Homayouni, S. Mahdi; Fontes, Dalila B.M.M.

Article — Published Version

Optimizing job shop scheduling with speed-adjustable machines and peak power constraints: A mathematical model and heuristic solutions

International Transactions in Operational Research

Provided in Cooperation with:
John Wiley & Sons

Suggested Citation: Homayouni, S. Mahdi; Fontes, Dalila B.M.M. (2023) : Optimizing job shop scheduling with speed-adjustable machines and peak power constraints: A mathematical model and heuristic solutions, International Transactions in Operational Research, ISSN 1475-3995, Wiley, Hoboken, NJ, Vol. 32, Iss. 1, pp. 194-220, https://doi.org/10.1111/itor.13414

This Version is available at:
https://hdl.handle.net/10419/306214

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

WILEY

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 32 (2025) 194–220
DOI: 10.1111/itor.13414

Optimizing job shop scheduling with speed-adjustable machines and peak power constraints: A mathematical model and heuristic solutions

S. Mahdi Homayouni[a,b,*] and Dalila B.M.M. Fontes[c,d]

[a] *School of Business, Social and Decision Sciences, Constructor University, Campus Ring 1, 28759, Bremen, Germany*
[b] *CESE - INESCTEC, Porto 4200-465, Portugal*
[c] *LIAAD - INESCTEC, Porto 4200-465, Portugal*
[d] *Faculdade de Economia, Universidade do Porto, Porto 4200-464, Portugal*
E-mail: smh@inesctec.pt[Homayouni]; fontes@fep.up.pt[Fontes]

Received 17 March 2023; received in revised form 9 September 2023; accepted 6 November 2023

Abstract

This paper addresses a job shop scheduling problem with peak power constraints, in which jobs can be processed once or multiple times on either all or a subset of the machines. The latter characteristic provides additional flexibility, nowadays present in many manufacturing systems. The problem is complicated by the need to determine both the operation sequence and starting time as well as the speed at which machines process each operation. Due to the adherence to renewable energy production and its intermittent nature, manufacturing companies need to adopt power-flexible production schedules. The proposed power control strategies, that is, adjusting processing speed and timing to reduce peak power requirements may impact production time (makespan) and energy consumption. Therefore, we propose a bi-objective approach that minimizes both objectives. A linear programming model is developed to provide a formal statement of the problem, which is solved to optimality for small-sized instances. We also proposed a multi-objective biased random key genetic algorithm framework that evolves several populations in parallel. Computational experiments provide decision and policymakers with insights into the implications of imposing or negotiating power consumption limits. Finally, the several trade-off solutions obtained show that as the power limit is lowered, the makespan increases at an increasing rate and a similar trend is observed in energy consumption but only for very small makespan values. Furthermore, peak power demand reductions of about 25% have a limited impact on the minimum makespan value (4–6% increase), while at the same time allowing for a small reduction in energy consumption.

Keywords: job shop scheduling problem; energy efficiency; peak power; speed-adjustable machines; biased random key genetic algorithm

*Corresponding author.

1. Introduction

Energy efficiency is of uttermost importance not only due to the high energy costs incurred but also due to the environmental impacts associated with energy consumption. In 2020, the industry sector was responsible for 26% and 35% of the energy consumed in the European Union (EU) and the United States, respectively (EIA, 2021; EUROSTAT, 2022a). Within the industrial sector, manufacturing accounts for about 80% of energy consumption. Data for 2020 show that about one-third of the energy consumed in the EU's industry sector is electricity, most of it to operate equipment. Although the environmental performance of manufacturing has improved over recent decades, it is currently responsible for 17% of the U.S. emissions and is believed to increase to about 26% by 2050, while emissions of other high producing sectors will hold steady or decline (NIST, 2020). Additionally, since most of the environmental impacts associated with manufacturing activities are due to electrical energy consumption, improving energy efficiency may result in potentially large cost reductions as well as lower environmental impacts.

The production and use of renewable energy have been increasing; for example, in the EU, renewable energy resources account for about 37% of the electricity generated and about 22% of the electricity consumed EUROSTAT (2022b). However, renewable energy generation is irregular, particularly wind and solar energy generation, as it depends on meteorological conditions. In addition to the large power requirements, due to the different nature of manufacturing activities, these requirements vary significantly. Together, these two sources of power variability provoke large, and to some extent unpredictable, swings in net demand that impact the power quality of the network as a whole. Furthermore, large peaks in power consumption require significant capital investment by the electric power companies, which, in turn, lead to an increase in electricity charges. Hence, manufacturing companies have the social responsibility of reducing not only energy consumption but also peak power consumption.

Although balancing energy efficiency, power demand and production targets is very challenging, there are several alternatives available, such as using more efficient machinery and sustainable materials and processes. Nevertheless, most of such alternatives require substantial financial investment products and process changes and adaptation. In contrast, scheduling optimization can be used to improve energy efficiency and reduce peak power consumption at almost no cost (Gao et al., 2020).

Several works have been published on the job shop scheduling problem (JSP), and some of its variants consider energy efficiency (for recent and comprehensive reviews, see, e.g., Gao et al., 2020; Fernandes et al., 2022; Márquez and Ribeiro, 2022; Xiong et al., 2022). Most of the research on JSP with energy considerations addresses energy efficiency by tackling the original problem while optimizing some energy-related objectives. According to Fernandes et al. (2022), almost 70% of the works published between 2011 and March 2022 do so. The energy objectives have been, typically, optimized together with performance objectives (e.g., makespan, earliness or tardiness) and/or economic objectives (e.g., production costs). Among the energy-related performance measures, energy consumption is the most used one. More recently and in response to the utility's current trend of introducing time-dependent energy prices, such as time of use tariffs and critical peak pricing, many works minimizing energy costs have been reported (see, e.g., Borges et al., 2020; Catanzaro et al., 2023).

Also recently, other authors have studied JSP versions that, although optimizing energy-related performance measures, consider additional decisions, such as choosing the machine state when not working (leaving it idle or switching it off or to a standby mode) and choosing the operations processing speed. Several authors have addressed scheduling problems in which the machines have several energy states. In most works, the states include on (when the machine is processing), idle (when the machine is on, but waiting for an operation) and off (when the machine is shut down). However, some works also consider that a machine may be on standby, similar to idle but with lower energy consumption. A very small number of works consider transition states associated with switching the machines to the standby and off states (ramp down) as well as switching them back on (ramp up) (see, e.g., Gonzalez et al., 2022). Recent literature on the energy-efficient JSP has favoured speed-adjustable machines, rather than machines with multiple energy states since turning the machine on and off/standby frequently may damage them. In addition, switching the machines off/standby and then back on implies transition states and thus, may lead to a spike in power consumption and require a setup time. On the other hand, if the transition states are ignored then, as the machines' processing speed, processing time and power requirements are all constant, minimizing the total energy/power consumption amounts to minimizing the total machines' idle time as only the non-processing energy/power consumption can be reduced (Afsar et al., 2022). According to Fernandes et al. (2022), between 2011 and March 2022, among the energy-efficient JSP works published incorporated additional problem characteristics, 22 considered adjustable speed machines and seven considered machines with multiple energy states.

Regarding adjustable speed machines, it is generally accepted that processing operations at a higher speed lead to smaller processing times and larger energy requirements. Clearly, there exists a conflict between processing time and required energy, and hence many different trade-off solutions exist. The choice of the speed with which the machines process the operations can be used to reduce energy consumption while maintaining the level of service (e.g., makespan); for example, by decelerating operations processing as much as possible without affecting any other operations. On the other hand, such a choice can be used to improve the level of service (i.e., reduce the makespan) while maintaining energy consumption; for example, by simultaneously speeding up the processing of critical operations and slowing down the processing of non-critical ones. Furthermore, as shown by Fontes et al. (2023), it is even possible, in some cases, to improve both the energy consumption and the makespan simultaneously. Although not many works on the JSP with speed-adjustable machines have yet been reported, an increasing trend can be observed in the literature (Fernandes et al., 2022). This problem was first introduced by Tang and Dai (2015), who proposed a mixed integer programming (MIP) model and a genetic and simulating annealing algorithm to improve the energy consumption of a given schedule by adjusting the speed with which each operation is processed. The results they reported show that, for small-sized instances, it is possible to reduce the energy consumption, between 5% and 10%, without increasing (or increasing very slightly) the makespan. For larger ones, the energy savings range from 0.5% to 4%, but the makespan increases by 1% and up to 7%.

The power demands of manufacturing activities have been largely ignored, in favour of reducing energy consumption or energy costs by minimizing energy consumption or time of use energy costs, respectively. However, power consumption is a very important factor. On the one hand, usually, industrial consumers have a contract that specifies a charge for energy consumption (i.e., the rate at which each kWh is paid) and a charge for the maximum power (peak power) contracted; quite

often, it is not possible or prohibitively expensive to go beyond the power limit contracted. On the other hand, the (increasing) use of renewable energy sources cannot respond to surges in power demand. Hence, to satisfy the peak power demand, which is increasing at a faster rate than the average power demand, implies an increase in the infrastructure investment, which in turn, leads to an increase in infrastructure costs. Therefore, all consumers end up paying more to satisfy their energy needs.

Fang et al. (2011) seem to have been the first to address scheduling problems by tackling peak power requirements directly. They propose a multi-objective MIP model for a flow shop problem with speed-adjustable machines that minimize the makespan, the peak power and the carbon footprint. The model proved to be extremely difficult to solve as shown by resorting to a simple two-machine flow shop with and without machine buffers. Later, they considered a similar problem, that is, a flow shop in which the makespan is to be minimized, and the maximum power (peak power) required at any given time is limited to a given threshold (Fang et al., 2013). Two MIP formulations are proposed, namely, a disjunctive formulation based on that of Manne (1960) and an assignment and position formulation that is a variant of the one previously proposed by Fang et al. (2011). The computational experiments were performed using the example in Fang et al. (2011).

However, work on the JSP considering power consumption is rare. As far as we know, only five such works have been reported: some minimizing the power consumption and others imposing a power consumption threshold (Kawaguchi and Fukuyama, 2016; Kemmoe et al., 2017; Masmoudi et al., 2019; Gondran et al., 2020; Carlucci et al., 2023).

Kawaguchi and Fukuyama (2016) propose a reactive tabu search to find solutions to a JSP in which every job is processed exactly once on each machine, and the machines have a single processing speed. They propose a single-objective approach since they minimize the weighted sum of the makespan and the total power consumed over the peak time interval (the daytime interval during which electricity demand is higher). Additionally, they restrict the maximum power availability at each time period in the peak time interval. Computational experiments were conducted on one instance with six jobs and six machines in which the peak power interval contains 12 time periods. The authors show that by imposing a power limit (10 kW) during the peak time interval and by minimizing the total power consumed during that time interval, there is a shift in the power consumed to time periods outside of the peak interval. However, it should be noticed that the power peak observed outside the peak interval is very high (over 14 kW). Indeed, the power consumption is not evenly distributed; actually, the highest values of power consumption are four times larger than the lowest ones.

