Line balancing and task scheduling to minimise power peak of reconfigurable manufacturing systems

Xavier Delorme, Paolo Gianessi

Line balancing and task scheduling to minimise power peak of reconfigurable manufacturing systems[1]

Xavier Delorme[a] and Paolo Gianessi[a]

[a]Mines Saint-Étienne, Univ. Clermont Auvergne, INP Clermont Auvergne, CNRS, UMR 6158 LIMOS, F-42023 Saint-Étienne France (email: delorme@emse.fr, paolo.gianessi@emse.fr)

ABSTRACT

Energy efficiency has become a major concern for manufacturing systems, due to industry being the largest user of scarce, finite energy sources, and also to recent events which have pushed energy prices to alarming levels. In the present Industry 4.0 context, Reconfigurable Manufacturing Systems (RMS) are therefore one of the most promising manufacturing paradigm. In this paper, we investigate the suitability of one of the most common types of RMS, the Parallel-Serial manufacturing line with Crossover, to help minimize the peak of the electric power consumption. More specifically, the balancing of such a production line is studied, so as to integrate power peak minimization from the design stage. Thus, we define the Parallel-Serial-with-Crossover Assembly Line Balancing Problem with Power Peak Minimization, a new combinatorial NP-hard problem. We also propose a suitable time-indexed Integer Linear Program that integrates balancing and scheduling decisions and a matheuristic algorithm designed to tackle large-size instances. Both approaches are tested on a wide set of instances. The computational results show that relevant power peak reductions can be achieved (33% on average), opening up promising perspectives from both algorithmic and managerial viewpoints.

KEYWORDS

Reconfigurable Manufacturing Systems; Line Balancing; Task Scheduling; Power Peak; Integer Linear Programming; Metaheuristics

1. Introduction

Most industrial sectors, from manufacturing to construction, refining and mining, make use of huge amounts of energy for their activities. Industry is thus accountable for the largest share of global total final energy consumption (TFEC): from 36% in 2014 (International Energy Agency 2017), the overall energy consumption of industry has since increased regularly and is now well beyond 50%, according to U.S. Energy Information Administration (2019). The same source indicates that this general trend is set to continue, as energy consumption worldwide is expected to rise by 2050 to nearly 50% more than its current level. Since most production systems mostly make use of scarce and finite energy resources and generate greenhouse gas (GHG) emissions, these figures suggest that energy usage in industry raises major sustainability issues (Koren et al. 2018). Among the major typologies of production and industrial systems, Manufacturing Systems (MS) are certainly among the largest energy consumers (Menghi et al. 2019; Renna and Materi 2021). Hence, the most consistent contribution to reducing GHG emissions in forthcoming years is expected to come from increasingly energy-efficient MSs (Lawrence et al. 2019), making greater use of renewable energy sources (Battaïa et al. 2020). In this respect, electricity usage is no exception. The International Energy Agency (2021) reports that industry was responsible for 22% of total final electricity consumption in 2020. Since this figure is expected to grow to 46% in 2050 due to the electrification of several processes, the conclusions drawn above also remain valid for electric energy.

Moreover, the recent geopolitical events have had further, profound effects on energy (United Nations 2022). On the one hand, countries are being forced to return to increased usage of fossil fuels in the short term (International Energy Agency 2022b), while at the same time energy prices have risen to alarming levels (European Central Bank 2022), electricity included (International Energy Agency 2022a). On the other hand, over the longer term, the present situation could boost the transition towards renewable energy sources (Nature 2022). These major short- and long-term factors are contributing to pushing even more MSs towards energy efficiency.

Not surprisingly, an increased scientific effort has been devoted in recent years to the study of energy-aware MSs and the design of decision-support methods capable of achieving optimised energy management. The European Strategic Energy Technology (SET) Plan of the European Commission stated in

[1]This is an Accepted Manuscript version of the following article, accepted for publication in International Journal of Production Research: "Xavier Delorme, Paolo Gianessi (2023) Line balancing and task scheduling to minimise power peak of reconfigurable manufacturing systems, International Journal of Production Research, DOI:10.1080/00207543.2023.2283568".

its 2021 Implementation Plan that an *enriched process understanding and evaluation of the impact of each decision from early process/plant design phase to production management is essential* (European Commission 2021). To this end, as shown e.g. in Bänsch et al. (2021), objectives of three types are usually aimed at in evaluating energy efficiency in MS: minimisation of the total energy consumption, of the total energy cost w.r.t. to some pricing policy, or of the peak of the overall power consumption. However, most of the scientific output dealing with energy-efficient production systems seems to focus on planning and scheduling problems. As far as the authors are aware, few works seem to consider energy efficiency in the design phase of the system, and even fewer works aim to minimise the production-related power peak. Among the measures mentioned, limiting the peak of electric power used contributes to energy efficiency in two major ways. Firstly, it prevents power outages, which always result in disruption costs for MS, not only in developing countries (Fakih, Ghazalian, and Ghazzawi 2020), but also in the West: for instance in 2021, the percentages of MSs having to cope with electrical failures in France and Germany were 30.2% and 22.1% respectively, with average outage durations of 2.3 and 6.8 hours respectively, and resulting average losses in annual sales of 0.8% and 0.1% respectively (World Bank 2023). Secondly, limiting the electric power peak helps smooth the consumption of energy. This reduces the effects of power supply volatility, which is of strategic importance these days, with greater efforts being put into diversifying the types of energy sources that are used (Carlucci, Renna, and Materi 2021). Limiting the power peak from the design stage is of particular importance for paced flow lines, which represent a large share of MSs, as many important branches of industry rely heavily on them, e.g. the automotive industry, or mass production of consumer electronics, trucks, and airplanes (Boysen, Schulze, and Scholl 2022). Since power consumption results from the overlapping processing of tasks and therefore depend on how those tasks are assigned to workstations and scheduled, in paced lines the power peak will be repeated at each production cycle and it is thus important to have it minimised from the design phase.

In the past decade the industrial sector has also faced the challenge of a major transition, due to market uncertainty, demand volatility and rapidly changing regulatory frameworks, particularly as to energy. The advent of Industry 4.0 is a key factor in rising to the numerous resulting challenges. Among the most interesting possibilities offered by Industry 4.0 are improved manufacturing sustainability and energy efficiency (Ghobakhloo and Fathi 2021; Jamwal et al. 2021). One of the main levers for this is the capacity to track or control energy consumption (Mohamed, Al-Jaroodi, and Lazarova-Molnar 2019). Moreover, Industry 4.0 is pushing the concept of smart, reconfigurable manufacturing machines (Morgan et al. 2021), which can help achieve higher energy efficiency.

This latter aspect has led to greater maturity of Reconfigurable Manufacturing Systems (RMS). In spite of being a relatively recent manufacturing paradigm, RMS are increasingly widely used nowadays, as they are designed to combine the throughput performance of dedicated lines with the flexibility of flexible manufacturing systems. First studied in Koren et al. (1999), RMS attempt to achieve this through six main features: modularity, integrability, convertibility, diagnosability, customisation and scalability (Koren, Wang, and Gu 2017). One of the most common type of RMS is the *Parallel-Serial manufacturing line with Crossover* (see e.g. Freiheit, Shpitalni, and Hu 2004), of which Figure 1 shows an example. In such a system, workstations are organised in a serial line and have multiple parallel identical resources each, while conveyors and gantries allow parts to be moved from one workstation to the next. Parallel-serial lines with crossovers are a general case of single-product paced flow lines, in which produced items, when reaching a workstation, are dispatched by turns on its parallel resources, which function at the same time so as to provide an increase in throughput.

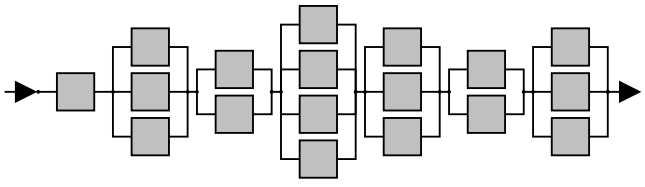

Figure 1. A Parallel-Serial line with 7 workstations, each with 1 to 4 parallel resources. Crossover can occur between any two resources of two consecutive workstations

Motivated by the potential of RMS to help achieve energy efficiency, in this work we study the problem of balancing a paced Parallel-Serial line with Crossover (PSC), with no special equipment to be assigned to the workstations, in order to guarantee a given production pace, with the objective of minimising the peak of the electric power consumption related to processing the production tasks. The balancing

decision process occurs in the design phase of the line: since no equipment choice is considered, energy consumption is not affected. Nor is the economic cost of energy, which will be determined during later planning processes. However, the paced nature of PSC lines requires that if a limitation needs to be imposed on the peak of the electric power used, then it must be sought during the design phase. Hence, in the problem studied here, energy efficiency can be pursued by minimising the peak of the power consumption profile.
In this work, we first provide a definition and full description of the problem of balancing a Parallel-Serial line with Crossover to minimise the electric power peak, which has not been the subject of any scientific effort to date, to the best of our knowledge. Then we introduce an original Integer Linear Programming (ILP) model for the problem. In order to be able to deal with instances of industrial interest, we further develop a matheuristic algorithm. The proposed approaches are tested extensively, and the numerical results are presented and discussed, along with some managerial insights. This work is an extension of Delorme and Gianessi (2022) with a full description of the problem considered, the definition of a matheuristic algorithm, extensive numerical experiments and a detailed analysis of the results and their managerial implications.
In the following, after a short review of the related literature in Section 2, Section 3 describes the problem and presents an Integer Linear Programming model for it, while Section 4 proposes a matheuristic algorithm. Section 5 covers the computational experiments conducted to assess the performance of the two methods, along with the results analysis. Finally, Section 6 gives some conclusions and perspectives.

2. Literature Review

In this section, we present a brief literature review of the scientific works that deal with similar topics to the one presented here. We will focus on papers dealing with RMS and sustainability as a more general context, first; then, we will have a look at articles concerning balancing of RMS, and works that address energy-efficient line balancing problems; finally, we will review works that consider power peak in manufacturing systems.

