yang, and Yong-Xin Su. Genetic algorithms with greedy strategy for green batch scheduling on non-identical parallel machines. *Memetic Computing*, 11(4):439–452, 2019. ISSN 1865-9284, 1865-9292.

[102] Xinlin Tan and Weiwei Cui. Production scheduling problem under peak power constraint. In *2020 IEEE Sustainable Power and Energy Conference (iSPEC)*, pages 2083–2088. IEEE, 2020. ISBN 978-1-72819-164-5.

[103] Guohua Wan and Xiangtong Qi. Scheduling with variable time slot costs. *Naval Research Logistics (NRL)*, 57(2):159–171, 2010. ISSN 0894069X.

[104] Fang Wang, Pan Qin, and Jing You. Discrete optimization model for permutation flow shop under time-of-use electricity tariffs. *Journal of Physics: Conference Series*, 1633:012146, 2020. ISSN 1742-6588, 1742-6596.

[105] Huiji Wang, Yanan Zheng, and Ming Zhou. Unit scheduling considering the flexibility of intelligent temperature control appliances under TOU power price. *International Journal of Electrical Power & Energy Systems*, 125:106477, 2021. ISSN 01420615.

[106] Kai Wang. Calibration scheduling with time slot cost. *Theoretical Computer Science*, 821:1–14, 2020. ISSN 03043975.

[107] Shijin Wang, Ming Liu, Feng Chu, and Chengbin Chu. Bi-objective optimization of a single machine batch scheduling problem with energy cost consideration. *Journal of Cleaner Production*, 137:1205–1215, 2016. ISSN 09596526.

[108] Shijin Wang, Xiaodong Wang, Jianbo Yu, Shuan Ma, and Ming Liu. Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan. *Journal of Cleaner Production*, 193:424–440, 2018. ISSN 09596526.

[109] Shijin Wang, Zhanguo Zhu, Kan Fang, Feng Chu, and Chengbin Chu. Scheduling on a two-machine permutation flow shop under time-of-use electricity tariffs. *International Journal of Production Research*, 56(9):3173–3187, 2018. ISSN 0020-7543, 1366-588X.

[110] Shijin Wang, Xiaodong Wang, Feng Chu, and Jianbo Yu. An energy-efficient two-stage hybrid flow shop scheduling problem in a glass production. *International Journal of Production Research*, 58(8):2283–2314, 2020. ISSN 0020-7543, 1366-588X.

[111] Peng Wu, Junheng Cheng, and Feng Chu. Large-scale energy-conscious bi-objective single-machine batch scheduling under time-of-use electricity tariffs via effective iterative heuristics. *Annals of Operations Research*, 296(1):471–494, 2021. ISSN 0254-5330, 1572-9338.

[112] Zijian Wu, Kaili Yang, Jiangxin Yang, Yanlong Cao, and Yi Gan. Energy-efficiency-oriented scheduling in smart manufacturing. *Journal of Ambient Intelligence and Humanized Computing*, 10(3):969–978, 2019. ISSN 1868-5137, 1868-5145.

[113] Ye Yang, Weida Chen, Li Wei, and Xu Chen. Robust optimization for integrated scrap steel charge considering uncertain metal elements concentrations and production scheduling under time-of-use electricity tariff. *Journal of Cleaner Production*, 176:800–812, 2018. ISSN 09596526.

[114] YiZeng Zeng, Ada Che, and Xueqi Wu. Bi-objective scheduling on uniform parallel machines considering electricity cost. *Engineering Optimization*, 50(1):19–36, 2018. ISSN 0305-215X, 1029-0273.

[115] Yu-jiao Zeng and Yan-guang Sun. Short-term scheduling of steam power system in iron and steel industry under time-of-use power price. *Journal of Iron and Steel Research International*, 22(9):795–803, 2015. ISSN 1006-706X, 2210-3988.

[116] Zhiqiang Zeng, Xiaobin Chen, and Kaiyao Wang. Energy saving for tissue paper mills by energy-efficiency scheduling under time-of-use electricity tariffs. *Processes*, 9(2):274, 2021. ISSN 2227-9717.

[117] Hao Zhang, Fu Zhao, Kan Fang, and John W. Sutherland. Energy-conscious flow shop scheduling under time-of-use electricity tariffs. *CIRP Annals*, 63(1):37–40, 2014. ISSN 00078506.

[118] Hao Zhang, Fu Zhao, and John W. Sutherland. Energy-efficient scheduling of multiple manufacturing factories under real-time electricity pricing. *CIRP Annals*, 64 (1):41–44, 2015. ISSN 00078506.

[119] Hongliang Zhang, Youcai Fang, Ruilin Pan, and Chuanming Ge. A new greedy insertion heuristic algorithm with a multi-stage filtering mechanism for energy-efficient single machine scheduling problems. *Algorithms*, 11(2):18, 2018. ISSN 1999-4893.

[120] Hongliang Zhang, Yujuan Wu, Ruilin Pan, and Gongjie Xu. Two-stage parallel speed-scaling machine scheduling under time-of-use tariffs. *Journal of Intelligent Manufacturing*, 32(1):91–112, 2021. ISSN 0956-5515, 1572-8145.

[121] Hua Zhang, Ziwei Dai, Wenyu Zhang, Shuai Zhang, Yan Wang, and Rongyu Liu. A new energy-aware flexible job shop scheduling method using modified biogeography-based optimization. *Mathematical Problems in Engineering*, 2017:1–12, 2017. ISSN 1024-123X, 1563-5147.

[122] Mingyang Zhang, Jihong Yan, Yanling Zhang, and Shenyi Yan. Optimization for energy-efficient flexible flow shop scheduling under time of use electricity tariffs. *Procedia CIRP*, 80:251–256, 2019. ISSN 22128271.

[123] Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[124] Shibohua Zhang, Ada Che, Xueqi Wu, and Chengbin Chu. Improved mixed-integer linear programming model and heuristics for bi-objective single-machine batch scheduling with energy cost consideration. *Engineering Optimization*, 50(8): 1380–1394, 2018. ISSN 0305-215X, 1029-0273.

[125] Shengnan Zhao, Ignacio E. Grossmann, and Lixin Tang. Integrated scheduling of rolling sector in steel production with consideration of energy consumption under time-of-use electricity prices. *Computers & Chemical Engineering*, 111:55–65, 2018. ISSN 00981354.

[126] Xiancong Zhao, Hao Bai, Qi Shi, Xin Lu, and Zhihui Zhang. Optimal scheduling of a byproduct gas system in a steel plant considering time-of-use electricity pricing. *Applied Energy*, 195:100–113, 2017. ISSN 03062619.

[127] Shengchao Zhou, Xiaolin Li, Ni Du, Yan Pang, and Huaping Chen. A multi-objective differential evolution algorithm for parallel batch processing machine scheduling considering electricity consumption cost. *Computers & Operations Research*, 96: 55–68, 2018. ISSN 03050548.

[128] Shengchao Zhou, Mingzhou Jin, and Ni Du. Energy-efficient scheduling of a single batch processing machine with dynamic job arrival times. *Energy*, 209:118420, 2020. ISSN 03605442.

[129] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.