Kemmoe et al. (2017) propose a mixed-integer linear programming (MILP) model to minimize the makespan in a job-shop system in which each job must be processed exactly once on each machine and a power threshold, which cannot be exceeded at any time, is imposed. Although machines have a single processing speed, operations have a power profile that includes more than one segment and thus have a power consumption that varies with the processing stage. As in Weinert et al. (2011), each operation is modelled as a set of sub-operations that are processed consecutively without delays. Two sub-operations are considered: the first one consumes more power than the second one. The disjunctive MILP model proposed is an improved version of that of Kemmoé-Tchomté et al. (2015), which is based on the one proposed in Roy and Sussmann (1964). Given the problem's complexity, they also propose a greedy randomized adaptive search procedure hybridized with an evolutionary local search (GRASP×ELS) metaheuristic, which Prins

(2009) has shown to be successful on several combinatorial problems. The authors have generated several problem instances to test their approaches. For small-sized instances, the GRASP×ELS solutions were compared to the optimal solutions obtained by solving the MILP using CPLEX. For larger instances, the GRASP×ELS was compared to two other heuristics: a variable neighbourhood search (VNS) and a memetic algorithm (MA) adapted from the one reported in Wang et al. (2012). The results have shown that the MILP model can only be solved optimality for very small-sized instances. The optimality gap increases very quickly with instance size; for instances with 32–40 operations (8–10 jobs and four machines), the optimality gap can be as high as 80%. The VNS is the fastest of the three heuristics; however, it is also the worst regarding solution quality. The GRASP×ELS and the MA find almost always the same best solution and require a similar computational effort. Finally, the GRASP×ELS is the most robust as the standard deviation values are the lowest. More recently, Gondran et al. (2020) address a similar problem; the main difference being that the power threshold, not to be exceeded, is now minimized. Hence, they address a bi-objective JSP, since they minimize both the makespan and the peak power, for which they find a set of Pareto front schedules. As in Kemmoe et al. (2017), each operation is modelled as a succession of two sub-operations that must be processed consecutively and without delays, each with its own processing time and power demand. Two heuristics are proposed to find solutions in the Pareto front, a hybrid NSGA-II (Deb et al., 2002) and an iterated GRASP×ELS adapted from the one proposed in Kemmoe et al. (2017). The computational experiments involved a set of 40 problem instances adapted from those proposed by Lawrence (1984) for the JSP. The results show that, on average, the hybrid NSGA-II has better performance regarding solution quality and distribution and a number of non-dominated solutions. However, the iterated GRASP×ELS finds solutions with smaller makespans in the presence of higher peak power values.

Another work imposing a peak power limit is that of Masmoudi et al. (2019). They minimize the total energy costs incurred, which are calculated by considering that the planning horizon is divided into several time periods and associating a specific energy price with each of these periods. However, they do not consider productivity performance measures such as makespan. Furthermore, the machines have only one single processing speed. Optimal solutions are obtained by resorting to two MILP models, namely a disjunctive formulation based on that of Manne (1960) and a time-indexed formulation based on that of Bowman (1959). They report computational experiments involving one instance taken from Fisher and Thompson (1963) with six jobs and six machines (mt06) and another taken from Lawrence (1984) with 10 jobs and five machines (la04). From these two instances, they derived 90 instances. The results show that the time-indexed model struggles to find feasible solutions; however, when it does the optimality gap is smaller than that of the disjunctive model.

A JSP with peak power constraints and speed-adjustable machines was addressed, for the first time, in Carlucci et al. (2023) where every job must be processed exactly once on each machine and minimizes the makespan. As was the case in Kemmoe et al. (2017) and Masmoudi et al. (2019), the approach proposed is a disjunctive mathematical model based on that of Manne (1960). They report on two computational experiments conducted on one instance with three jobs and four machines, each with 20 possible speed values. The first experiment considers a constant power threshold for the whole horizon, as this is, typically, the case when supplied by a utility company. Regarding the second case, a variable threshold is considered as this would be the case in the presence of in-house renewable generation (e.g., photovoltaic system). As expected, the processing speed is

adjusted to the available power and, hence, the value obtained for the makespan is larger in the presence of power constraints. It was observed that in the presence of a variable power threshold, most operations were processed at full speed when the (renewable) system supplies more power and at the lowest speed when the power available is at its lowest. This is not surprising since the authors are only concerned with the makespan. A more realistic version of this problem should optimize the energy consumption along with the makespan.

As seen, very few works incorporate peak power constraints; however, it is common to have such constraints in industry. The main motivations to consider peak power constraints are, on the one hand, the nature of power and energy charges and contracts and, on the other hand, the several issues associated with renewable energy production (intermittency, unpredictability and lack of affordable power storage). Hence, this work addresses the JSP with peak power constraints. In line with recent research on scheduling problems, we consider machines that can be operated at more than one speed, that is, speed-adjustable machines. The speed choice impacts power consumption, production time and energy consumption. Typically, the faster the processing speed the larger the power consumption, the lower the processing time and the larger the energy consumption. Therefore, since we are considering peak power limits (upper limits on the value of the maximum instant power consumption), we optimize both the total production time (makespan) and the total energy consumed. In contrast with the papers reviewed above, we consider that jobs may be processed once or more on all machines or just on a subset of them; this way considering the additional flexibility present in many modern manufacturing systems. The problem is formally defined and formulated as a MILP model that can be solved to optimality, but only for small-sized instances. The problem understudy is very complex; indeed, it extends the JSP, which is an NP-hard problem. Hence, heuristic approaches are needed to be able to solve more reasonably sized problem instances. To that end, we propose a multi-objective and multi-population biased random key genetic algorithm (BRKGA) based on that of Fontes and Homayouni (2023).

This work can provide decision-makers and production managers with a tool that balances productivity and energy efficiency while complying with power consumption restrictions, resulting in production plans that align with power availability. Additionally, the smother power profile and the lower power and energy peaks are of uttermost importance since, if generalized, can contribute to preventing fast degradation of the power grid, reducing energy production costs for all and helping a further increase, at a faster pace, the renewable energy penetration. Recall that the industrial sector is known to be the first energy consumer and greenhouse gas emitter in the world.

Hence, the contributions of this work are several: First, we contribute to the definition of a variant of the JSP that considers a shop floor in which (1) the jobs may be processed once or more on all machines or just on a subset of them and (2) the machines are speed adjustable, that is, the machines are capable of processing operations at one of the possible processing speeds as well as peak power constraints, for which we develop a MILP model. To the best of our knowledge, finding a schedule complying with peak power constraints for a job shop environment consisting of a set of speed-adjustable machines and a set of jobs that can be processed once or more on all or a subset of the given machines while minimizing both the makespan and the energy consumption has never been considered. Second, we proposed a new BRKGA that, based on that of Fontes and Homayouni (2023), adapts the BRKGA framework of Gonçalves and Resende (2011) by simultaneously optimizing two objectives and simultaneously evolving several single-objective and multi-objective populations (mop-BRKGA). Finally, we provide extensive computational experiments to assess the

efficacy of the developed mop-BRKGA and provide relevant managerial insights since by adjusting the processing timings, and the speed of the machines they can adapt the schedules to obtain different trade-offs involving productivity, energy and power. Furthermore, the set of trade-offs provides alternative solutions and information that can be used to discuss and negotiate the power thresholds. This way, managers are provided with a tool that allows them to analyse and rethink their production objectives and practices.

The remainder of this paper is organized as follows. The problem is formally defined in Section 2. Section 3 describes the mop-BRKGA proposed, and Section 4 reports the computational experiments. Finally, Section 5 draws some conclusions and points out future research directions.

2. Problem definition and formulation

The classical JSP comprises a set $\mathcal{J}$ of independent jobs and a set $\mathcal{I}$ of machines. Each job comprises a set $\mathcal{O}_j = \{1, 2, \ldots, n_j\}, \forall j \in \mathcal{J}$ of n_j ordered operations. Each operation must be processed on a predefined machine without preemption. Machines can only process one operation at a time and jobs can only be processed on one machine at a time. Machine buffers are large enough to accommodate any reasonable number of jobs; hence, whenever a machine finishes processing an operation, it can start processing the next operation immediately. Additionally, operations can be processed as soon as the previous operation of the same job is completed, or immediately if it is the job's first operation. However, both the machine and the job need to be available to process an operation.

The problem understudy considers speed-adjustable machines; that is, machines that can process each operation at one of the available speeds, each requiring a predefined time and power. Usually, higher processing speeds imply lower processing times and higher instant power requirements. Machines not actively processing operations are considered in 'stand-by' mode with negligible power requirements. Moreover, we consider that each job may be processed on all machines or a subset of them and that the machines processing a job may process it once or more. Lastly, we also consider peak power constraints.

Among all possible solutions, we are interested in those that provide trade-offs between the makespan and the energy consumed to process all production operations, since we are interested in minimizing both of them. Thus, the energy-efficient JSP with speed-adjustable machines and peak power constraints, designated as $\text{EEJSP}^{\text{pp}}_{\text{sa}}$, requires determining simultaneously the operations sequence on each machine (machine sequencing), the processing start time of each operation and the speed with which each operation is processed (speed assignment) while ensuring that the instant power requirements satisfy the peak power limits. Next, we provide a formal statement of the $\text{EEJSP}^{\text{pp}}_{\text{sa}}$ by formulating it as a MILP model. Let us first introduce the notation used.

Sets, indices and parameters:

$\mathcal{I}$: Set of machines indexed by m;
$\mathcal{O}$: Set of operations indexed by k, l;
$\mathcal{O}'$: Set of jobs' first operation;

$\mathcal{F}$: Set of operation pairs that cannot be processed consecutively, i.e., pairs of operations that do not satisfy the precedence constraints $\mathcal{F} = \{(k, l) : l \leq k, k, l \in \mathcal{O}_j, j \in \mathcal{J}\}$.
$\mathcal{T}$: Set of time periods indexed by t;
$\mathcal{V}_l$: Set of speeds available to process operation $l \in \mathcal{O}$;
m_l: Machine processing operation $l \in \mathcal{O}$;
τ_l^v: Processing time of operation $l \in \mathcal{O}$ at speed $v \in \mathcal{V}_l$;
ρ_l^v: Instant power required to process operation $l \in \mathcal{O}$ at speed $v \in \mathcal{V}_l$;
$\mathcal{P}_t^{limit}$: Maximum instant power available at time $t \in \mathcal{T}$;