2.1. Reconfigurable Manufacturing Systems and Sustainability

In addition to their initial purpose, RMS have also attracted attention in the research community due to their great potential to help develop new paradigms for sustainability (Putnik et al. 2013), as well as to improve energy efficiency in production systems (Battaïa et al. 2020). The two reviews Bortolini, Galizia, and Mora (2018) and Haapala et al. (2013) agree in asserting that higher reconfigurability of manufacturing systems leads to better environmental and economic performance, notably by reducing energy consumption. Koren et al. (2018) state that RMS can help achieve sustainable production by allowing improved efficiency in usage of resources, particularly in energy consumption, among others by means of its modularity. Dubey et al. (2017) argue that RMS is one of the manufacturing paradigms that help gain in manufacturing agility while minimising costs and waste, and state that RMS should be designed for sustainability and for energy consumption and environmental impact minimisation. Huang, Badurdeen, and Jawahir (2018) observe that RMS sustainable performance can be improved by suitable tuning of its convertibility level, i.e. the ability to change the functionality or move from one product to another.
We cite some recent works dealing with optimisation problems in RMS that focus on sustainability and energy. In Khezri, Benderbal, and Benyoucef (2021) a multi-objective problem to generate a sustainable process plan in an RMS is addressed: the three criteria to be minimised are a sustainability measure and total production time and cost. A multi-objective Integer Linear Program (ILP) is proposed, as well as two evolutionary approaches, which are tested on a numerical example. Dahmani, Benyoucef, and Mercantini (2022) review recent works concerning sustainability in manufacturing, and particularly in RMS, and discuss the main research challenges to be taken up to achieve energy-efficient RMS, and some open questions to be addressed. Massimi et al. (2020) propose a Mixed-Integer Nonlinear Program (MINLP) to minimise the energy consumption of a RMS based on modularity and integrability. Ghanei and AlGeddawy (2020) propose a Mixed-Integer Linear Program (MILP) to minimise the total cost of energy consumption, system reconfiguration and part transportation between machines facing fluctuating demand and energy prices, along with a genetic algorithm to tackle large- size instances. Finally, the Bilevel Optimization problem of designing the configuration set of RMS and planning the configuration usage is studied in Delorme et al. (2023). The objective is to minimise energy-related costs w.r.t. a given Time-Of-Use pricing scheme while meeting a given demand over a known time horizon.

2.2. Line Balancing and Reconfigurable Manufacturing Systems

Among the various optimisation problems associated with RMS, line balancing problems have been among the most studied.

The term Line Balancing (LB) is used by the research community studying optimisation in production systems to refer to a wide class of design optimisation problems. The name comes from the first such problem to have been studied, the Simple Assembly Line Balancing Problem (SALBP), but ALB problems can also describe other types of industrial environments, e.g. machining or disassembly systems (Battaïa and Dolgui 2013).

In ALBPs, the optimal assignment of the tasks of an assembly process to a set of workstations is sought w.r.t. some criteria in such a way as to comply with the precedence constraints among the tasks. These problems occur during the design of a production system and determine some of its main features, e.g. cycle time and number of workstations. The simplest and best known of these problems, the aforementioned SALBP, focuses on a paced, synchronous, mono-product line, with deterministic, workstation-independent task execution times. In the SALBP-1, the minimum number of workstation m must be determined, given the line cycle time or takt time, c, while it is the inverse in the SALBP-2. Other basic versions are SALBP-E, which aims at minimising the line efficiency $c \cdot m$, and SALBP-F, which assesses the feasibility of a given pair (c, m). SALBP is NP-hard (see e.g. Scholl 1999) and is still a relevant problem, with best known solutions having been yielded in the last decade (Cerqueus and Delorme 2019; Pape 2015; Sewell and Jacobson 2012).

Many ALB problems exist incorporating more generalised settings, industrial constraints of different natures, or more complex line patterns (Boysen, Schulze, and Scholl 2022). To mention some other variants which are closer to the problem studied here, the balancing of assembly lines with parallel resources has been studied e.g. by Buxey (1974) or Pinto, Dannenbring, and Khumawala (1981), while Borisovsky, Delorme, and Dolgui (2014) and Essafi et al. (2010) studied the problem of balancing an RMS with sequence-dependent task setup times – a feature that requires task sequencing decisions to be taken into account. Lahrichi et al. (2021) tackle the Reconfigurable Transfer Line Balancing Problem. Here, tasks must be assigned to the workstations of a Parallel-Serial machining line so as to take sequence-dependent setup times into account, as well as constraints among tasks of various natures, and to minimise the total number of resources used by all the workstations. Finally, in a very recent work, Cerqueus and Delorme (2023) study the problem of balancing a mono-product RMS so as to optimise an original scalability measure; the reliability of this latter is tested by solving the problem by complete enumeration on a set of benchmark instances.

2.3. Line Balancing and Energy

Robotic ALB problems (RALBP) (Borba, Ritt, and Miralles 2018) are of particular interest here as they represent, to the best of our knowledge, one of the rare production system design problems to consider energy efficiency. This is mostly due to robotic assembly lines requiring special equipment to be assigned to workstations to perform tasks: since energy consumption is equipment-dependent, it is often relevant to take it into account and minimise it. An example is Janardhanan, Huang, and Ponnambalam (2015), who seek to minimise the line cycle time and total energy consumption of a robotic assembly line, and propose a Particle Swarm Optimization algorithm. The setting of Li, Tang, and Zhang (2016) is a two-sided robotic assembly line: the authors seek the optimal balancing w.r.t. both energy consumption and takt time. A MILP is presented, then a simulated annealing-based metaheuristic is proposed to seek Pareto-optimal sets. More recently, Zhang et al. (2019) consider the problem of balancing an energy-efficient U-shaped robotic assembly line. A multi-objective MINLP is developed and linearised, its objective functions are suitably aggregated to get a good approximation of the Pareto front, and a multi-objective evolutionary algorithm is proposed.

There are some LB problems other than RALBPs that integrate energy-related criteria exist, however. In Kovalev et al. (2017), the assignment of the tasks to the stations of a paced straight machining line that can produce a set of part types is dealt with. The minimum number of stations is the primary objective, and the overall activation cost of the stations is the second, with the latter considering energy consumption, maintenance, setup and manpower. Fang et al. (2019) focus on the balancing of a mixed-model disassembly line with multi-robotic workstations, with the goal of simultaneously minimising the number of robots used, takt time, overall energy consumption and peak workstation energy consumption. Both Wang et al. (2020) and Liang et al. (2021) aim at minimising energy consumption in the balancing of a disassembly line; in the latter case, the line serves for waste electrical and electronic equipment disposal, and the authors show that energy consumption can be reduced

while pursuing disassembly profit. A bi-objective balancing problem is studied in Liu et al. (2021) in which both workers and tasks must be assigned to workstations, and tasks must be scheduled, under a takt time constraint. Both an overall production cost, taking into account processing and manpower, and energy consumption must be minimised. Zhang, Xu, and Zhang (2020) study the multi-objective problem of balancing a semi-automated assembly line: the additional decision of whether a station must be automated, semi-automated or human-operated must be taken, based on energy consumption, smoothness index and total cost.

As far as the authors are aware, the only problem that attempts to minimise the peak of power consumption in the balancing of an MS is the Simple Assembly Line Balancing Problem with Power Peak Minimization (SALB3PM). In this, tasks must be assigned to workstations and scheduled, so that the power peak due to the overlapping processing of tasks is minimised. First studied in Gianessi, Delorme, and Masmoudi (2019), the SALBP3PM is harder than the SALBP, since intermediate idle time between tasks can be considered, hence adding scheduling decisions to the assignment decisions. A special case of SALB3PM that provides a better fit with manual or semi-automated systems is studied in Lamy, Delorme, and Gianessi (2020), in which scheduling decisions become sequencing decisions since tasks must be triggered at the earliest available starting time. However, the problem studied here differs significantly from the SALB3PM in two major aspects, and actually generalises it. First, in the problem studied here, the power consumption profile is given not only by the overlapping of tasks processed on different workstations, but also on different resources of the same workstation. Second, in a PSC line, the number of resources assigned to each workstation may be subject to decision, e.g. because a set of resources already exist and the purchase of new resources is not under consideration. For these reasons, the model of Gianessi, Delorme, and Masmoudi (2019) does not suit the problem presented in this work.

2.4. Power Peak in Manufacturing Systems

The scarcity of works concerning the peak of the electric power consumption is not restricted to LB problems. According to Bänsch et al. (2021), the scientific works concerning the optimisation of the power peak or its cost amount to no more than 2% of the literature, and in most cases they deal with problems arising at a more tactical or operational stage, especially in scheduling problems. Similar conclusions are drawn by Fernandes, Homayouni, and Fontes (2022), who review the scientific production of the last decade on energy-efficient job-shop and flexible job-shop problems.

In Bruzzone et al. (2012), a flexible flow-shop scheduling problem is tackled and a weighted sum of total tardiness and makespan must be minimised, and a limitation on power peak is enforced. The authors propose an exact approach based on a time-indexed MILP and a local search-based matheuristic. Fang et al. (2013) study a flow-shop scheduling problem in which energy consumption, makespan and carbon footprint are minimised, with an additional constraint on the power peak. Artigues, Lopez, and Haït (2013) introduce a parallel machines scheduling problem where a limitation is imposed on the use of electric power.

Kawaguchi and Fukuyama (2016) study a job-shop scheduling problem seeking a solution that minimises a weighted sum of the makespan and the electric energy consumption. Since energy consumption is considered per time intervals of 10 minutes, minimising the energy consumption of the most consuming time interval can be assimilated with minimising the peak in the power consumption. Kemmoe, Lamy, and Tchernev (2017) address a job-shop problem in which the power profile of a task is considered as having a peak value at the beginning, followed by a lower consumption value for the remainder of its processing. The minimum makespan is sought, subject to a limitation on the overall power peak over the whole planning horizon. Gondran et al. (2020) takes the work of Kemmoe, Lamy, and Tchernev (2017) one step further and, based on the same model of task power consumption profile, tackles a bi-objective job-shop problem to seek Pareto-optimal solutions w.r.t. the makespan and the negotiable threshold on the overall power peak. Masmoudi et al. (2017) proposes a single-item, lot-sizing problem in a flow-shop system with energy consideration. The objective is to minimise the sum of overall setup, storage and energy-related costs. This latter term accounts for both the price of consumed energy w.r.t. a TOU pricing scheme, and the cost of the allocated peak power, on which a limitation is imposed in addition. Similarly, Masmoudi, Delorme, and Gianessi (2019) study a job-shop problem with energy cost minimisation w.r.t. known TOU tariffs and a limit on both the overall power peak consumption and the makespan. A MILP and a time-indexed ILP are proposed, along with a matheuristic based on the latter and used to provide warmstart solutions for harder instances. Carlucci, Renna, and Materi (2021) consider a job-shop scheduling problem in which makespan is minimised and variable machine

speed are taken into account, along with a variable power limit. Finally, Módos, Šucha, and Hanzálek (2021) study a scheduling problem in a system with parallel dedicated machines, in which a limit is imposed on the energy that can be consumed in fixed metering intervals. A study of the computational complexity of some variants of the problem is presented.

3. Problem Description

This section provides a formal definition of the studied problem, which we refer to as the Parallel-Serial-with-Crossover Assembly Line Balancing Problem with Power Peak Minimization (PSCALB3PM), as well as a time-indexed Integer Linear Program (ILP) inspired from that of Gianessi, Delorme, and Masmoudi (2019) for the SALB3PM. As will be discussed, the PSCALB3PM generalises the SALB3PM.
A Parallel-Serial manufacturing line with Crossover has to be designed to implement a production process. Let m and $\mathcal{M} = \{0...m-1\}$ denote, respectively, the maximum allowed number of workstations and the workstation set. Each workstation $k \in \mathcal{M}$ can be assigned a number r_k of identical resources, r_k being bounded by a maximum number $r_{\max}$. The maximum total number of resources of the line, $\sum_{k\in\mathcal{M}} r_k \leq R_{\max}$, is also subject to a bound $R_{\max}$, with $m \leq R_{\max} \leq m \cdot r_{\max}$. A targeted minimum line throughput is imposed, in the form of a given takt time value, which we denote by c. Given a workstation k' s.t. $r_{k'} > 1$, its resources function at the same time (and not, e.g., by turns) and process the same tasks according to the same schedule, but shifted by multiples of the line takt time. Therefore, the processing times of the assigned tasks can sum up to $r_{k'} \cdot c$, without preventing k' producing an item each c time units, i.e. without compromising the target throughput.
It is of particular importance to point out that this latter aspect is a major difference w.r.t. the SALB3PM. Indeed, the SALB3PM is the particular case of the PSCALB3PM with $r_{\max} = 1$, i.e. that forbids nonserial configurations. On the one hand, tasks repeat with period c, and so does the power consumption profile, as in the SALB3PM for a simple assembly line; on the other hand, however, for each workstation k' s.t. $r_{k'} > 1$, the timespan *virtually* expands to $r_{\max} \cdot c$: therefore the power consumption sums up not only for tasks processed at the same time on different workstations, but also for tasks assigned to the same workstation with multiple resources and whose processings are shifted in such a way as to occur at the same time on different resources.
Let us also denote by $\mathcal{O} = \{0...n-1\}$ the set of the n production tasks of the same process. A task $j \in \mathcal{O}$ is characterised by a processing time t_j and a power consumption value w_j, with both being integer, deterministic and workstation-independent. Precedence constraints exist among the tasks such that given $i, j \in \mathcal{O}$, if i precedes j (which we denote by $i \prec j$), the processing of i must be over before j can begin.
The objective is to decide:

- the number of resources of each workstation $k \in \mathcal{M}$,
- the assignment of tasks to workstations,
- the starting time of tasks,

so that the target takt time is complied with by the workload of all workstations, precedence constraints are met, and the overall power consumption profile, determined by the overlapping execution of tasks on different resources (of one or different workstations), has its peak minimised. The core assignment and scheduling decision of the SALB3PM are found in the PSCALB3PM, which in addition must incorporate structure decisions to shape the parallel-serial line.
In order to represent the scheduling decisions in the PSCALB3PM, we follow the choice of the ILP model of Gianessi, Delorme, and Masmoudi (2019) and make use of time-indexed variables, a very common choice in the literature of scheduling problems (Bowman 1959). We denote by $\mathcal{T} = \{0...r_{\max} \cdot c - 1\}$ the set of the time slots of the augmented timespan $r_{\max} \cdot c$: $\mathcal{T}$ acts as the index set of time-indexed variables, since it allows the time horizon of workstation k to be expressed even in the case in which it virtually expands to $r_k \cdot c$. Moreover, let $\mathcal{T}^j = \{0...r_{\max} \cdot c - t_j\}$ denote the set of candidate starting times of task j, and $\mathcal{T}^j_r = \{0...r \cdot c - t_j\}$ denote the set of possible starting times for any task on a workstation with r resources, $1 \leq r \leq r_{\max}$, and $\mathcal{T}^j_{r_{\max}} = \mathcal{T}^j$.
The following binary decision variables allow the abovementioned decisions of the problem to be represented:

- *assign* variables $X_{j,k}$, with $j \in \mathcal{O}$ and $k \in \mathcal{M}$, and $X_{j,k} = 1 \Leftrightarrow$ task j is assigned to workstation k,
- *trigger* variables $S_{j,t}$, where $j \in \mathcal{O}$ and $t \in \mathcal{T}^j$, and $S_{j,t} = 1 \Leftrightarrow$ task j is triggered at time slot t,
- *resource* variables $R_{k,r}$, with $k \in \mathcal{M}$ and $r \in \{1...r_{\max}\}$, and $R_{k,r} = 1 \Leftrightarrow$ workstation k uses its r-th

resource,

to which we add a non-negative decision variable:

- *power-peak* variable W_{M}, an upper bound on the power consumption peak all along the time horizon $\{0...c-1\}$.

We further introduce for each $j \in \mathcal{O}$ and $t \in \mathcal{T}^j$ the binary decision expression (1):

$$F(j,t) = \sum_{\tau=t-t_j+1}^{t} S_{j,\tau}, \quad F(j,t) = 1 \Leftrightarrow \text{task } j \text{ is } \textit{running} \text{ at time slot } t \tag{1}$$

The correctness of this expression is based on the fact that only one $S_{j,t}$ variable, corresponding to the starting time of j, can take value 1. $F(j,t)$ then takes value 1 if j has been triggered at a time $\tau \in \{t-t_j+1...t\}$.
The proposed model $\mathcal{M}^{PSC}$ for PSCALB3PM is then as follows:

$$\min W_{M} \tag{2}$$

$$\text{s.t.} \sum_{k\in\mathcal{M}} X_{j,k} = 1 \qquad \forall j \in \mathcal{O} \tag{3}$$

$$\sum_{j\in\mathcal{O}} t_j \cdot X_{j,k} \leq c \cdot \sum_{r\in\{1...r_{\max}\}} R_{k,r} \qquad \forall k \in \mathcal{M} \tag{4}$$

$$X_{j,k} \leq \sum_{h\in\mathcal{M}:h\leq k} X_{i,h} \qquad \forall i,j \in \mathcal{O} : i \prec j, k \in \mathcal{M} \tag{5}$$

$$\sum_{t\in\mathcal{T}^j} S_{j,t} = 1 \qquad \forall j \in \mathcal{O} \tag{6}$$

$$X_{j,k} - R_{k,r} \leq \sum_{t\in\mathcal{T}^j_{r-1}} S_{j,t} \qquad \forall j \in \mathcal{O}, k \in \mathcal{M}, r \in \{1...r_{\max}\} \tag{7}$$

$$S_{j,t} \leq \sum_{\tau=0}^{t-t_i} S_{i,\tau} + 2 - X_{i,k} - X_{j,k} \qquad \forall i,j \in \mathcal{O} : i \prec j, k \in \mathcal{M}, t \in \mathcal{T}^j \tag{8}$$

$$X_{i,k} + X_{j,k} + F(i,t) + F(j,t) \leq 3 \qquad \forall i,j \in \mathcal{O} : i < j, k \in \mathcal{M}, t \in \mathcal{T} \tag{9}$$

$$\sum_{j\in\mathcal{O}, r\in\{1...r_{\max}\}} w_j \cdot F(j,(r-1)c+t) \leq W_{M} \quad \forall t \in \{0...c-1\} \tag{10}$$

$$R_{k,r+1} \leq R_{k,r} \qquad \forall k \in \mathcal{M}, r \in \{1...r_{\max}-1\} \tag{11}$$

$$\sum_{k\in\mathcal{M}, r\in\{1...r_{\max}\}} R_{k,r} \leq R_{\max} \tag{12}$$

$$X_{j,k}, S_{j,t}, R_{k,r} \in \{0,1\}, W_{M} \in \mathbb{Z}_{+}$$

Constraints (3) state that each task must be assigned to exactly one workstation. Relations (4) bound the workload of each workstation k, i.e. the sum of the processing times of the tasks assigned to it, to the number of resources assigned to it, i.e. $\sum_{r\in\{1...r_{\max}\}} R_{k,r}$, times the takt time c. Constraints (5) are precedence relations: if task i is a predecessor of task j, then the assignment of j to a workstation, say k, forces i to be assigned either to k or to another, upstream of k. Inequalities (8) enforce that in case they are assigned to the same workstation, then i must be triggered at least t_i slots before j; they are redundant otherwise.
Relations (6) impose the trigger of task j at exactly one of its candidate time slots $\mathcal{T}^j$. Moreover, let k be the station, with r_k resources, to which j is assigned, i.e. $X_{j,k} = 1$: (11) then ensure that $R_{k,1} = ... = R_{k,r_k} = 1$, and consequently (7) forbids starting j at a time $t \geq r_k \cdot c - t_j + 1$. Along with (6), this enforces that the processing of j must start at one of the time slots $\{0...r_k \cdot c - t_j\}$. Constraints (9) prevent two tasks i and j from being processed at the same time on the same workstation: either they are assigned to the same workstation, i.e. $(\exists k \in \mathcal{M})\ X_{i,k} = X_{j,k} = 1$, but then at most one among them can be running at time t, i.e. $F(i,t) + F(j,t) \leq 1$; or their processings overlap at least in part, i.e. $(\exists t \in \mathcal{T})\ F(i,t) = F(j,t) = 1$ but then they must assigned to different workstations, i.e. $(\nexists k \in \mathcal{M})\ X_{i,k} = X_{j,k} = 1$.

Each constraint (10) refers to a time slot t of the *actual* timespan $\{0...c-1\}$ of the line; it considers the sum of the power consumption of all the tasks running at t concurrently on some resource of the (k, r) of the line, i.e. which are running at some time slot $(r-1)c+t$ of the *virtual* schedule of the workstation they are assigned to. In this way, (10) bounds the overall power consumption at time t to W_{M}, which we minimise in (2). Finally, inequality (12) bounds the overall number of resources at $R_{\max}$. The size of the proposed model $\mathcal{M}^{PSC}$, in terms of number of variables NV and of constraints NC, depends on the PSCALB3PM instance size as shown by (13) and (14):

$$\mathrm{NV}(\mathcal{M}^{PSC}) = n \cdot m + n \cdot c \cdot r_{\max} + m \cdot r_{\max} \tag{13}$$

$$\mathrm{NC}(\mathcal{M}^{PSC}) = 2 \cdot n + c + m \cdot (p + r_{\max}) + m \cdot r_{\max} \cdot (n + p \cdot c + \tfrac{n \cdot (n-1) \cdot c}{2}) + 1 \tag{14}$$

where $p = |\{(i, j) : i, j \in \mathcal{O}, i \prec j\}|$ is the number of precedence constraints.
Although inspired by the model of Gianessi, Delorme, and Masmoudi (2019), $\mathcal{M}^{PSC}$ has some major differences, related to the possibility of assigning more than one resource per workstation, which should be underlined. The main and most straightforward difference is the addition of resource variables to model such assignment, as well as the need to limit the overall number of resources by means of (12). Secondly, the total possible workload of a workstation depends on the number of its resources, with a direct impact on balancing decisions, as represented by constraints (4). Of course, since the timespan of a workstation k can be any of $r_k \cdot c$, $r_k \in \{1...r_{\max}\}$, not only a much larger number of trigger variables per task is required, which in itself is an element making the solving harder and not a modelling difference: the decision of whether a set of time slots is available to trigger a task must now be taken, which is the role of constraints (7). Moreover, due to the nature of resource variables, symmetry-breaking constraints (11), although not necessary to model the problem, are preferable to reduce the computational time later. However, the subtlest difference in terms of modelling may be represented by constraints (10) that bound the cumulative power used to W_{M}, since the power consumption, as previously explained, must sum up not only for tasks whose processings overlap as they are assigned to different workstations, but also to the same workstation and shifted in such a way so to occur at the same time on different resources. An example is given in Figure 3, the details of which will be described in depth in Section 3.1. In the right subfigure, workstation $k = 3$ is assigned 2 resources and the subset of tasks $\{5, 4, 6, 7\}$: the latter are scheduled in such a way that the processing of task 5 overlaps with task 4, then with task 6, for respectively 5 and 7 time units. Representing such overlaps is a major modelling contribution. Incidentally, the fact that each constraint (10) must consider a set of contributions whose number is not bounded by m, as it was in the SALB3PM, but by $\sum_k r_k$, which is under decision, can make the solving considerably harder.