Decision and auxiliary variables:

x_{kl}^v: Binary variable set to 1 if a machine processes consecutively operations $k, l \in \mathcal{O}$ and the latter is processed at speed $v \in \mathcal{V}_l$ and set to 0 otherwise (recall that x_{kl}^v is only defined if $(k, l) \notin \mathcal{F}$);
$w_{lm_l}^v$: Binary variable set to 1 if operation $l \in \mathcal{O}$ is the first operation on machine m_l and is processed at speed $v \in \mathcal{V}_l$ and set to 0 otherwise;
z_{lm_l}: Binary 'dummy' variable set to 1 if operation $l \in \mathcal{O}$ is the last operation on machine m_l and set to 0 otherwise;
s_l: Starting time of operation $l \in \mathcal{O}$;
c_l: Completion time of operation $l \in \mathcal{O}$;
θ_{lt}^v: Binary variable set to 1 if operation $l \in \mathcal{O}$ is being processed at speed $v \in \mathcal{V}_l$ at time $t \in \mathcal{T}$ and set to 0 otherwise;
y_{lt}: Binary auxiliary variable set to 1 if operation $l \in \mathcal{O}$ has been started at or before time $t \in \mathcal{T}$ and set to 0 otherwise;
q_{lt}: Binary auxiliary variable set to 1 if operation $l \in \mathcal{O}$ is concluded at or after time $t \in \mathcal{T}$ and set to 0 otherwise;
ρ_t: Total instant power required at time $t \in \mathcal{T}$;
C: Makespan, total time required to process all operations;
E: Total energy consumed to process all productions.

$$\text{Minimize:}\{C,\ E\} \tag{1}$$

Subject to:

$$C \geq c_l, \qquad \forall l \in \mathcal{O}, \tag{2}$$

$$E = \sum_{k,l \in \mathcal{O}} \sum_{v \in \mathcal{V}_l} \rho_l^v \tau_l^v x_{kl}^v + \sum_{l \in \mathcal{O}} \sum_{v \in \mathcal{V}_l} \rho_l^v \tau_l^v w_{lm_l}^v, \tag{3}$$

$$\sum_{v \in \mathcal{V}_l} w_{lm_l}^v + \sum_{k \in \mathcal{O}} \sum_{v \in \mathcal{V}_l} x_{kl}^v = 1, \qquad \forall l \in \mathcal{O}, \tag{4}$$

$$\sum_{k \in \mathcal{O}} \sum_{v \in \mathcal{V}_l} x_{lk}^v + z_{lm_l} = 1, \qquad \forall l \in \mathcal{O}, \tag{5}$$

$$\sum_{l \in \mathcal{O}} \sum_{v \in \mathcal{V}_l} w_{lm_l}^v = 1, \qquad \forall m \in \mathcal{I}, \tag{6}$$

$$\sum_{l \in \mathcal{O}} z_{lm_l} = 1, \qquad \forall m \in \mathcal{I}, \tag{7}$$

$$s_l \geq c_{l-1}, \qquad \forall l \in \mathcal{O} \setminus \mathcal{O}', \tag{8}$$

$$s_l \geq c_k + M\left(\sum_{v \in \mathcal{V}_l} x_{kl}^v - 1\right), \qquad \forall k, l \in \mathcal{O}, \tag{9}$$

$$c_l = s_l + \sum_{k \in \mathcal{O}} \sum_{v \in \mathcal{V}_l} x_{kl}^v \tau_l^v + \sum_{v \in \mathcal{V}_l} w_{lm_l}^v \tau_l^v, \qquad \forall l \in \mathcal{O}, \tag{10}$$

$$\rho_t = \sum_{l \in \mathcal{O}} \sum_{v \in \mathcal{V}_l} \theta_{lt}^v \rho_l^v, \qquad \forall t \in \mathcal{T}, \tag{11}$$

$$My_{lt} \geq t - (s_l - 1), \qquad \forall l \in \mathcal{O},\ t \in \mathcal{T}, \tag{12}$$

$$Mq_{lt} \geq (c_l + 1) - t, \qquad \forall l \in \mathcal{O},\ t \in \mathcal{T}, \tag{13}$$

$$2 + \theta_{lt}^v \geq y_{lt} + q_{lt} + \left(\sum_{k \in \mathcal{O}} x_{kl}^v + w_{lm_l}^v\right), \qquad \forall l \in \mathcal{O}, v \in \mathcal{V}_l, t \in \mathcal{T}, \tag{14}$$

$$\rho_t \leq \mathcal{P}_t^{limit}, \qquad \forall t \in \mathcal{T}, \tag{15}$$

$$C, E, c_l, s_l, \rho_t \geq 0, \qquad \forall l \in \mathcal{O}, t \in \mathcal{T}, \tag{16}$$

$$x_{kl}^v, w_{lm_l}^v, z_{lm_l}, y_{lt}, q_{lt}, \theta_{lt}^v \in \{0, 1\}, \qquad \forall k, l \in \mathcal{O}, v \in \mathcal{V}_l, t \in \mathcal{T}. \tag{17}$$

As stated by Equation (1), the objective function minimizes two objectives, namely the makespan and the total energy consumption. The makespan is given by the completion time of the last job (constraints (2)), while the total energy consumption is given by the summation, over all operations, of the energy consumed to process the operations (Equation (3)).

Constraints (4) and (5) ensure, respectively, that each operation is either the first operation of a machine or it is preceded by another operation and that each operation is either the last operation of a machine or is followed by another operation. Each machine has exactly one first operation and one last operation as enforced by constraints (6) and (7), respectively.

Constraints (8)–(10) set the starting and completion time of the operations. An operation can only be started after the completion of the previous operation of the same job (constraints (8)) and the completion of the previous operation processed on the same machine (constraints (9)). Finally, the operation completion time is given by adding the operation processing time to its starting time (constraints (10)).

Peak power constraints are enforced through constraints (11)–(15). In constraint (11), the instant power required at time t is calculated by adding the required instant power (to process the operations at the chosen speed) of all operations being processed at time t. An operation is being processed at time t if it has been started at or before time t and is concluded at or after time t, this is determined by constraints (12) and (13), respectively. Hence, the operations being processed at speed v at time t are determined in constraints (14) as the ones that have been started at or before time t, concluded at or after time t and processed at speed v. Finally, constraints (15) ensure that

the instant power required at any time is at most its availability. Lastly, constraints (16) and (17) specify the variables' domain.

3. The proposed multi-objective BRKGA

BRKGAs have been successfully proposed and implemented to find good quality solutions in several applications, (e.g., Gonçalves et al., 2016; Brandão et al., 2017; Andrade et al., 2021; Kummer et al., 2024; Lima et al., 2024; Homayouni et al., 2023). The multi-objective and multi-population BRKGA proposed here (mop-BRKGA) is based on that of Fontes and Homayouni (2023) and adapts the BRKGA framework, first proposed in Gonçalves and Resende (2011). The mop-BRKGA evolves $\Omega + \Pi$ populations: each of the Ω populations considers one of the Ω objectives, while each of the Π populations considers simultaneously all Ω objectives. The initial populations are randomly generated. Additionally, for the Ω populations, in 20% of the initial solutions we set the processing speeds to the maximum or the minimum value depending on whether the objective under consideration is the makespan (C) or the total energy consumption (E), respectively.

3.1. The evolutionary strategy

The evolutionary strategy used in this study is similar for all populations. At each generation, a new population is created by combining three sets of solutions: (i) the set of elite solutions of the current population, which consists of the best n_e solutions; (ii) the set of offspring solutions, which consists of n_o solutions generated through crossover (see subsection 3.4); and (iii) the set of mutant solutions, which consists of n_m solutions randomly generated in the same manner as the initial population.

For each of the $\omega \in \Omega$ single-objective populations, the set of elite solutions N_e^{ω} comprises the best n_e solutions based on the population-specific fitness function. Regarding each of the $\pi \in \Pi$ multi-objective populations, the set of elite solutions N_e^{π} comprises the best n_e solutions chosen from a pool of solutions consisting of the best n_e solutions of its current population and the best n_e solutions of the current population of each of the Ω single-objective populations. (Repeated solutions are removed.) Additionally, every so often, the pool of solutions also includes the best n_e solutions of the other multi-objective populations. Therefore, the pool of solutions may have up to $(\Omega + \Pi) \times n_e$ solutions. However, only the best n_e solutions of the pool are of interest. Regarding multi-objective populations, to find the best solutions we resort to the non-dominated ranking procedure to rank them and the crowding distance to sort the solutions of the same rank (Deb et al., 2002).

The evolutionary process, that is, the process of creating and selecting the best solutions is repeated for a predetermined number of generations $G_{\max}$. In the end, a final pool of solutions is created by joining the best n_e solutions of each of the Ω single-objective populations and the non-dominated solutions of the Π multi-objective populations. After removing all repeated solutions and all dominated solutions, we are left with a set of Pareto solutions that provide an approximation to the Pareto optimal front.

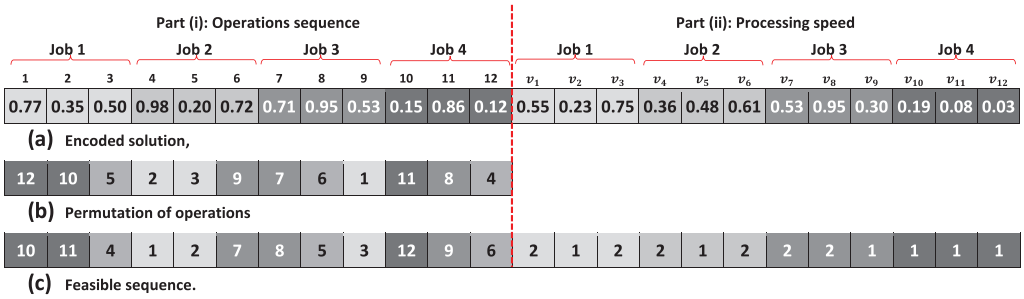

Fig. 1. Solution encoding and decoding for an $EEJSP^{pp}_{sa}$ with four jobs with three operations each that can be processed at one of the two/three available speeds.

Table 1
Data of the first problem instance (Yin et al., 2017)

Jobs	Operations	Machines	$\tau_l^v(\rho_l^v)$			Jobs	Operations	Machines	$\tau_l^v(\rho_l^v)$		
			$v=1$	$v=2$	$v=3$				$v=1$	$v=2$	$v=3$
1	1	1	14(2)	7(5)	6(8)	3	7	1	6(2)	5(5)	4(8)
	2	2	7(2)	6(4)			8	4	5(3)	4(5)	
	3	3	8(4)	7(7)			9	5	13(5)	12(8)	8(10)
2	4	5	7(5)	6(8)	5(10)	4	10	3	9(4)	6(7)	
	5	2	4(2)	3(4)			11	4	6(3)	3(5)	
	6	4	7(3)	4(5)			12	1	5(2)	3(5)	2(8)

3.2. Encoding and decoding

A solution to the $EEJSP^{pp}_{sa}$ is represented by a two-part vector, each of which with N elements, where $N = \sum_{j\in\mathcal{J}} n_j$ is the total number of operations to be scheduled across all jobs, $\mathcal{J}$ is the set of all jobs and n_j is the number of operations of job j. The first part of the vector provides a sequence of operations, indicating the order in which the operations are to be scheduled, and the second part assigns a processing speed to each operation.

Each element of the vector is represented by a random key (RK), which is a real number in the interval [0, 1]. By using this encoding scheme, we are able to represent a wide range of potential solutions to the $EEJSP^{pp}_{sa}$. Additionally, the use of random keys enables the efficient exploration of the solution space as well as the quick identification of high-quality solutions. Figure 1a illustrates the proposed encoding for an $EEJSP^{pp}_{sa}$ involving four jobs with three operations each that can be processed at one of the available speeds (two or three, depending on the operation under consideration). The full data for the problem instance (Yin et al., 2017) is provided in Table 1.

The decoding procedure for the first N elements of the vector resorts to the smallest position value rule (Bean, 1994) to obtain a permutation representation of the random keys, where the elements are sorted in ascending order. This provides a sequence of operations, as shown in part (i) of Fig. 1b. The permutation is then converted into a feasible sequence of operations by sorting the operations of each job in ascending order to ensure the precedence constraints, as illustrated in

Algorithm 1. Earliest priority scheduling

1: Initialize the ordered set of unscheduled operations $\mathcal{U}$ and the set of associated processing speeds $\mathcal{V}_l$;
2: **while** $\mathcal{U} \neq \emptyset$ **do**
3: Consider the first operation in $\mathcal{U}$, l, and its processing speed v_l;
4: Determine the EPS_l and set the current time $t_1 \leftarrow \text{EPS}_l$;
5: **while** Operation l is not scheduled **do**
6: **if** $\rho_l^{v_l} \leq \mathcal{P}_t^{\text{limit}}, \ \forall t \in \{t_1, \ldots t_1 + \tau_l^{v_l}\}$ **then**
7: Schedule the operation, $s_l \leftarrow t_1$, $c_l \leftarrow t_1 + \tau_l^{v_l}$, and $c_{m_l} \leftarrow c_l$;
8: Update the remaining power, $\mathcal{P}_t^{\text{limit}} \leftarrow \mathcal{P}_t^{\text{limit}} - \rho_l^v, \ \forall t \in \{s_l, \ldots c_l\}$;
9: **else**
10: Update current time, $t_1 \leftarrow \min_{k \in \mathcal{O} \setminus \mathcal{U}} \{c_k : c_k > t_1\}$;
11: **end if**
12: **end while**
13: Update the set of unscheduled operations, $\mathcal{U} \leftarrow \mathcal{U} \setminus \{l\}$;
14: **end while**
15: Return the schedule and the values of C and E.

part (i) of Fig. 1c. The second part of the chromosome assigns a processing speed to each operation, which is the smallest integer greater than or equal to $\text{RK}_l \times |\mathcal{V}_l|$, where $|\mathcal{V}_l|$ is the number of speeds available to process operation l, as depicted in part (ii) of Fig. 1c. The decoding procedure generates only solutions that satisfy the precedence constraints of the operations, significantly reducing the search space and eliminating the need for repair mechanisms.

3.3. Scheduling strategy

After determining the operations' sequence and processing speeds, we still have to determine the (processing) starting time of each operation. An operation, say l, can only be processed once the previous operation of the same job $(l-1)$ is completed, which happens at time c_{l-1}, and the required machine (m_l) is available, which happens when the previous operation on the same machine has been completed (c_{m_l}). Thus, the earliest possible starting (EPS) time of an operation is either c_{m_l}, if l is the first operation of a job, or $\max\{c_{l-1}, \ c_{m_l}\}$, otherwise. However, not all operations may be started at their EPS, since the (remaining) available power at the EPS might not be enough to process them at the chosen speed. Hence, to determine the actual starting time of each operation we propose an earliest priority scheduling scheme that (i) considers the operations in the order given by the first part of the chromosome and (ii) starts each operation as soon as possible. This scheduling strategy is articulated in Algorithm 1.

The ordered set of unscheduled operations $\mathcal{U}$ is initialized with all operations in the order determined by the decoding scheme previously described. The operations are then considered, one at a time, in the given order, and for each operation its EPS is calculated (lines 3 and 4). If possible, that is, if there is enough available power, the operation is scheduled to start at its EPS; otherwise, we search for the closest time with enough power to schedule it (lines 5–12). This procedure is repeated until all operations have been scheduled. Note that whenever an operation cannot be scheduled

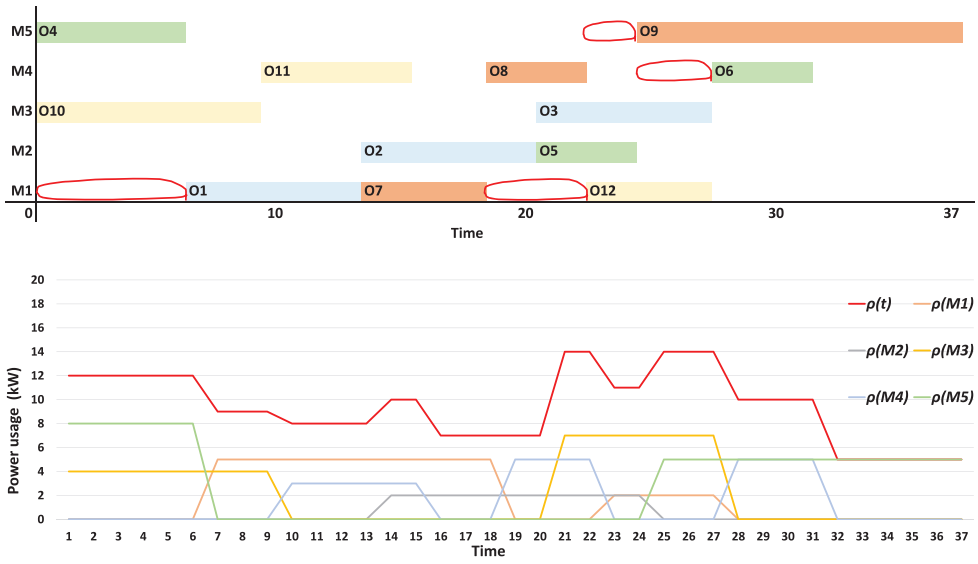

Fig. 2. Gantt chart solution associated with the operations' sequence and speeds provided in Fig. 1, while enforcing the 15 kW power limit and the corresponding power profile.

at its EPS due to insufficient power availability, we only check future time periods at which an operation is completed since only at those some power is released (line 10).

Let us now illustrate the complete solution obtained after applying the decoding procedure and the scheduling strategy just described for the problem instance in Table 1 imposing a peak power limit of 15 kW. In Table 1, for each operation of each job, we provide information on the processing machine as well as the processing time (in minutes) and power consumption (in kW) required to process it at each of the available speeds. Each operation $l \in \mathcal{O}$ has a constant power consumption ρ_l^v over its duration τ_l^v and an energy consumption given by $\rho_l^v \tau_l^v$, with $v \in \mathcal{V}_l$. We obtain the complete solution, which is represented by the Gantt chart depicted in Fig. 2 by applying the earliest priority scheduling strategy to the decoded chromosome represented in Fig. 1c.

The makespan, that is, the time required to process all operations is 37 minutes and the total energy consumption is $\sum_{l \in \mathcal{O}} \rho_l^{v_l} \frac{\tau_l^{v_l}}{60} = 5.8$ kWh. As it can be seen in Fig. 2, the peak power limit affects the schedule. The power limit prevents four operations (operations 1, 6, 9 and 12) from being scheduled at their earliest possible starting time. Hence, they must be postponed to a time at which there is enough available power; this is depicted by the red circles. Note that although we have set a peak power limit of 15 kW, the power requirements are typically well below that value. Indeed, the peak power demanded ranges from 5 kW (in six time periods) to 14 kW (in three time periods), this is also shown in Fig. 2.

If no power limit is imposed, the makespan comes down from 37 to 32 minutes, about 13.5% lower. However, the peak power demanded goes up to 19 kW, over 26% higher. Furthermore, the power profile shows a much larger variability (see Fig. 3).

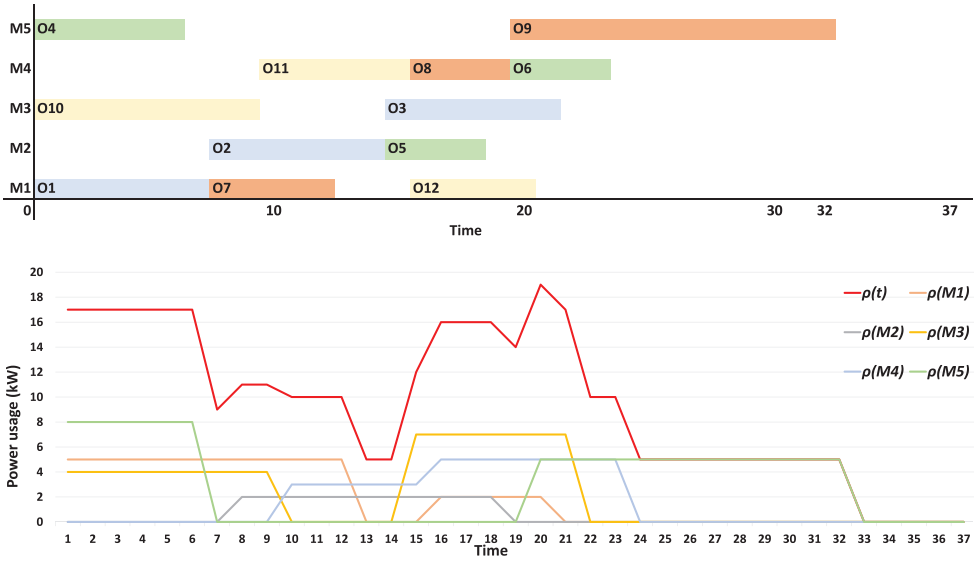

Fig. 3. Gantt chart solution associated with the operations' sequence and speeds provided in Fig. 1 with no power limit and corresponding power profile.

3.4. *Crossover procedure*

The offspring solutions are generated through biased parameterized uniform crossover (Gonçalves and Resende, 2012; Homayouni et al., 2023), for all populations. The parents selection process is biased since one of the parents is always chosen from the set of elite solutions. By doing so, we ensure that one good solution is always chosen, which tends to lead to better and fitter offspring. However, to prevent a solution (or a few solutions) from taking over the entire population in a few generations (i.e., premature convergence), the other parent is chosen from the remaining population.

To generate a new solution, we combine the genetic material of the selected parents by resorting to a biased uniform crossover operator. Hence, we consider one gene at a time and 'flip a coin' to decide whether the offspring inherits the gene from the elite parent or the other parent. Bias is introduced by giving a higher chance to inherit the gene of the elite parent. Hence, the crossover probability p_e is larger than 0.5, which ensures that the offspring will have more genetic material from the elite parent.

Figure 4 illustrates the crossover procedure used considering a 0.7 crossover probability. Once the two parents have been selected, we create a mapping vector by generating a random number between 0 and 1 for each gene. To obtain the offspring we copy the gene from the elite parent whenever the corresponding value of the mapping vector is less than or equal to 0.7 and from the non-elite parent otherwise. In the example given, 15 genes are taken from the elite parent and nine from the non-elite one; as expected the offspring resembles the elite parent the most.