3.1. Effects of allowing parallel-serial configurations with crossover

In the following, we try to take a closer look at the implications of allowing parallel-serial configurations with crossover to afford the reader a better grasp of the problem being studied.
Let us consider two of the SALB3PM instances defined in Gianessi, Delorme, and Masmoudi (2019), namely bowman-1 and jaeschke-2. The former has $n = 8$, $m = 5$, $c = 20$, the latter $n = 9$, $m = 3$, $c = 18$. Figure 2 resumes their task features: for each node of the precedence graphs, the processing time is indicated above the node, and the power consumption below.

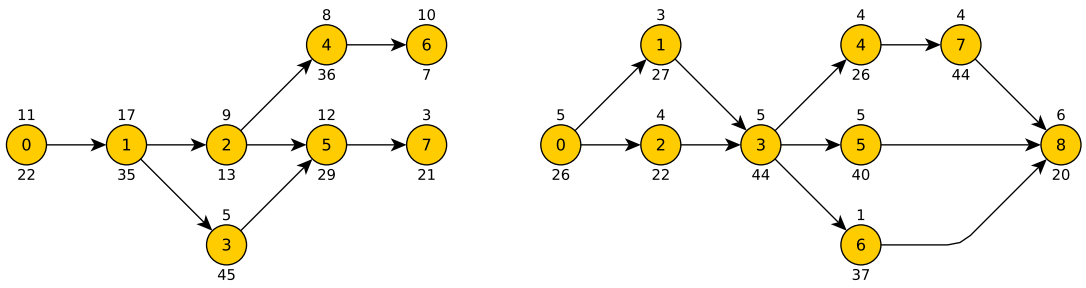

Figure 2. SALB3PM instances bowman-1 (left) and jaeschke-2 (right)

Instance bowman-1 offers an example, depicted in Figure 3, to illustrate how beneficial the adoption of parallel-serial configurations with crossover can be in terms of power peak reduction.

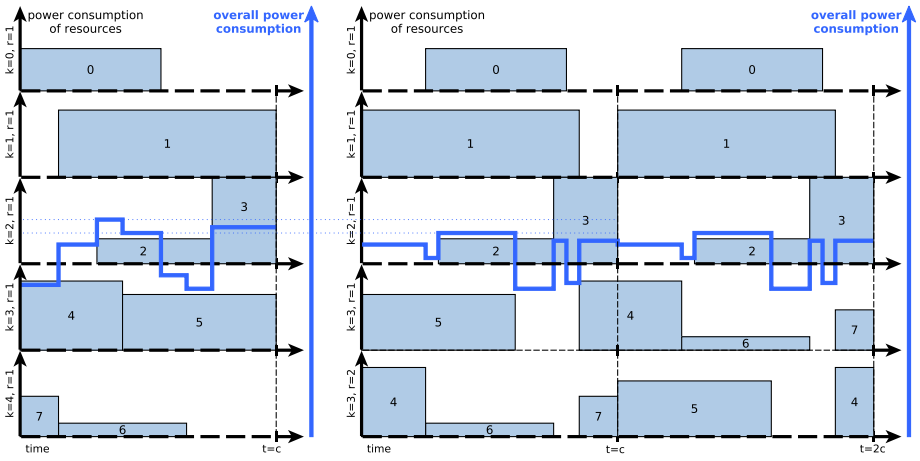

Figure 3. Benefits on power peak of switching to parallel-serial configurations with crossover, bowman-1 instance

This Figure shows via Gantt diagrams the optimal solutions to different PSCALB3PM problems based on bowman-1. Each task is represented by a box whose width and height are proportional to its processing time and power consumption, respectively, while the horizontal position denotes its starting time in the schedule. A separate chart is shown for the schedule of each resource, with thick dotted lines separating the resources of different workstations, and thin ones the resources of one workstation. The overall power consumption profile is highlighted by a thick continuous line. The leftmost subfigure is the optimal solution when $R_{\max} = m = 5$ and $r_{\max} = 1$: in fact, it represents the SALB3PM optimum. In the rightmost subfigure, the same maximum authorised number of 5 resources holds, but $r_{\max} = 2$, meaning that parallel-serial configurations with crossover are allowed and up to 2 resources per workstation can be used. In the solution with $r_{\max} = 1$, the peak results from the overlap of tasks 0, 1, 2, 4 and 6, and its value is $W_M = w_0 + w_1 + w_2 + w_4 + w_6 = 22 + 35 + 13 + 36 + 7 = 113$ power units. When $r_{\max} = 2$, four workstations are used: the first three $k = 0 \ldots 2$ have $r_k = 1$ and their workloads do not exceed the takt time, thus their cycle has period c; the last workstation has two resources working in parallel, i.e. $r_3 = 2$, which have a period $2c$ and process the same tasks, but with schedules shifted by c. This arrangement of the available resources allows the described overlap to be avoided: the peak occurs now when tasks 0, 1, 2, 5 and 6 are running concurrently, decreasing W_M by $w_4 - w_5 = 7$ to 106. Note that assigning two resources to workstation $k = 3$ in the second case allows it to have an augmented timespan of $2c$: the execution of task $j = 4$ can then span over the first and the second half of it – which would obviously have been impossible with $r_{\max} = 1$– and be delayed, so as to avoid the most power-consuming overlap. From a combinatorial viewpoint, this relaxation in scheduling tasks on a workstation with parallel resources, due to having an expanded time horizon of $r_k \cdot c$ instead of multiple compartmentalised timespans of width c, is among the main reasons behind the power peak reduction in parallel-serial configurations with crossover.
Another noteworthy observation is how increasing the number of resources in an existing parallel-serial configuration impacts the peak of the power profile. Figure 4 illustrates this on instance jaeschke-2, showing the optimal solutions when $r_{\max} = 2$ and the overall number of resources increases from $R_{\max} = 3$ (leftmost subfigure) to $R_{\max} = 4$ (rightmost subfigure). Both the Gantt diagrams cover a time extent of $2c$, since in both cases at least one workstation ($k = 0$ for $R_{\max} = 3$, $k = 0$ and $k = 1$ for $R_{\max} = 4$) uses all the $r_{\max} = 2$ resources available. As seen before, the parallel resources of a workstation k s.t. $r_k > 1$ process the same tasks and have the same schedule shifted by c. With $R_{\max} = 3$, the power peak occurs at the overlap of tasks 1, 4 and 8, amounting to $w_1 + w_4 + w_8 = 27 + 26 + 20 = 73$ power units. Once again, adding a resource allows to avoid this overlap, thanks to a less-constrained range of scheduling possibilities: the peak occurs now when tasks 3 and 4, or 0 and 7, overlap, with a consumption of $w_3 + w_4 = w_7 + w_0 = 44 + 26 = 70$.

4. A Matheuristic Approach

As stated before, the PSCALB3PM is NP-hard, and therefore a metaheuristic (see e.g. Glover and Kochenberger 2003; Talbi 2009) can be useful to deal with large-size instances. This is notably the case for instances with large takt time c, which can cause the size of the time-indexed model $\mathcal{M}^{PSC}$ to grow significantly.

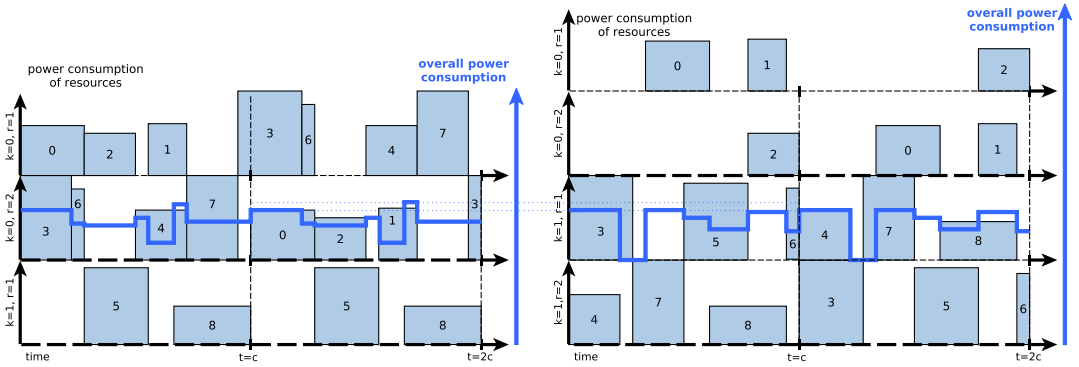

Figure 4. Adding resources to an existing parallel-serial solution can also be beneficial for the power peak, as shown here on jaeschke-2 instance, $r_{\max} = 2$, and $R_{\max}$ going from 3 to 4

In this section, we provide the definition of MS×LS^{PSC}, a matheuristic for the PSCALB3PM, which allows to tackle large-size instances in reasonable computational time. MS×LS^{PSC} takes its name from its structure, i.e. a multi-start (MS) algorithm based on local search (LS), and uses an ILP as decoder. MS×LS^{PSC} performs its search in the space of *solution codings*. A *solution coding* σ is defined as an ordered set of workstations σ_k, each representing the ordered set of the tasks assigned to it, in which each task is assigned to one and only one workstation. For example, the coding of the PSCALB3PM solution of the rightmost subfigure of Figure 3 is given by (15):

$$\sigma = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\} = \{\{0\}, \{1\}, \{2, 3\}, \{5, 4, 6, 7\}\} \tag{15}$$

A solution coding σ for the PSCALB3PM only incorporates the assignment decisions of the problem (represented by variables X in model $\mathcal{M}^{PSC}$). Resource decisions (variables R in model $\mathcal{M}^{PSC}$) can be derived for each σ_k as the minimum number of resources required based on the total workload of the tasks assigned to it. A *decoder* is then needed to set the starting times of tasks and thus compute the power peak. It is noteworthy to point out that the resources required by a solution coding could violate resource bounds $r_{\max}$ and/or $R_{\max}$, thus leading to an infeasible PSCALB3PM solution. Such a solution coding will also be called *infeasible*.
Since a solution coding implicitly defines an overall task sequence, which results from merging the ordered sets associated with workstations in such a way as to preserve both the order among and within the sets themselves, it is useful to define a binary operator on tasks, named *σ-precedence*. Task i is said to *σ-preceed* task j ($i \ll_\sigma j$) if either it is in a lower-index workstation of σ, or it has a lower index in the same workstation. This is shown in (16):

$$\begin{aligned}
&(\forall i, j \in \mathcal{O}, i \neq j)\ i \ll_\sigma j \Leftrightarrow \\
&\quad (\exists k_1, k_2 \in \{0 \ldots |\sigma|-1\}, k_1 < k_2,\ i \in \sigma_{k_1} \wedge j \in \sigma_{k_2}) \vee \\
&\quad (\exists k \in \{0 \ldots |\sigma|-1\}, \lambda_1, \lambda_2 \in \{0 \ldots |\sigma_k|-1\}, \lambda_1 < \lambda_2,\ i = \sigma_k(\lambda_1) \wedge j = \sigma_k(\lambda_2))
\end{aligned} \tag{16}$$

where notation $\sigma_k(\lambda)$ denotes the λ-th task of the ordered set $\sigma_k \in \sigma$. σ-precedence allows to define the search space $\mathcal{D}$ formally, as in (17):

$$\begin{aligned}
\mathcal{D} = \Big\{ \sigma = \{\sigma_0 \ldots \sigma_{|\sigma|-1}\} \mid \textstyle\bigcup_k \sigma_k = \mathcal{O} \wedge \\
(\forall k, k' \in \{0 \ldots |\sigma|-1\}, k \neq k')\ \sigma_k, \sigma_{k'} \subseteq \mathcal{O}, \sigma_k \cap \sigma_{k'} = \varnothing \wedge \\
(\forall i, j \in \mathcal{O})\ i \prec j \Rightarrow i \ll_\sigma j \Big\}
\end{aligned} \tag{17}$$

i.e. as the set of all σ, partitions of $\mathcal{O}$, for which the precedence constraints are respected.
From now on, solution codings of PSCALB3PM solutions will be referred to by using the term *coding* alone, whereas the term *solution* will be used exclusively to refer to PSCALB3PM solutions, i.e. in which starting times of tasks have been decided, and the power peak can be (or has been) evaluated.