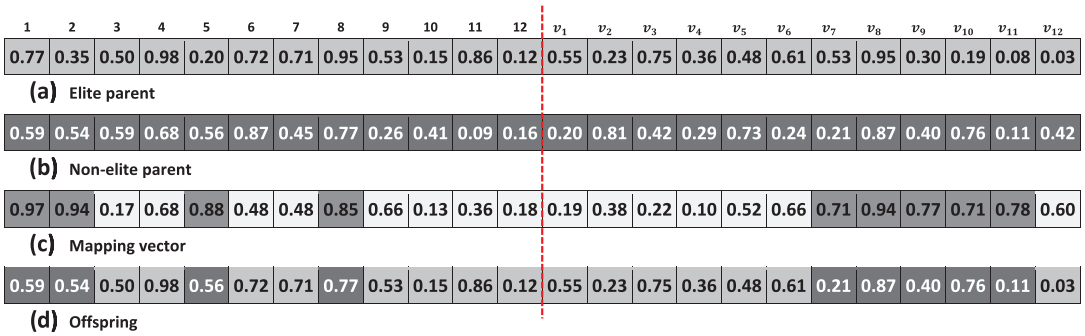

Fig. 4. Crossover procedure: obtaining an offspring solution by resorting to the parameterized crossover with a 0.7 crossover probability.

4. Computational experiments

We address the JSP with speed-adjustable machines and peak power constraints while minimizing both the makespan and the energy consumption. The JSP instances proposed in Salido et al. (2016) and Yin et al. (2017), which have all the required data, are used in our computational experiments. Both works propose a single-objective genetic algorithm that finds solutions to the JSP with speed-adjustable machines while minimizing the normalized weighted sum of the makespan and the energy consumption in Salido et al. (2016) and of the makespan, the energy consumption and the noise emissions in Yin et al. (2017).

In total, we consider 33 problem instances. The 30 instances proposed in Salido et al. (2016) all have three jobs and machines that can be operated at one of three speeds. These instances can be divided into three groups with 10 instances each: in the first group (Sal01–Sal10) all jobs have five operations that are to be processed on three machines, in the second group (Sal11–Sal20) all jobs have 10 operations that are to be processed on seven machines and in the third group (Sal21–Sal30) all jobs have 25 operations that are to be processed on three machines. The other three instances are designated Yin01, Yin02 and Yin03 and have four, 10 and 20 jobs with a total of 12, 40 and 60 operations, respectively. Instances Yin01 and Yin03 have five machines each, while instance Yin02 has six machines. Some machines have two processing speeds, while others have three (all instances can be downloaded from https://fastmanufacturingproject.wordpress.com/problem-instances).

The MILP model was solved using Gurobi® 10.0. and the mop-BRKGA was coded in Python® 3.8. All computational experiments were carried out on a 3.20-GHz Intel® Core™ i7-8700 PC with 24 GB RAM.

We were able to find solutions for all small-sized instances using both the MILP model and the mop-BRKGA. However, the MILP model could only be solved to optimality for medium-sized instances when no peak power constraints were imposed. For the large-sized instances, the MILP model could not find any feasible solutions within a reasonable time, even without considering peak power constraints. It is worth noting that both the number of decision variables and the number of constraints increase with problem instance size. Furthermore, since the magnitude of the total processing time increases, the number of time instants also increases, which in turn leads to a further increase of both the number of decision variables and the number of constraints.

The MILP model was solved by employing a strategy that combines the 'lexicographic' and the ϵ-constraint methods, as proposed in Fontes et al. (2023). First, we obtain optimal boundary solutions by solving two single-objective MILP models: One model (model A) optimizes the makespan, and the other model (model B) optimizes the energy consumption. Second, each of these models is solved again, this time while optimizing one of the objectives we must ensure the previously found optimal value for the other. Then, we solve a series of single-objective pairs of MILP models (models A and B): model A optimizes the makespan with an additional constraint imposing a decreasing upper bound on the energy consumption followed by model B that optimizes the energy consumption while enforcing the makespan found by model A.

The mop-BRKGA was run with the problem-independent parameters set according to literature recommendations as follows: the number of elite solutions to be passed onto the next population was set to $n_e = 0.2P$, the number of mutants introduced into the next population to $n_m = 0.1P$, and the crossover probability to $p_e = 0.7$ (see, e.g., Gonçalves and Resende, 2011; Fontes and Gonçalves, 2013; Roque et al., 2014). The problem-dependent parameters we set based on our previous experience are as follows: population size $P = 20N$, number of generations $G_{\max} = 300$, number of bi-objective populations $\Pi = 2$ and number of generations after which the multi-objective populations exchange the best solutions $g_{ex} = 30$ (Fontes and Gonçalves, 2013; Fontes et al., 2023; Homayouni et al., 2023).

4.1. Performance evaluation

To infer the quality of the mop-BRKGA solutions, we use performance indicators of three groups, namely, convergence, distribution and spread (Audet et al., 2021) and cardinality. Convergence indicators measure the degree of proximity between the true Pareto front (PF*) and its approximation (PF). In this study, we used the modified generation distance GD^+ that provides information on the average distance between each element of PF and its closest neighbour in PF* and is calculated as in Equation (18).

$$GD^+(\text{PF}) = \frac{1}{n}\sum_{i=1}^{n} \min_{j\in \text{PF}^*} \sqrt{\left(\max\left\{C_i - C_j, 0\right\}\right)^2 + \left(\max\left\{E_i - E_j, 0\right\}\right)^2}. \tag{18}$$

Among the distribution and spread indicators, we chose the spread Δ as it captures both the properties of spread and uniformity that indicate the variation of the distance between consecutive elements of PF as well as the extent of PF. This indicator is particularly interesting for bi-objective problems and is calculated as

$$\Delta(\text{PF}) = \frac{d_u + d_b + \sum_{i=1}^{n-1} \left|d_i - \bar{d}\right|}{d_u + d_b + \bar{d}(n-1)}, \tag{19}$$

where n is the number of solutions in PF; d_i is the Euclidean distance between the objective function values of two consecutive solutions; $\bar{d}$ is the average distance, over all d_i distances; and d_u and d_b are the Euclidean distances between the two extreme solutions of PF (i.e., the boundary solutions obtained by the mop-BRKGA) and those of PF* (i.e., the boundary solutions obtained by the

Table 2
Results: small-sized problem instances when no power limit is imposed

Instances			MILP						mop-BRKGA						Performance indicators			
Ins	J-N-M	$\mathcal{P}_t^{limit}$	C^*	E	C	E^*	|PF*|	T	C^*	E	C	E^*	|PF|	$\overline{\text{T}}$	$\overline{\delta_C}$	$\overline{\delta_E}$	$\overline{GD^+}$	$\overline{\Delta}$
Yin01	4-12-5	21	22	5.81	35	4.82	8	26.8	22	5.81	35	4.82	8	45.4	0.0	0.0	0.0000	0.4452
Sal01	3-15-3	266	34	112	54	79	17	1683.5	34	112	54	79	17	44.1	0.0	0.0	0.0280	0.2844
Sal02	3-15-3	362	46	177	79	116	29	5455.0	46	177	79	116	29	44.4	0.0	0.0	0.0153	0.4371
Sal03	3-15-3	258	39	122	63	73	23	4708.1	39	122	63	73	23	44.7	0.0	0.0	0.0171	0.4022
Sal04	3-15-3	248	39	127	65	70	25	2840.4	39	127	65	70	25	45.3	0.0	0.0	0.0384	0.3547
Sal05	3-15-3	337	34	127	54	80	20	2820.7	34	127	54	80	20	44.9	0.0	0.0	0.0117	0.3868
Sal06	3-15-3	222	45	119	72	88	21	6872.6	45	119	72	88	21	45.1	0.0	0.0	0.0061	0.3657
Sal07	3-15-3	295	38	107	60	69	20	1853.7	38	107	60	69	20	44.3	0.0	0.0	0.0200	0.3953
Sal08	3-15-3	268	40	128	63	68	23	4737.0	40	128	63	68	23	44.7	0.0	0.0	0.0209	0.3030
Sal09	3-15-3	354	49	150	75	105	26	9415.5	49	150	75	105	26	45.1	0.0	0.0	0.0175	0.3231
Sal10	3-15-3	319	46	163	73	96	27	11137.0	46	163	73	96	27	44.6	0.0	0.0	0.0308	0.3020
Mean			39.3	121.6	63.0	77.2	21.7	4686.4	39.3	121.6	63.0	77.2	21.7	44.8	0.0	0.0	0.0187	0.3636

MILP model). Regarding cardinality indicators, we consider the total number of non-dominated solutions generated by the mop-BRKGA (|PF|) and by the MILP model (|PF*|). We also calculate the average percentage deviation between the values of the MILP boundary solutions and those of the mop-BRKGA (see Equations (20) and (21)).

$$\delta_C = 100 \times \sqrt{\left(\frac{C^*_{GA} - C^*_{MILP}}{C^*_{MILP}}\right)^2 + \left(\frac{E_{GA} - E_{MILP}}{E_{MILP}}\right)^2}, \tag{20}$$

$$\delta_E = 100 \times \sqrt{\left(\frac{C_{GA} - C_{MILP}}{C_{MILP}}\right)^2 + \left(\frac{E^*_{GA} - E^*_{MILP}}{E^*_{MILP}}\right)^2}. \tag{21}$$

To capture the stochastic nature of our proposed solution approach, the mop-BRKGA was run 10 times for each problem instance. The values of the performance indicators are averaged over the 10 runs. Additionally, we provide the results for the best boundary solutions obtained by MILP and mop-BRKGA for each problem instance.