While a solution has only one possible coding, a coding can lead to different solutions, depending on the decoder behaviour.
During this section, the solution of the mentioned subfigure in Figure 3 (related to instance bowman-1, on the left of Figure 2), will be used as the reference example, along with the associated coding of (15).

4.1. Notations related to a Coding

Before describing algorithm MS×LS^{PSC} in detail, it is helpful to define some additional notations concerning a coding σ. Let us denote by:

- $|\sigma|$, the number of workstations *used* in σ, i.e. with at least one task;
- $|\sigma_k|$, the number of tasks of workstation $\sigma_k \in \sigma$;
- $\sigma_k(\lambda)$, the λ-th task of the ordered set $\sigma_k \in \sigma$, as previously defined in (16);
- $r_k(\sigma)$, the number of resources of workstation $\sigma_k \in \sigma$, initially computed as $r_k(\sigma) = \lceil \frac{1}{c} \sum_{i \in \sigma_k} t_i \rceil$;
- $R(\sigma)$, the total number of resources in σ, equal to $\sum_{k \in \{0 \dots |\sigma|-1\}} r_k(\sigma)$;
- $\text{pen}(\sigma)$, the *penalty units* associated with a coding, computed as $\max\{0, R(\sigma) - R_{\max}\} + \sum_k \max\{0, r_k(\sigma) - r_{\max}\}$: $\text{pen}(\sigma)$ is greater than 0 if $r_{\max}$ and/or $R_{\max}$ (the maximum number of resources per workstation and for the whole line) are violated and the coding is infeasible, 0 otherwise;
- $\iota_k(\sigma)$, the idle time of workstation k in σ, equal to $\iota_k(\sigma) = r_k(\sigma) \cdot c - \sum_{i \in \sigma_k} t_i$;
- $\mathcal{I}_\nu(\sigma)$, the index of the nonempty workstation in σ which ranks $(\nu+1)$-th by increasing idle time, i.e. $\mathcal{I}_0(\sigma) = \text{argmin}_k\, \iota_k(\sigma)$ is the workstation with minimum idle time, and $(\forall\, \nu > 0)$ $\iota_{\mathcal{I}_{\nu-1}(\sigma)}(\sigma) \leq \iota_{\mathcal{I}_\nu(\sigma)}(\sigma)$;
- $\mathcal{T}_{k,\lambda}$, the set $\{d^e_{k,\lambda}(\sigma) \dots d^l_{k,\lambda}(\sigma)\}$ of the possible trigger times for $\sigma_k(\lambda)$, $d^e_{k,\lambda}(\sigma) = \sum_{\lambda'=0}^{\lambda-1} t_{\sigma_k(\lambda')}$ being the earliest possible value (if all the tasks preceding $\sigma_k(\lambda)$ in σ_k are also scheduled at their earliest possible time) and $d^l_{k,\lambda}(\sigma) = d^e_{k,\lambda}(\sigma) + \iota_k(\sigma)$ the latest possible value (corresponding to the whole idle time $\iota_k(\sigma)$ occurring at the beginning of the schedule of σ_k).

In the coding σ of (15) associated with the reference example, the $|\sigma| = 4$ workstations feature number of assigned tasks $\{|\sigma_k|\} = \{1, 1, 2, 4\}$, workload values $\{\sum_{i \in \sigma_k} t_i\} = \{11, 17, 14, 33\}$ and number of resources $\{r_k(\sigma)\} = \{1, 1, 1, 2\}$; hence, $R(\sigma) = 5$. Since $r_{\max} = 2$ and $R_{\max} = 5$, σ is feasible and thus has $\text{pen}(\sigma) = 0$ penalty units. Idle times are $\{\iota_k(\sigma)\} = \{9, 3, 6, 7\}$, therefore $\{\mathcal{I}_\nu(\sigma)\} = \{1, 2, 3, 0\}$. Finally, earliest/latest possible trigger times of the tasks can be computed, e.g. on $k = 3$ we have $\mathcal{T}_{3,0} = \{0 \dots 7\}$, due to $\iota_3(\sigma) = 7$, and $\mathcal{T}_{3,2} = \{20 \dots 27\}$, due to $t_{\sigma_3(0)} + t_{\sigma_3(1)} = t_5 + t_4 = 20$.

4.2. General structure of MS×LS^{PSC}

Figure 5 outlines the structure of the matheuristic MS×LS^{PSC}, and Figure 6 provides a flowchart diagram for it.
The multi-start (MS) character of MS×LS^{PSC} resides in the main loop (lines 2-17, Figure 5) being executed as many times (runs) as possible within the parameter time limit.
The outline of MS×LS^{PSC} is straightforward. Starting from a PSCALB3PM instance I, and for each each run g, the **Construction()** routine provides a randomly-generated initial coding $\sigma^\star_g$ in two steps:

1. generate a random topological ordering, i.e. a precedence-compliant sequence, of the tasks
2. randomly choose $m - 1$ split point for the generated sequence

The initial coding $\sigma^\star_g$ is then decoded by means of the **Decoder()** routine (line 4, Figure 5), so as to obtain a solution $\boldsymbol{s}^\star_g$, which is the starting point of the local search (LS, lines 5-14, Figure 5).
It is straightforward to see that the random choice of the split points in **Construction()** can cause $\sigma^\star_g$ to be infeasible, as no guarantee is given about its compliance w.r.t. $r_{\max}$ and $R_{\max}$. An infeasible coding $\sigma^\star_g$ generated by the **Construction()** routine is not discarded: instead, after computing the power peak of the corresponding (infeasible) solution $\boldsymbol{s}^\star_g$, **Decoder()** adds a penalty to it. Such penalty is given by an upper bound UB_W on the value of the power peak, multiplied by $\text{pen}(\sigma^\star_g)$, the number of penalty units associated with $\sigma^\star_g$. By doing so, infeasible solutions can be visited during the LS.
At each LS iteration, a stochastic descent is performed. Routine **GetNeighbors()** yields a set of neighbours of the current coding: each is evaluated via **Decoder()**, and the best coding σ' and associated solution $\boldsymbol{s}'$ are retained. If the power peak associated with $\boldsymbol{s}'$ is less than or equal to that of $\boldsymbol{s}^\star_g$, σ' becomes the new current coding.
Note that if the coding initially yielded by **Construction()** is infeasible, the LS can possibly walk through a sequence of infeasible codings. However, once a feasible coding is found, no more infeasible

MS×LS^{PSC}()

```
globals  I: PSCALB3PM instance; UB_W: upper bound on peak power;
globals  b^1_LS, b^2_LS: loop sizes; timeL: time limit;
returns  s*: best solution found;
declare  s*_g, s': local solutions; ϒ: set of codings; σ*_g, σ': local codings;
declare  g, f1, f2: iteration counters;
 1  s* ← ∅ ; g ← 1
 2  while(runtime < timeL)
 3     σ*_g ← Construction(I)
 4     s*_g ← Decoder(σ*_g) ; f1 ← 0 ; f2 ← 0
 5     repeat
 6        f1 ← f1 + 1
 7        ϒ ← GetNeighbors(σ*_g)
 8        σ' ← argmin_{σ∈ϒ} W_M(Decoder(σ))
 9        if(W_M(s') ≤ W_M(s*_g)) //s' previously obtained when decoding σ'
10           if(W_M(s') < W_M(s*_g)) f2 ← 0 else f2 ← f2 + 1
11           σ*_g ← σ'
12        else
13           f2 ← f2 + 1
14     until(f1 = b^1_LS || f2 = b^2_LS || runtime ≥ timeL)
15     if(W_M(s*_g) < W_M(s*)) s* ← s*_g
16     g ← g + 1
17  endwhile
18  return s*
```

Figure 5. Pseudo-code of MS×LS^{PSC}

codings will be accepted, since the power peak of the corresponding infeasible solutions is increased by some multiple of the upper bound UB_W, and hence is always strictly greater than any value attainable by a feasible solution.
The f_1 and f_2 variables keep track of the total number of LS iterations and the number of those without power peak improvement, respectively: LS ends when either f_1 or f_2 exceed the given limits b^1_{LS} or b^2_{LS}, or the time limit timeL is reached.
The neighbourhood on which **GetNeighbors()** is based is described in Section 4.3, whereas Sections 4.4-4.6 delve into the details of routine **Decoder()**.