4.2. Results for $EEJSP^{pp}_{sa}$

First, we solved all instances while disregarding the power limit and determined the peak (i.e., largest instant) power consumption (PP) over the planning period. Then, the instances are solved again considering that only 80% of PP is available, that is, $\mathcal{P}_t^{limit} = 0.8 \times PP, \ \ \forall t \in \mathcal{T}$. A summary of the results for small-sized instances is reported in Tables 2 and 3. These tables have four parts: the first part, which has three columns, reports the problem instances characteristics (J, number of jobs; N, total number of operations; M, number of machines; and $\mathcal{P}_t^{limit}$, instant power limit); the second and third parts, which have six columns each, report the Pareto boundary solutions (optimal makespan C^* and corresponding energy consumption E and optimal energy consumption

Table 3
Results: small-sized problem instances when a limit of 80% of the maximum power is imposed

Instances			MILP						mop-BRKGA						Performance indicators			
Ins	J-N-M	$\mathcal{P}_t^{limit}$	C^*	E	C	E^*	|PF*|	T	C^*	E	C	E^*	|PF|	$\overline{\mathrm{T}}$	$\overline{\delta_C}$	$\overline{\delta_E}$	$\overline{GD^+}$	$\overline{\Delta}$
Yin01	4-12-5	16	23	5.97	35	4.82	7	28.0	24	5.65	35	4.82	6	123.2	6.9	0.0	0.0278	0.5969
Sal01	3-15-3	200	37	101	54	79	14	1211.8	38	101	54	79	14	88.7	3.5	0.0	0.0522	0.3380
Sal02	3-15-3	281	48	165	79	116	27	6533.1	48	165	79	116	27	90.4	2.1	0.0	0.0126	0.4484
Sal03	3-15-3	201	43	111	63	73	21	5294.2	43	115	63	73	21	88.4	3.6	0.0	0.0265	0.3849
Sal04	3-15-3	168	44	105	65	70	20	3541.1	44	105	65	70	20	89.6	1.1	0.0	0.0359	0.3169
Sal05	3-15-3	237	35	118	54	80	19	3476.7	35	118	54	80	19	86.7	0.0	0.0	0.0249	0.3275
Sal06	3-15-3	173	47	122	72	88	19	6445.7	48	114	72	88	18	88.5	5.3	0.0	0.0108	0.4901
Sal07	3-15-3	226	39	102	60	69	19	2044.5	39	102	60	69	18	88.8	4.4	0.0	0.0099	0.4416
Sal08	3-15-3	206	43	117	63	68	20	4752.5	43	117	63	68	20	87.9	0.0	0.0	0.0171	0.2878
Sal09	3-15-3	258	50	149	75	105	25	12446.1	50	149	75	105	25	88.9	0.0	0.0	0.0101	0.3009
Sal10	3-15-3	253	48	153	73	96	25	10141.9	49	152	73	96	24	88.1	3.0	0.0	0.0367	0.3438
Mean			41.5	113.5	63.0	77.2	19.6	5083.2	41.9	113.1	63.0	77.2	19.3	91.7	2.7	0.0	0.0240	0.3888

E^* and corresponding makespan C), the number of Pareto solutions |PF| and the computational time for the MILP and the mop-BRKGA, respectively; and the last part, reports the mop-BRKGA performance, that is, the average value of the four performance indicators ($\overline{\delta_C}$, $\overline{\delta_E}$, $\overline{GD^+}$ and $\overline{\Delta}$) over the 10 runs. It is worthwhile to notice that the MILP model runs multiple times to find the non-dominated solutions in PF*. Therefore, for each instance, the reported computational time refers to the total running time required to find all non-dominated solutions. In contrast, the mop-BRKGA finds the entire PF at once, resulting in a computational time equivalent to its running time.

Regarding the solutions found when no power limit is imposed, the PFs found by the mop-BRKGA are well populated and provide good approximations to the PF*s found by solving the MILP model. The average number of non-dominated solutions found by the mop-BRKGA, over the 10 runs of each of the 11 small-sized instances, is the same as that of the MILP model. As it can be seen from the results reported in Table 2, the mop-BRKGA found all optimal boundary solutions (two for each problem instance). Moreover, the Pareto fronts found by the mop-BRKGA are quite close to the MILP one, as can be seen by the reported values of the modified generation distance (a measure of proximity). Furthermore, these fronts are also well spread and uniform as shown by the low values obtained for the spread. The mop-BRKGA takes significantly less computation time; on average, it takes about only 1% of the computation time required by the MILP model. This shows that the mop-BRKGA not only provides highly accurate approximations of the PF*s but it also does so very quickly. It should be noted that in these instances all operations can be scheduled at their earliest feasible times; hence, the mop-BRKGA does not make use of the earliest priority scheduling algorithm (Algorithm 1).

Table 3 shows the results when the available instant power is restricted to 80% of the peak power used when no limits are imposed. However, as it can be seen in Table 3, the instant peak power usage is actually lower than the power available. For example, in Sal04, although the available instant power is $0.8 \times 248 = 198$ kW, the power used has a peak value of 168 kW. On average, although the available power is 80% of that reported in Table 2, the peak power demand is just about 75% of that value, ranging from under 68% to under 79% of the original value. The best makespan

Table 4
Results: Medium-sized problem instances without power limits and when a limit of 80% of the maximum power is imposed

Instances			No peak power constraint								80% of PP is available			
			MILP				mop-BRKGA				mop-BRKGA			
Ins	J-N-M	$\mathcal{P}_t^{limit}$	C^*	E	C	E^*	C^*	E	C	E^*	C^*	E	C	E^*
Yin02	10-40-6	38	40	18.31	56	17.73	40	18.31	56	17.73	43	18.54	56	17.73
Sal11	3-30-7	343	645	2192	1056	1309	645	2192	1056	1309	656	2292	1056	1309
Sal12	3-30-7	380	620	2597	969	1542	620	2597	969	1542	647	2554	969	1542
Sal13	3-30-7	388	729	3295	1205	2162	729	3295	1205	2162	756	3065	1205	2162
Sal14	3-30-7	440	527	2530	804	1396	527	2530	804	1396	536	2468	804	1396
Sal15	3-30-7	339	638	2656	1082	1485	638	2656	1082	1485	682	2411	1082	1485
Sal16	3-30-7	293	693	2672	1145	1722	693	2672	1145	1722	781	2521	1145	1722
Sal17	3-30-7	375	588	1951	894	1301	588	1951	894	1301	588	2018	894	1301
Sal18	3-30-7	394	630	2957	989	1818	630	2957	989	1818	651	2808	989	1818
Sal19	3-30-7	409	585	2514	960	1595	585	2514	960	1595	593	2480	960	1595
Sal20	3-30-7	326	604	2296	959	1384	604	2296	959	1384	629	2126	959	1384
Mean			572.6	2334.4	919.9	1430.2	572.6	2334.4	919.9	1430.2	596.5	2251.0	919.9	1430.2

increases slightly; on average, it is under 6% larger, while, at the same time, the corresponding energy consumption decreases, on average, by just under 6%, ranging from about 3% larger to 17% lower. Furthermore, when the energy consumption is the main consideration, an average, peak power reduction of about 75% is accomplished at no cost as both the production time and the production energy are the same. The consideration of peak power constraints appears to increase the difficulty of solving the problem to optimality, as indicated by the longer computational times reported for the MILP model. On average, peak power constraints result in an 8.5% increase in computational time. Introducing peak power constraints and incorporating the execution of the earliest priority scheduling algorithm (Algorithm 1) also leads to a twofold increase in the computational time of the mop-BRKGA. Nevertheless, it does not seem to impact the performance of the mop-BRKGA, except for a slight increase in the values of the average percentage deviation between the best makespan values. The values of the modified generation distance and the spread are slightly larger; however, they remain very low.

In the presence of peak power constraints, the MILP model was unable to find any feasible solution for medium-sized instances. However, if no peak power constraints are considered, the MILP model can be solved to optimality. Table 4 reports the boundary solutions obtained by both the MILP model and the mop-BRKGA without peak power constraints as well as the solutions found by the mop-BRKGA when only 80% of the previously used peak power is available. The mop-BRKGA is able to find boundary solutions for all medium-sized instances.

As it can be seen, a reduction of about 20% of the power availability leads to an average increase in the minimum makespan C^* of about 4% while accomplishing, at the same time, a reduction of the corresponding energy consumption E of almost 4%. On the other hand, the minimum energy consumption E^* and corresponding makespan C remain the same. On average, the mop-BRKGA solves medium-sized problems in approximately 62.5 seconds when there are no peak power constraints and 310 seconds when such constraints are imposed. This suggests that Algorithm 1 may

Table 5
Results: Large-sized problem instances without power limits and when a limit of 80% of the maximum power is imposed

Instances			No peak power constraint					80% of *PP* is available				
Ins	J-N-M	$\mathcal{P}_t^{limit}$	C^*	E	C	E^*	|PF|	C^*	E	C	E^*	|PF|
Yin03	20-60-5	35	62	22.86	111	20.96	31	62	23.29	111	20.96	20
Sal21	3-75-3	473	1524	6191	2502	3542	56	1649	6067	2502	3542	48
Sal22	3-75-3	367	1996	7390	3065	4458	67	2139	7013	3100	4458	60
Sal23	3-75-3	425	1678	6506	2810	3469	63	1782	6926	2810	3469	58
Sal24	3-75-3	455	1684	6538	2884	3760	53	1721	6550	2922	3760	65
Sal25	3-75-3	373	1611	6413	2641	3891	47	1715	6104	2628	3891	46
Sal26	3-75-3	476	1530	6207	2488	3739	52	1549	6228	2505	3739	40
Sal27	3-75-3	453	1507	5988	2544	3159	58	1563	5654	2547	3159	57
Sal28	3-75-3	477	1607	6782	2625	4001	71	1634	6630	2625	4001	54
Sal29	3-75-3	401	1814	7087	3125	4056	54	1853	7270	3125	4056	45
Sal30	3-75-3	375	1853	6885	3173	4203	52	1931	6621	3173	4203	70
Mean			1533.3	6000.9	2542.5	3481.7	54.9	1599.8	5916.9	2549.8	3481.7	51.2

be quite time consuming, but it is still reasonable for this class of problem instances. Nevertheless, the variability of the running time across instances is negligible.

For large-sized instances, the MILP model could not find any feasible solutions, regardless of the peak power constraints. Therefore, Table 5 summarizes the results obtained by the mop-BRKGA with and without the peak power constraints. The power availability reduction of 20% results in a 6.3% average increase in the minimum production time and simultaneously in a 2.5% decrease in the corresponding energy consumption. As before, power reduction does not have any effect on the minimum energy consumption and corresponding makespan. Regarding the cardinality of Pareto fronts, a small decrease is observed in the presence of peak power constraints; mainly due to the fact that the solutions associated with the smallest makespan values are not feasible. When dealing with large-sized instances, the average computational time is around 102 seconds with no power availability limits and 630 seconds when limited to 80% of the original required peak power.

4.3. Results for $EEJSP_{sa}$

When no power limit is imposed, the problem under study is comparable to the $EEJSP_{sa}$ for which the benchmark instances were solved by Salido et al. (2016) using a multi-objective GA (moGA) and constraint programming (CP) and by Yin et al. (2017) using a genetic algorithm (GA). Both works employed a weighted summation approach and reported the C and E values under different weight assumptions. Table 6 reports the average boundary solutions obtained by the MILP model and by the mop-BRKGA, over 10 problem instances (i.e., Sal01–10, Sal11–20 and Sal21–30), since this is the way in which the CP and the moGA results are reported in Salido et al. (2016). Figure 5 depicts the Pareto fronts obtained by the mop-BRKGA, by the GA (Yin et al., 2017), and, when available, the true Pareto front found by the MILP model for instances Yin01–03. As it can be seen, the mop-BRKGA is capable of finding very good solutions and it outperforms the GA, the CP and the moGA approaches.