4.3. Stochastic descent

For each start, MS×LS^{PSC} generates a coding σ and then improves it during the LS. A straightforward neighbourhood $\mathcal{N}$ is considered, which is defined in the following. Similar neighbourhoods can be widely found in the literature, e.g. in Gourgand, Grangeon, and Norre (2007) to build metaheuristics for the SALBP.
The neighbourhood $\mathcal{N}(\sigma)$ of a coding σ can be defined as in (18):

$$\begin{aligned}\mathcal{N}(\sigma) = \Big\{\sigma' \in \mathcal{D}\backslash\{\sigma\} \mid (\exists k_1, k_2 \in \{0...|\sigma|-1\}, \exists i \in \sigma_{k_1}),\\ (\forall k \neq k_1, k_2,\ \sigma'_k = \sigma_k)\ \wedge\ (\forall j \in \sigma_{k_1}\backslash\{i\},\ j \in \sigma'_{k_1})\ \wedge\ (\sigma'_{k_2} = \sigma_{k_2} \cup \{i\})\ \wedge\\ (\forall j_1, j_2 \in \sigma'_{k_1} \cup \sigma_{k_2},\ j_1 \ll_\sigma j_2 \Rightarrow j_1 \ll_{\sigma'} j_2)\Big\} \quad (18)\end{aligned}$$

$\mathcal{N}(\sigma)$ is the set of all codings that can be obtained from σ by removing one task $i \in \mathcal{O}$ from one of its workstations, k_1, and inserting it in another, k_2, of course in precedence-compliant manner. All other tasks are left in the workstation they are assigned to and σ-precedence relations among them are preserved. Since the case $k_1 = k_2$ is possible, σ must be explicitly removed.
Based on the definition of neighbourhood $\mathcal{N}$, each call to **GetNeighbors()** performs the following steps:

(1) a workstation $k \in \{0...|\sigma|-1\}$, and a task assigned to it, $j \in \sigma_k$, are picked randomly;
(2) the boundary insertion positions for task j are determined as follows:

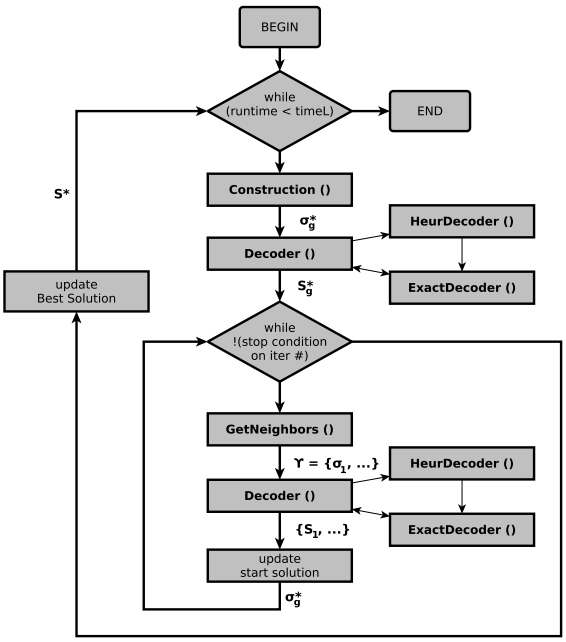

Figure 6. Flowchart diagram of MS×LS^{PSC}

(a) obtain the ordered set $\overline{\sigma}$, defined as in (19), i.e. by merging the sets σ_k in the order defined by σ:

$$\overline{\sigma} = \Big\{\sigma_0(0), \sigma_0(1), ..., \sigma_0(|\sigma_0|-1) \Big\| \sigma_1(0) ... \Big\| \sigma_{|\sigma|-1}(0), ..., \sigma_{|\sigma|-1}(|\sigma_{|\sigma|-1}| - 1) \Big\} \tag{19}$$

in which, for the sake of clarity, workstation boundaries are highlighted;

(b) go backward in $\overline{\sigma}$ down to the first task j_p which is an immediate predecessor of j, i.e. such that $j_p \prec j$;

(c) go forward in $\overline{\sigma}$ up to the first task j_s which is an immediate successor of j, i.e. such that $j \prec j_s$;

(d) find $k_p \leq k$ and λ_p as the workstation and index of j_p, s.t. $j_p = \sigma_{k_p}(\lambda_p)$, and $k_s \geq k$ and λ_s as the workstation and index of j_s, s.t. $j_s = \sigma_{k_s}(\lambda_s)$;

For instance, for the coding of the reference example (see (15)), we get $\overline{\sigma} = \{0 \,\|\, 1 \,\|\, 2, 3 \,\|\, 5, 4, 6, 7\}$; by picking $k = 3$ and task $j = \sigma_3(1) = 4$, the nearest immediate predecessor and successor are respectively tasks $j_p = 2 = \sigma_2(0)$ on $k_p = 2$, and $j_s = 6 = \sigma_3(2)$ on $k_s = 3$, see (20):

$$\Big\{0 \Big\| 1 \Big\| [2], 3 \Big\| 5, \underline{4}, [6], 7\Big\} \tag{20}$$

(3) the set of possible insertion positions, given by (21), is built:

$$\begin{aligned}\{&(\sigma_{k_p}(\lambda_p), \sigma_{k_p}(\lambda_p + 1), k_p), ..., \\ &(\sigma_{k_p}(|\sigma_{k_p}| - 1), \sigma_{k_p+1}(0), k_p), (\sigma_{k_p}(|\sigma_{k_p}| - 1), \sigma_{k_p+1}(0), k_p + 1), ..., \\ &(\sigma_{k_s}(\lambda_s - 1), \sigma_{k_s}(\lambda_s), k_s)\}.\end{aligned} \tag{21}$$

The generic element (j_1, j_2, k') of this set represents the choice of inserting task j between j_1 and j_2 on workstation k', where j_1 and j_2 are consecutive elements in the ordered set $\overline{\sigma}$. In the particular case in which j_1 and j_2 are on different, consecutive workstations (e.g. $j_1 = 3$ and $j_2 = 5$ in (20)), there are two possibilities of inserting j between them, i.e. as the new last element of the workstation of j_1, or as the new first element of that of j_2. Therefore, the set defined in (21) descends from the fact that $j_p \prec j \prec j_s$, and the first available position is after $j_p = \sigma_{k_p}(\lambda_p)$ on k_p, while the last is before $j_s = \sigma_{k_s}(\lambda_s)$ on k_s.

In our example, after picking $k = 3$ and $j = 4 \in \sigma_3$ as in (20), three possible insertion positions exist, yielding as many neighbour codings, namely:

- $(2, 3, k = 2)$, yielding $\{\{0\}, \{1\}, \{[2], \underline{4}, 3\}, \{5, [6], 7\}\}$,

- $(3, 5, k = 2)$, yielding $\{\{0\}, \{1\}, \{[2], 3, \underline{4}\}, \{5, [6], 7\}\}$,
- $(3, 5, k = 3)$, yielding $\{\{0\}, \{1\}, \{[2], 3\}, \{\underline{4}, 5, [6], 7\}\}$.

Of course, when building the set of insertion positions as in (21), the option corresponding to leaving j in its current position is disregarded, e.g. in our example, $(5, 6, k = 3)$.

(4) for each possible insertion position $(j_1^\star, j_2^\star, k^\star)$:
   (a) the task j is removed from set σ_k (and workstation k);
   (b) j is then inserted in set $\sigma_{k^\star}$ (and workstation $k^\star$) after task $j_1^\star$ and/or before task $j_2^\star$;
   (c) the obtained neighbour coding is evaluated by means of **Decoder()**;

As previously discussed (line 8 of MS×LS^{PSC}, Figure 5), the best resulting coding is retained.

4.4. An exact decoder

The decoding of a coding σ, i.e. finding the trigger times of tasks leading to minimum power peak, can be performed in exact fashion based on an Integer Linear Program (ILP). Such an ILP can be derived as a simplified version of $\mathcal{M}^{PSC}$, since not only does a coding σ already integrate decisions about assignment of tasks and resources to workstations (X and R variables of $\mathcal{M}^{PSC}$, respectively), but it also imposes sequencing constraints on the tasks of each workstation due to σ-precedence, which in turn implies the enforcement of precedence constraints, as by (17).
The only decision variables inherited from $\mathcal{M}^{PSC}$ are hence the power-peak variable $W_{_M}$ and the trigger variables S. For the sake of clarity, let us change the definition of the latter slightly:

- binary trigger variables $S_{k,\lambda,t}$, $t \in \mathcal{T}_{k,\lambda}$, $S_{k,\lambda,t} = 1 \Leftrightarrow$ task $\sigma_k(\lambda)$ is triggered at t

A model tailored on σ, which we refer to as $\mathcal{M}^{PSC}(\sigma)$, can be defined as follows:

$$\min W_{_M} \tag{22}$$

$$\text{s.t.} \sum_{t \in \mathcal{T}_{k,\lambda}} S_{k,\lambda,t} = 1 \qquad \forall k \in \{0 \dots |\sigma| - 1\}, \ \forall \lambda \in \{0 \dots |\sigma_k| - 1\} \tag{23}$$

$$S_{k,\lambda+1,t} \leq \sum_{\substack{\tau \in \mathcal{T}_{k,\lambda} \cap \\ \{0 \dots t - t_{\sigma_k(\lambda)}\}}} S_{k,\lambda,\tau} \qquad \forall k \in \{0 \dots |\sigma| - 1\}, \ \forall \lambda \in \{0 \dots |\sigma_k| - 2\}, \ \forall t \in \mathcal{T}_{k,\lambda+1} \tag{24}$$

$$\sum_{\substack{k \in \{0 \dots |\sigma|-1\} \\ \lambda \in \{0 \dots |\sigma_k|-1\}: \\ (\exists t' : d^e_{k,\lambda}(\sigma) \leq t', \\ t' \leq d^l_{k,\lambda}(\sigma) + t_{\sigma_k(\lambda)} - 1) \\ t' \bmod c = t}} \ \sum_{\substack{\tau \in \mathcal{T}_{k,\lambda} \cap \\ \{t - t_{\sigma_k(\lambda)} + 1 \dots t\}}} w_{\sigma_k(\lambda)} \cdot S_{k,\lambda,\tau} \leq W_{_M} \qquad \forall t \in \{0 \dots c - 1\} \tag{25}$$

$$S_{k,\lambda,t} \in \{0, 1\}, W_{_M} \in \mathbb{Z}_+$$

Constraints (23), similarly to (6), state that each task must be triggered at exactly one among the available times. Relations (24) result from an adaptation of (8), and apply only to pairs of tasks that are on the same workstation and consecutive in the coding σ, a relation that is stronger than precedence constraints, which are implicit in σ.
Each constraint (25) refers to a time t of the *actual* timespan $\{0 \dots c-1\}$ of the line, as did the matching constraint (10), and only considers tasks that could possibly be running at t, or t plus some multiple of c on the virtual timespan of the workstation they are assigned to. For instance, in the reference example (see Figure 3) and associated coding (see (15)), task $\sigma_2(0) = 2$ is not considered in the constraint (25) associated with $t = 15$. This is because workstation $k = 2$ has total workload $\sum_{i \in \sigma_k} t_i = t_2 + t_3 = 9 + 5 = 14$, hence number of resources $r_2(\sigma) = 1$ and total idle time $\iota_2(\sigma) = r_2(\sigma) \cdot c - t_2 - t_3 = 6$. Task $\sigma_2(0) = 2$ then has earliest and latest possible starting times $d^e_{2,0}(\sigma) = 0$ and $d^l_{2,0}(\sigma) = d^e_{2,0}(\sigma) + \iota_2(\sigma) = 6$, hence its latest possible running time slot is $d^l_{2,0}(\sigma) + t_2 - 1 = 14$.
Among all the tasks that *could* possibly be running at time t, only those that are *actually* running at t, i.e. that have been triggered at a compatible time, are considered by (25) to be contributing to the cumulative power consumption at t. By bounding this latter term by $W_{_M}$ and $\forall t \in \{0 \dots c - 1\}$, constraints (25) allow the objective function (22) to minimise the overall power peak.
ExactDecoder() can be derived as an ILP solver based on $\mathcal{M}^{PSC}(\sigma)$.

4.5. A heuristic decoder

A quick, greedy construction heuristic decoder, **HeurDecoder()**, can be obtained as shown in Figure 7. **HeurDecoder()** performs two steps:

(1) it schedules all the tasks of station $\mathcal{I}_0(\sigma)$, i.e. that with minimum idle time, at the earliest possible time (lines 1-4);
(2) it considers each other workstation σ_k in the sense of increasing idle time, i.e. for $k = \mathcal{I}_\nu(\sigma)$, $\nu = 1..|\sigma| - 1$, and triggers the tasks of σ_k in their order, i.e. $\sigma_k(\lambda)$, $\lambda = 0..|\sigma_k| - 1$, at time $t \in \mathcal{T}_{k,\lambda}$ s.t. the power peak increment is the minimum.

HeurDecoder(σ)

```
globals  I: PSCALB3PM instance;
argtype  σ: coding;
returns  s: solution obtained by construction;
declare  s: local solution; t, δ_min: time; λ_0, λ, j, k_0, k, ν: integer;
 1  k_0 ← I_0(σ) ; t ← 0
 2  for λ_0 ← 0 to |σ_{k_0}| − 1
 3     j ← σ_{k_0}(λ_0) ; schedule task j at time t ; t ← t + t_j
 4  next λ_0
 5  for ν ← 1 to |σ| − 1
 6     k ← I_ν(σ) ; t ← 0
 7     for λ ← 0 to |σ_k| − 1
 8        j ← σ_k(λ) ; δ_min ← max{t, d^e_{k,λ}(σ)}
 9        t ← time slot in {δ_min ... d^l_{k,λ}(σ)} causing the min peak increment
10        schedule task j at time t ; t ← t + t_j
11     next λ
12  next ν
13  s ← PSCALB3PM solution given by σ and the chosen trigger times
14  return s
```

Figure 7. Pseudo-code of **HeurDecoder()**

4.6. Iteratively improving solutions obtained from feasible codings

Based on the decoders described in Sections 4.4-4.5, routine **Decoder()** obtains a solution from a coding σ as depicted in Figure 8.
First, a solution $\boldsymbol{s}_h$ returned by **HeurDecoder()** is computed. Then, if σ is infeasible (lines 2-4), the value of $\boldsymbol{s}_h$ is subject to a penalty $\text{pen}(\sigma)\cdot\text{UB}_W$, and **Decoder()** can terminate.
If σ is feasible, a better decoding of σ is sought in iterative fashion by means of **ExactDecoder()**. In the first call to it (line 5), $\boldsymbol{s}_h$ serves as warmstart, a time limit tL is set, and a new solution $\boldsymbol{s}'(0)$ is obtained. The use of a warmstart guarantees at least one feasible solution, most of all if time limit tL is tight w.r.t. the hardness of the PSCALB3PM instance and hence, possibly, of the solving of $\mathcal{M}^{PSC}(\sigma)$. Then, since σ is feasible w.r.t. both $r_{\max}$ and $R_{\max}$, **Decoder()** tries to see whether by using more resources, a better solution than $\boldsymbol{s}'(0)$ can be achieved. To do so, each possible number $p = 1..R_{\max} - R(\sigma)$ of additional resources is considered iteratively, trying to add one more resource to each workstation $k \in K \subseteq \mathcal{M}$, i.e. that still has available resource slots (lines 7-11), and keeping the placement of an additional resource that determines the best improvement (lines 12-13). At each iteration, the solution of the previous iteration, $\boldsymbol{s}'(p-1)$, is used as a warmstart (line 10), and a different time limit tL' is used.
At most m iterations per additional resource are run: the procedure stops (line 14) when the addition of a resource to the line does not allow improvement for any of the workstations of K.

5. Computational Experiments

In this section we describe in detail the computational experiments conducted to assess the suitability of ILP model $\mathcal{M}^{PSC}$ and the performance of the matheuristic MS×LS^{PSC}. Since the PSCALB3PM is, as far as the authors are aware, a novel problem studied here for the first time, no previous methods exist for this problem to allow a comparison.
Models $\mathcal{M}^{PSC}$ and $\mathcal{M}^{PSC}(\sigma)$ were implemented and solved with CPLEX 12.7.1 solver; both the solver

DECODER(σ)

globals I: *PSCALB3PM instance*; UB_W: *upper bound on peak power*;
globals timeL, tL, tL': *time limits*;
argtype σ: *coding*;
returns $\boldsymbol{s}^\star$: *best solution obtained*;
declare $\boldsymbol{s}'()$, $\boldsymbol{s}''()$: *arrays of local solutions*; $\boldsymbol{s}_h$: *local solution*;
declare p: *iteration counter*; K: *set of integer*; R', k: *integer*;

```
1  s_h ← HeurDecoder(I, σ)
2  if(pen(σ) > 0)
3     s* = s_h ; W_M(s*) = W_M(s*) + pen(σ)·UB_W ; return s*
4  endif
5  s'(0) ← ExactDecoder(I, σ, WS=s_h, tL) ; s* ← s'(0) ; R' ← R(σ) ; p ← 1
6  while(p ≤ R_max − R' && runtime < timeL)
7     K ← {k ∈ {0...|σ| − 1} | r_k(σ) < r_max}
8     foreach(k ∈ K) if(runtime < timeL)
9        r_k(σ) ← r_k(σ) + 1 // allow one more resource in workstation σ_k
10       s''(k) ← ExactDecoder(I, σ, WS=s'(p − 1), tL') ; r_k(σ) ← r_k(σ) − 1
11    next k
12    k̄ ← argmin_{k∈K} W_M(s''(k)) ; s'(p) ← s''(k̄)
13    r_k̄(σ) ← r_k̄(σ) + 1 ; ι_k̄(σ) ← ι_k̄(σ) + c(I) ; update ranking I(σ)
14    if(W_M(s'(p)) < W_M(s'(p − 1))) s* ← s'(p) else break
15    p ← p + 1
16 endwhile
17 return s*
```

Figure 8. Pseudo-code of **Decoder()**

and MS×LS^{PSC} were run on a Intel Xeon E5-2660 v3 2.6 Ghz machine with 62.65Gb RAM.
A set of 17 SALB3PM instances from Gianessi, Delorme, and Masmoudi (2019) were considered with up to $n = 30$ tasks with power consumption values between 5 and 50, $m = 14$ workstations and target takt time values of up to $c = 94$. The aforementioned SALB3PM instances were associated with values of $r_{\max}$ ranging from 1 to 3, and of $R_{\max}$ from m to $r_{\max} \cdot m$. For each SALB3PM benchmark instance, $3m + 3$ PSCALB3PM cases can thus be obtained: one with $r_{\max} = 1$, $m + 1$ with $r_{\max} = 2$, and $2m + 1$ with $r_{\max} = 3$, giving rise to 375 PSCALB3PM cases in total.
For each such PSCALB3PM instance, the $\mathcal{M}^{PSC}$-based ILP solver was given a 3600s time limit, in order to evaluate its limits w.r.t. the hardness of instances.
MS×LS^{PSC} is given a time limit of 600s (line 2, Figure 5), which seems a reasonable value for a practitioner as to the purposes mentioned at the beginning of Section 4.

5.1. Comparing MS×LS^{PSC} with some variants and reference algorithms

In order to assess the effectiveness of the proposed multi-start matheuristic MS×LS^{PSC}(Figure 5), a set of variants were implemented and tested that differ with respect to three main features:

- the behaviour of **Decoder()** routine (lines 4, 8 of MS×LS^{PSC}, Figure 5);
- the behaviour of **GetNeighbors()** routine (line 7, Figure 5);
- the terminating condition (line 14, Figure 5) for the LS loop.

For the decoding of the current coding σ, three strategies are considered, all based on **Decoder()** routine (Figure 8). In all of them, infeasible codings are decoded by means of **HeurDecoder()**; the upper bound UB_W equals the sum of the $\min\{n, R_{\max}\}$ biggest task power values, plus 1, and hence is by definition unattainable by any feasible solution; the values of the time limits tL and tL' (lines 5, 10 of **Decoder()**, Figure 8) are set to 60s and 10s, respectively.

(D1) the strategy described in Section 4.6;
(D2) unlike D1, the first exact decoding (line 5, Figure 8) is sought without any warmstart: when no solution is found within the imposed time limit tL, σ is considered *nondecoded* and its evaluation is UB_W, i.e. a value which is better than those of any infeasible coding but worse than any feasible, successfully decoded coding;
(D3) unlike D1, **Decoder()** uses **HeurDecoder()** only, both in the first decoding and in the iterative improvement strategy (Figure 8).

It is noteworthy to observe that in case of a feasible coding σ, decoding strategy D1 always succeeds in finding a solution for it, which in the worst case is that returned by **HeurDecoder()**, and similar considerations apply for D3. Also note that D3 is the quickest strategy, but no guarantee is given concerning the quality of the decoding.
As regards the behaviour of **GetNeighbors()** w.r.t. the current coding σ, three options are also considered, listed in order of growing complexity:

(N1) contrary to what is described in Section 4.3, after randomly picking a workstation k and one of its tasks, $j \in \sigma_k$, the insertion position is chosen randomly among those available for it;
(N2) the strategy described in Section 4.3, i.e. random choice of the task j to move, enumeration of all the related insertion options, evaluated by means of **Decoder()**, and choice of the most improving one;
(N3) a complete enumeration of the neighbours of σ is performed: all tasks are taken into consideration for the movement, along with all the corresponding insertion options; every such option is evaluated by **Decoder()**, and the overall most improving is retained.

In all three cases, a neighbour of σ is accepted as the new current coding (lines 9-11 of MS×LS^{PSC}, Figure 5) if its decoding provides a better value, but also the same value, to promote diversification. Among the three options, only N3 leads to a deterministic descent, while N1 and N2 give rise to a stochastic descent.
Finally, three possible stopping criteria are tested for the local search of MS×LS^{PSC} (line 14, Figure 5): all are based on the description of Section 4.2 and differ by the values of b^1_{LS} and b^2_{LS}:

(S1) $b^1_{\text{LS}} = 50$, $b^2_{\text{LS}} = 50$, i.e. a total number of $b^1_{\text{LS}} = 50$ LS iterations are run;
(S2) $b^1_{\text{LS}} = +\infty$, $b^2_{\text{LS}} = 50$, i.e. a total number of $b^2_{\text{LS}} = 50$ LS iterations without improvement are run;
(S3) $b^1_{\text{LS}} = +\infty$, $b^2_{\text{LS}} = 1$, i.e. LS stops at the first iteration without improvement.

Criteria S1 and S2 are designed to stop the stochastic descent induced by options N1 and N2 for the behaviour of **GetNeighbors()**, while criterion S3 only fits option N3, making each run of MS×LS^{PSC} a deterministic descent to a local optimum. Other combinations would be possible but either make no sense (N3 combined with S2) or seem uninteresting (S3 combined with N1 or N2, i.e. a stochastic descent stopping at the first nonimproving step; S1 combined with N3, i.e. a deterministic descent stopping after a fixed number of steps). Therefore, the total range of combinations of possible **GetNeighbors()** behaviours and stopping criteria give rise to 5 options:

- 4 obtained by combining one of options N1 and N2, and one of criteria S1 and S2;
- 1 obtained by combining N3 and S3.