Table 6
The boundary solutions for benchmark problem instances when no power limit is imposed

	Sal01–10				Sal11–20				Sal21–30			
Approach	C^*	E	C	E^*	C^*	E	C	E^*	C^*	E	C	E^*
MILP	41.0	132.9	65.8	84.4	625.9	2566.0	1006.3	1571.4	–	–	–	–
mop-BRKGA	41.0	133.2	65.8	84.4	625.9	2566.0	1006.3	1571.4	1680.4	6598.7	2785.7	3827.8
CP	41.0	143.1	71.4	84.4	625.9	2664.1	1088.4	1571.4	1673.4	6732.2	3160.0	3827.1
moGA	41.0	152.0	65.8	84.4	625.9	2935.1	1006.3	1571.4	1697.0	7088.5	2888.8	3827.1

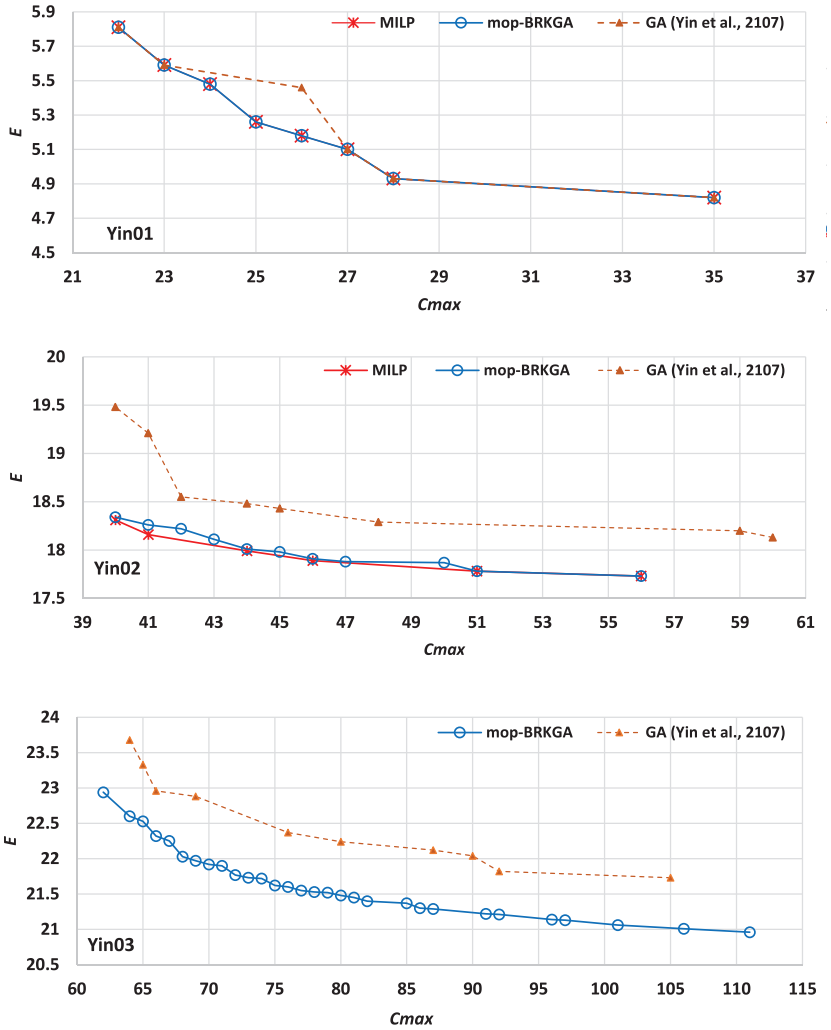

Fig. 5. Pareto front solutions for problem instances Yin01–03 when no power limit is imposed.

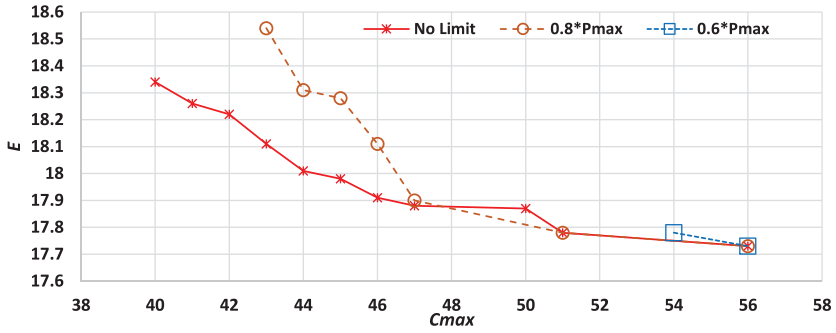

Fig. 6. Impact of the power limit on the Pareto fronts obtained by the mop-BRKGA for instance Yin02.

4.4. *Impact of considering peak power constraints and managerial insights*

Power demand and power consumption profiles have been ignored by academia, in favour of energy consumption or energy costs. However, instant power demand is a main cost driver regarding industrial consumers' contracts. Such contracts, usually, include a charge for the peak power contracted in addition to the energy charge. Furthermore, it is, quite often, impossible or prohibitively expensive to go beyond the power limit contracted. Another important factor is the heavy investment costs required to provide the electric grid with the ability to satisfy surges in power demand, which in turn leads to higher power and energy costs. Hence, providing a tool to industrial companies that allows them to manage power demand is of uttermost importance.

The results reported in Table 2 have shown that a 20% reduction in the power availability leads to a reduction in the peak power usage of about 25%. Additionally, it could be seen that such reduction implies just slightly larger values for the minimum makespan. However, the corresponding energy consumption decreases. On the other hand, it could be observed that the minimum energy consumption values are not impacted by the peak power reduction. Furthermore, the corresponding production times (makespans) are almost the same.

In this section, in order to better analyse the impacts of imposing peak power constraints we solved a subset of four problem instances (Yin02, Sal01, Sal10 and Sal20) under three different scenarios: (i) no power limit, (ii) a 20% reduction of the peak power availability and (iii) a 40% reduction of the peak power availability. The Pareto fronts obtained by the mop-BRKGA for each of the instances considering the above three scenarios are provided in Figs. 6–9.

The impacts of the power reduction in Yin02 are more visible, which does not come as a surprise since this is a small instance. Note that, if a 40% reduction is imposed, unless the operations are processed at the lowest speed, only one or two operations can be done simultaneously; despite having six machines available. From the Pareto fronts obtained, it can be seen that for larger makespans, it is possible to process all operations with similar time and energy requirements, regardless of the power constraints. On the other hand, for shorter makespan values to compensate for the power reduction a larger energy consumption is required. Furthermore, in the presence of a 40% reduction of the available peak power, not many Pareto solutions can be found.

In Sal01, the impacts are less drastic. The Pareto fronts obtained with no power limit and with a 20% peak power reduction are quite similar; however, in the latter case, it is not possible to find

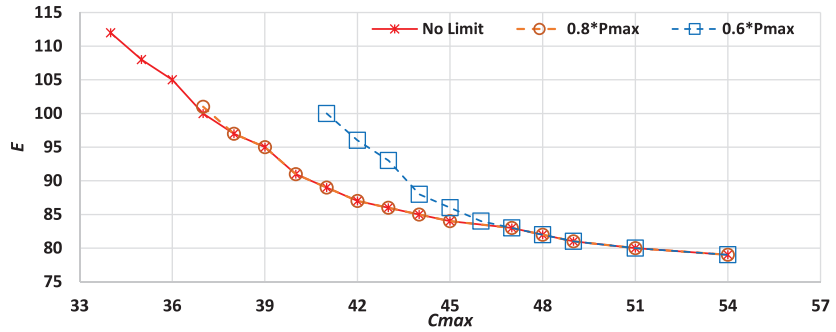

Fig. 7. Impact of the power limit on the Pareto front obtained by the mop-BRKGA for instance Sal01.

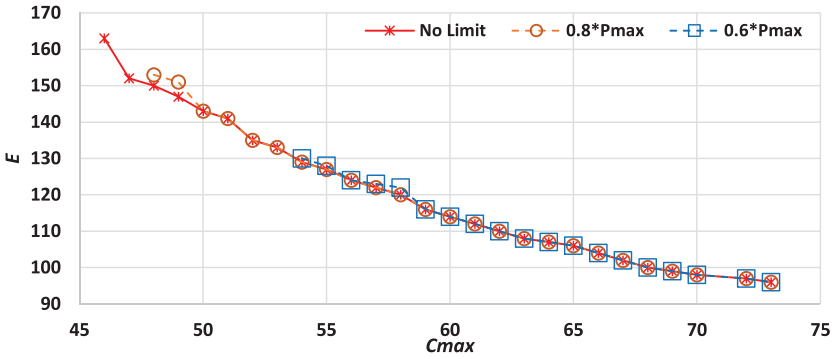

Fig. 8. Impact of the power limit on the Pareto front obtained by the mop-BRKGA for instance Sal10.

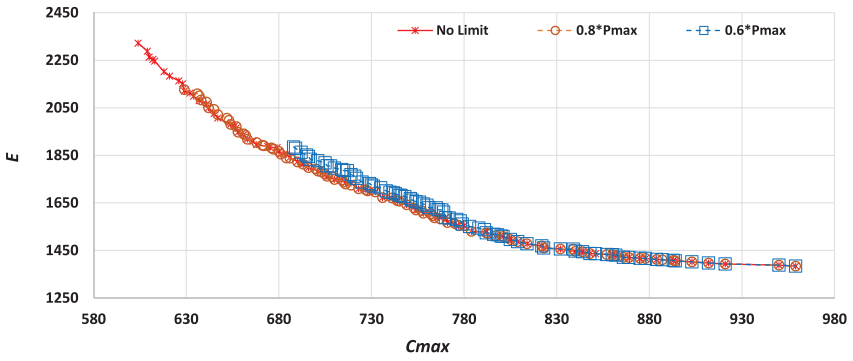

Fig. 9. Impact of the power limit on the Pareto front obtained by the mop-BRKGA for instance Sal20.

solutions for the smallest makespan values. In the presence of a 40% peak power reduction, for the lowest attainable makespan values the energy consumption is quite larger; nevertheless, this increase is quickly reduced as the makespan values increase. In Sal10 and Sal20, the impacts are only noticeable for the smallest makespan values (left part of the Pareto front) since some of them are not possible to achieve in the presence of the peak power reduction.

Regarding instance Sal20, as expected, the conclusions are slightly different since it involves seven machines. Under no peak power constraints, up to seven operations can be done simultaneously. However, when the instant peak power usage is reduced by 20% and by 40%, the number of simultaneous operations decreases and thus the minimum makespan increases from 604 to 629 and 688, respectively.

From the Pareto fronts obtained, it can be seen that managers can clearly choose strategies favouring peak power reductions. Furthermore, if the main concern is energy consumption, that is, its minimization, then power reductions can be achieved at no cost both regarding production time and energy consumption (bottom right side of the Pareto fronts). On the other hand, if the production time is the main issue, then power reduction implies an increase in the production time but, at the same time, provides a decrease in energy consumption.

5. Conclusions

In this paper, a JSP with peak power limits is studied. The objective is to find a production power profile that respects the power availability and minimizes both the energy consumption and the makespan. To find and smoothen the overall power consumption profile, one has to coordinate the activities of individual machines. Given that each machine can process each operation at one of the several machine speeds available, in addition to the sequence of operations on each machine, one must also determine both the starting time and the processing speed for each operation. There are two main reasons to include peak power constraints. On the one hand, the energy providers charge customers not only for the amount of energy consumed but also for the highest rate at which energy is consumed during any given interval of time, that is, the instant peak power demand. This encourages customers to manage their energy usage more efficiently and avoid excessive usage during peak periods. On the other hand, power consumption limits are becoming more and more important due to the increasing penetration of renewables. For example, solar power production depends on the availability of sunlight, and wind power production depends on wind strength. If renewable energy sources cannot provide the necessary (peak) power for a specific time period, then the energy grid may need to rely on other sources to meet the demand, which can be costly and environmentally unfriendly. To avoid this, energy providers may impose peak power limits to ensure that renewable energy sources are utilized efficiently and effectively.

We minimize the total energy consumption together with the makespan; since otherwise, the processing speed would be adjusted to the available power, that is, the maximum speed that satisfies the power constraint. Therefore, we obtain several trade-off solutions that show that it is possible to reduce the makespan or the energy consumed, without impacting much of the other.

The problem is formulated as a MILP model. This model is able to find true Pareto fronts for small- and medium-sized instances with up to four jobs, 40 operations, and five machines when no peak power constraints are imposed. Additionally, true Pareto fronts were obtained for small-sized instances when a 20% reduction is imposed on the instant peak power demand. To address this problem, a multi-objective and multi-population biased random key genetic algorithm (mop-BRKGA) was developed. This algorithm uses a heuristic scheduling approach to generate only feasible schedules that observe the instant peak power constraints. The mop-BRKGA algorithm finds very good trade-off solutions, with approximated Pareto fronts that align with the true Pareto

fronts obtained by the MILP model and are well distributed over the search space. An analysis of the impact of imposing instant peak power constraints shows that the minimum production time increases with the decrease of the instant peak power available. Additionally, a larger energy consumption is required to obtain similar makespan values, as expected. Nevertheless, it was observed that the slightly larger values for the minimum makespan (4–6%, on average) were accomplished while being able to decrease the required energy consumption (also 4–6%, on average). Additionally, it could also be observed that the minimum energy consumption values are not impacted by the peak power reduction. Furthermore, the corresponding production times (makespans) are almost the same. These findings have important implications for manufacturing companies that need to adopt power-flexible production schedules due to stricter regulations regarding environmental impacts and the intermittent nature of power availability associated with renewable energies. The results of this study provide decision and policymakers with valuable insights regarding power requirements, particularly in what concerns the implications of imposing or negotiating power consumption limits.

This work can be extended in several ways. Regarding machines, one may include machine states, that is, machines that when not processing can be switched into one of several possible states (e.g., idle, power saving, off) and thus have a different power and energy consumption. Regarding operations, it can be considered that the power needed to process each operation varies depending on the processing stage. Regarding power availability, it could be considered time-dependent, which is more representative of renewable energy availability, or even not known in advance, which would lead to real-time optimization.

Acknowledgments

Open access funding enabled and organized by Projekt DEAL.

References

Afsar, S., Palacios, J.J., Puente, J., Vela, C.R., González-Rodríguez, I., 2022. Multi-objective enhanced memetic algorithm for green job shop scheduling with uncertain times. *Swarm and Evolutionary Computation* 68, 101016.

Andrade, C.E., Toso, R.F., Gonçalves, J.F., Resende, M.G., 2021. The multi-parent biased random-key genetic algorithm with implicit path-relinking and its real-world applications. *European Journal of Operational Research* 289, 1, 17–30.

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

Bean, J.C., 1994. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing* 6, 2, 154–160.

Borges, Y.G., Schouery, R.C., Miyazawa, F.K., Granelli, F., da Fonseca, N.L., Melo, L.P., 2020. Smart energy pricing for demand-side management in renewable energy smart grids. *International Transactions in Operational Research* 27, 6, 2760–2784.

Bowman, E.H., 1959. The schedule-sequencing problem. *Operations Research* 7, 5, 621–624.

Brandão, J.S., Noronha, T.F., Resende, M.G., Ribeiro, C.C., 2017. A biased random-key genetic algorithm for scheduling heterogeneous multi-round systems. *International Transactions in Operational Research* 24, 5, 1061–1077.

Carlucci, D., Renna, P., Materi, S., 2023. A job-shop scheduling decision-making model for sustainable production planning with power constraint. *IEEE Transactions on Engineering Management* 70, 5, 1923–1932.

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

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

EIA, 2021. U.S. Energy Information Administration: the use of energy explained - energy used in industry. Available at: https://www.eia.gov/energyexplained/use-of-energy/industry.php (Accessed 31 July 2023).

EUROSTAT, 2022a. EU's industries dependent on electricity and natural gas. Available at: https://ec.europa.eu/eurostat/web/products-eurostat-news/w/DDN-20221202-2 (Accessed 31 July 2023).

EUROSTAT, 2022b. Renewable energy on the rise: 37% of EU's electricity. Available at: https://ec.europa.eu/eurostat/web/products-eurostat-news/-/ddn-20220126-1 (Accessed 31 July 2023).

Fang, K., Uhan, N., Zhao, F., Sutherland, J.W., 2011. A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. *Journal of Manufacturing Systems* 30, 4, 234–240.

Fang, K., Uhan, N.A., Zhao, F., Sutherland, J.W., 2013. Flow shop scheduling with peak power consumption constraints. *Annals of Operations Research* 206, 115–145.

Fernandes, J.M., Homayouni, S.M., Fontes, D.B., 2022. Energy-efficient scheduling in job shop manufacturing systems: a literature review. *Sustainability* 14, 10, 6264.

Fisher, H., Thompson, G., 1963. Probabilistic learning combinations of local job-shop scheduling rules. In Muth, J. and Thompson, G. (eds), *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, NJ, pp. 225–2517.

Fontes, D.B., Gonçalves, J.F., 2013. A multi-population hybrid biased random key genetic algorithm for hop-constrained trees in nonlinear cost flow networks. *Optimization Letters* 7, 1303–1324.

Fontes, D.B., Homayouni, S.M., 2023. A bi-objective multi-population biased random key genetic algorithm for joint scheduling quay cranes and speed adjustable vehicles in container terminals. *Flexible Services and Manufacturing Journal* 35, 241–268.

Fontes, D.B., Homayouni, S.M., Fernandes, J.M., 2023. Energy-efficient job shop scheduling problem with transport resources considering speed adjustable resources. *International Journal of Production Research* 1, 1–25.

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

Gonçalves, J.F., Resende, M.G., 2011. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics* 17, 5, 487–525.

Gonçalves, J.F., Resende, M.G., 2012. A parallel multi-population biased random-key genetic algorithm for a container loading problem. *Computers & Operations Research* 39, 2, 179–190.

Gonçalves, J.F., Resende, M.G., Costa, M.D., 2016. A biased random-key genetic algorithm for the minimization of open stacks problem. *International Transactions in Operational Research* 23, 1-2, 25–46.

Gondran, M., Kemmoe, S., Lamy, D., Tchernev, N., 2020. Bi-objective optimisation approaches to job-shop problem with power requirements. *Expert Systems with Applications* 162, 113753.

Gonzalez, M.A., Rasconi, R., Oddi, A., 2022. Metaheuristics for multiobjective optimization in energy-efficient job shops. *Engineering Applications of Artificial Intelligence* 115, 105263.

Homayouni, S.M., Fontes, D.B., Gonçalves, J.F., 2023. A multistart biased random key genetic algorithm for the flexible job shop scheduling problem with transportation. *International Transactions in Operational Research* 30, 2, 688–716.

Kawaguchi, S., Fukuyama, Y., 2016. Reactive tabu search for job-shop scheduling problems considering peak shift of electric power energy consumption. In *2016 IEEE Region 10 Conference (TENCON)*, IEEE, pp. 3406–3409.

Kemmoe, S., Lamy, D., Tchernev, N., 2017. Job-shop like manufacturing system with variable power threshold and operations with power requirements. *International Journal of Production Research* 55, 20, 6011–6032.

Kemmoé-Tchomté, S., Lamy, D., Tchernev, N., 2015. Job-shop like manufacturing system with time dependent energy threshold and operations with peak consumption. In *Advances in Production Management Systems: Innovative Production Management Towards Sustainable Growth: IFIP WG 5.7 International Conference, APMS 2015, Tokyo, Japan, September 7-9, 2015, Proceedings, Part I*, Springer, Berlin, pp. 617–624.

Kummer, A.F., de Araújo, O.C., Buriol, L.S., Resende, M.G., 2024. A biased random-key genetic algorithm for the home health care problem. *International Transactions in Operational Research* 31, 3, 1859–1889.

Lawrence, S., 1984. Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (supplement). Graduate School of Industrial Administration, Carnegie-Mellon University.

Lima, A., Aquino, A.L., Nogueira, B., Pinheiro, R.G., 2024. A matheuristic approach for the minimum broadcast time problem using a biased random-key genetic algorithm. *International Transactions in Operational Research* 31, 1, 246–273.

Manne, A.S., 1960. On the job-shop scheduling problem. *Operations Research* 8, 2, 219–223.

Márquez, C.R., Ribeiro, C.C., 2022. Shop scheduling in manufacturing environments: a review. *International Transactions in Operational Research* 29, 6, 3237–3293.

Masmoudi, O., Delorme, X., Gianessi, P., 2019. Job-shop scheduling problem with energy consideration. *International Journal of Production Economics* 216, 12–22.

NIST, 2020. National Institute of Standards and Technology: Sustainable manufacturing is smart manufacturing. Available at: https://www.nist.gov/blogs/sustainable-manufacturing-smart-manufacturing (Accessed 31 July 2023).

Prins, C., 2009. *A GRASP × Evolutionary Local Search Hybrid for the Vehicle Routing Problem*. Springer Berlin Heidelberg, Berlin, pp. 35–53.

Roque, L.A., Fontes, D.B., Fontes, F.A., 2014. A hybrid biased random key genetic algorithm approach for the unit commitment problem. *Journal of Combinatorial Optimization* 28, 140–166.

Roy, B., Sussmann, B., 1964. Les problemes d'ordonnancement avec contraintes disjonctives. *Note ds* 9.

Salido, M.A., Escamilla, J., Giret, A., Barber, F., 2016. A genetic algorithm for energy-efficiency in job-shop scheduling. *The International Journal of Advanced Manufacturing Technology* 85, 5, 1303–1314.

Tang, D., Dai, M., 2015. Energy-efficient approach to minimizing the energy consumption in an extended job-shop scheduling problem. *Chinese Journal of Mechanical Engineering* 28, 5, 1048–1055.

Wang, Y., et al., 2012. A new hybrid genetic algorithm for job shop scheduling problem. *Computers & Operations Research* 39, 10, 2291–2299.

Weinert, N., Chiotellis, S., Seliger, G., 2011. Methodology for planning and operating energy-efficient production systems. *CIRP Annals* 60, 1, 41–44.

Xiong, H., Shi, S., Ren, D., Hu, J., 2022. A survey of job shop scheduling problem: the types and models. *Computers & Operations Research* 142, 105731.

Yin, L., Li, X., Gao, L., Lu, C., Zhang, Z., 2017. Energy-efficient job shop scheduling problem with variable spindle speed using a novel multi-objective algorithm. *Advances in Mechanical Engineering* 9, 4, 1–21.