This gives rise to 15 variants of MS×LS^{PSC}, including the one presented in Section 4.
In order to enlarge the range of comparison terms, we also designed, implemented and tested an *Iterated Local Search* (ILS). ILS (see e.g. Lourenço, Martin, and Stützle 2003) is a LS-based metaheuristic which escapes a local optimum by randomly perturbating it to launch a new descent. In our case, the perturbation consists in applying 3 random neighbourhood movements, ensuring that each visited coding is different from the previous ones. Since ILS seeks local optima, we only used N3 and S3; however, the three decoding strategies can be used, leading to three ILS algorithms.
Table 1 allows a comparison among the 18 tested methods, labelled MS01 to MS15 and ILS1 to ILS3, respectively. The method proposed in this article (Section 4) is MS04. All methods are run 20 times for each instance to take their randomness into account. The same aforementioned time limit of 600s is imposed on each replication. For each PSCALB3PM instance, up to 361 results exist, namely the 20 replications from each method, plus the best solution possibly found by the ILP solver, as well as a lower bound, determined by solving $\mathcal{M}^{PSC}$. Hence, for each such instance, a best known lower (LB) and upper bound (UB) can be determined:

- the best known UB could come from one of the 18 methods, or the solving of $\mathcal{M}^{PSC}$, or from an instance with lower $R_{\max}$ or $r_{\max}$ values, whose solutions are also feasible for cases associated with larger values;
- the best known LB could come from an instance with larger $R_{\max}$ or $r_{\max}$ values, whose best known lower bounds are also valid for cases associated with lower values.

Therefore, for each solution $\mathcal{S}$, gaps $\%_{\text{LB}}$ and $\%_{\text{BK}}$ w.r.t. the associated LB and UB can be computed

as follows:

$$\%_{\text{LB}} = \frac{W_M(\mathcal{S}) - LB}{LB} \qquad \%_{\text{BK}} = \frac{W_M(\mathcal{S}) - UB}{UB} \tag{26}$$

Such gaps can then be averaged, for each instance, over all the replications, then over each group of instances derived from the same SALB3PM instance, that differ for their $r_{\max}$ and $R_{\max}$ values. Replications of some instances having yielded infeasible solutions or no solution are obviously not taken into account. The obtained average gaps are denoted in the following as $\overline{\%}_{\text{BK}}$ and $\overline{\%}_{\text{LB}}$. Table 1 shows, for each method and family of instances, the value obtained for $\overline{\%}_{\text{BK}}$. The last three rows report, for each version, the weighted average of $\overline{\%}_{\text{BK}}$, as well as the worst recorded value for $\%_{\text{BK}}$, and the percentage of found feasible solutions.

By looking at both the weighted average gap and maximum gap, the results of Table 1 suggest some considerations as to the decoding strategies. If we compare methods that only differ in the decoding, e.g. MS01 vs MS06 vs MS11, or MS04 vs MS09 vs MS14, or ILS1 vs ILS2 vs ILS3, we see that those based on strategy D3 perform significantly worse than those using D1 and D2, which shows the interest of the ILP decoder. As regards the comparison of D1 and D2, the impact of using a heuristic decoder before the exact decoder to provide it with a warmstart solution, which is what distinguishes them, is not negligible although sometimes weak, i.e. methods MS01 to MS05 seem to perform slightly better than MS06 to MS10, whereas the performance difference between ILS1 and ILS2 appears to be more significant.

By looking at the average percentage of feasible solutions, and again taking methods that only differ in the decoding, results are very similar, which is not very surprising if we consider that all decoders have the same behaviour until the algorithm finds a feasible solution. More remarkable differences occur among methods based on N3, e.g. MS05 vs MS10 but also ILS1 vs ILS2, as the neighbourhood complete enumeration is time-consuming and algorithms may not have converged at the end of a replications, inducing a stronger variability.

For all decoding strategies, the methods making use of a complete enumeration of the neighbourhood of a coding, i.e. the neighbourhood option N3, seemingly also have worse results, and among them, those making use of multi-start seem to behave better than those based on ILS: e.g. for strategy D1, MS05 has worse results than MS01 to MS04, but better than ILS1, and the same seems to hold for methods MS06 to MS10 and ILS2, and MS11 to MS15 and ILS3. This can probably be explained by the fact that option N3 is too time consuming w.r.t. the ratio between the time limit of a replication and those imposed to the calls of **ExactDecoder()** during a run of **Decoder()**. The same considerations apply to the comparison of the average number of feasible solutions.

If we now focus on the first four methods per decoding strategy, i.e. those using neighbourhood options N1 or N2, and consequently, stopping criteria S1 or S2, it seems that the combination of S2 and N2 yields the best results: this is true, in terms of average gap values, for MS04 w.r.t. MS01-MS03, for MS09 w.r.t. MS06-MS08, and for MS14 w.r.t. MS11-MS13, whereas this is not always the case for maximum gap values. These results are not completely surprising: neighbourhood option N2 is neither completely random as N1, nor time-consuming as N3, and option S2 certainly allows for a more effective exploration of the coding space. These considerations seem to hold also for the average number of feasible solutions.

The results of Table 1 seem then to suggest that the MS×LS^{PSC} version proposed in this article, identified here as MS04, is the most effective. Therefore, in the following the symbol MS×LS^{PSC} will refer to MS04.

5.2. Results Analysis for $\mathcal{M}^{PSC}$ and MS×LS^{PSC}

The two solving methods (solver, MS×LS^{PSC}) allowed the best known lower and upper bound (LB, UB) to be collected for the 375 instances. The two bounds obviously coincide for instances solved to optimality.

Table 2 shows the results of solving the ILP model $\mathcal{M}^{PSC}$ on the PSCALB3PM instances. Results are aggregated per original SALB3PM instance, all $R_{\max}$ value considered, and separated according to the value of $r_{\max}$. The reported figures are the average values of the optimality gap, $\overline{\%}_{\text{LB}}$, the computation time, $\overline{\text{cpu}}$, the percentage of instances for which a feasible solution has been found, $\overline{\text{feas}}(\%)$, and those which have been solved to optimality, $\overline{\text{opt}}(\%)$. The four rightmost columns show aggregate figures, all $r_{\max}$ value considered.

An overall average gap of 2.61% is obtained by solving the ILP model; the overall average running time

Table 1. Results of the 15 MS×LS^{PSC} variants and 3 ILS methods as to average/maximum gaps w.r.t. best known solutions ($\%_{\mathrm{BK}}$) and average percentage of feasible solutions found.

				method	MS01	MS02	MS03	MS04	MS05	ILS1	MS06	MS07	MS08	MS09	MS10	ILS2	MS11	MS12	MS13	MS14	MS15	ILS3
				decoding strategy	D1						D2						D3					
				neighborhood visiting strategy	N1		N2		N3	(N3)	N1		N2		N3	(N3)	N1		N2		N3	(N3)
instance	$(n,$	$m,$	$c)$	LS stop	S1	S2	S1	S2	S3	(S3)	S1	S2	S1	S2	S3	(S3)	S1	S2	S1	S2	S3	(S3)
bowman-1	8	5	20		0.00%	0.00%	0.00%	0.00%	0.12%	0.10%	0.00%	0.00%	0.00%	0.00%	0.12%	0.11%	0.00%	0.00%	0.00%	0.00%	1.45%	1.51%
buxey-1	29	14	25		0.82%	0.68%	0.71%	0.57%	0.67%	0.64%	0.82%	0.65%	0.73%	0.60%	0.71%	0.67%	0.91%	0.72%	0.62%	0.36%	2.48%	2.55%
buxey-2	29	7	47		2.28%	2.13%	2.18%	1.93%	2.44%	2.84%	2.34%	2.10%	2.26%	1.99%	2.49%	6.94%	1.56%	1.28%	1.15%	0.70%	5.01%	4.87%
jackson-1	11	8	7		0.00%	0.00%	0.00%	0.00%	0.23%	0.23%	0.00%	0.00%	0.01%	0.00%	0.23%	0.21%	0.09%	0.09%	0.09%	0.09%	1.78%	1.84%
jackson-2	11	3	21		0.00%	0.00%	0.00%	0.00%	1.54%	2.05%	0.00%	0.00%	0.00%	0.00%	1.54%	1.98%	2.16%	2.16%	2.16%	2.16%	11.32%	12.46%
jaeschke-1	9	8	6		0.00%	0.00%	0.00%	0.00%	0.19%	0.16%	0.00%	0.00%	0.00%	0.00%	0.19%	0.18%	0.37%	0.37%	0.37%	0.37%	2.92%	2.74%
jaeschke-2	9	3	18		0.00%	0.00%	0.00%	0.00%	0.14%	0.03%	0.00%	0.00%	0.00%	0.00%	0.14%	0.00%	0.71%	0.71%	0.71%	0.71%	7.53%	6.93%
mansoor-1	11	4	48		0.02%	0.02%	0.01%	0.02%	0.14%	0.20%	0.03%	0.03%	0.03%	0.03%	0.14%	0.45%	0.10%	0.10%	0.10%	0.10%	0.63%	0.47%
mansoor-2	11	2	94		0.00%	0.01%	0.01%	0.00%	0.58%	0.37%	0.00%	0.00%	0.00%	0.01%	0.58%	1.04%	0.00%	0.00%	0.00%	0.00%	1.33%	1.23%
mertens-1	7	6	6		0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.18%	0.17%
mertens-2	7	2	18		0.00%	0.00%	0.00%	0.00%	2.93%	2.49%	0.00%	0.00%	0.00%	0.00%	2.93%	2.40%	14.04%	14.04%	14.04%	14.04%	15.02%	14.93%
mitchell-1	21	8	14		0.63%	0.55%	0.54%	0.51%	0.87%	0.95%	0.62%	0.55%	0.52%	0.49%	0.89%	0.96%	0.53%	0.45%	0.31%	0.22%	3.11%	3.24%
mitchell-2	21	3	39		3.39%	3.15%	3.39%	2.79%	5.18%	5.42%	3.29%	3.11%	3.53%	3.28%	5.34%	5.40%	1.46%	1.43%	1.18%	1.12%	11.05%	10.93%
roszieg-1	25	10	14		0.50%	0.48%	0.49%	0.45%	0.49%	0.54%	0.49%	0.47%	0.49%	0.48%	0.53%	0.54%	0.41%	0.37%	0.29%	0.23%	2.25%	2.22%
roszieg-2	25	4	32		2.29%	2.23%	2.22%	1.97%	3.16%	5.43%	2.32%	2.25%	2.29%	2.05%	3.25%	5.51%	1.32%	1.22%	1.05%	0.71%	7.52%	9.21%
sawyer-1	30	14	25		0.84%	0.82%	0.75%	0.66%	0.76%	0.73%	0.82%	0.83%	0.72%	0.66%	0.76%	0.71%	0.99%	0.78%	0.65%	0.41%	2.38%	2.40%
sawyer-2	30	7	47		2.88%	2.64%	2.74%	2.42%	2.47%	2.50%	2.95%	2.69%	2.74%	2.49%	2.66%	2.68%	1.87%	1.54%	1.49%	1.02%	4.74%	4.61%
				Average ($\overline{\%_{BK}}$)	0.82%	0.76%	0.77%	0.66%	1.06%	1.19%	0.82%	0.76%	0.78%	0.70%	1.10%	1.49%	1.09%	0.99%	0.91%	0.77%	3.72%	3.79%
				Maximum ($\%_{BK}$)	10.38%	7.09%	5.32%	4.32%	12.72%	22.22%	10.38%	7.09%	5.28%	4.53%	12.72%	22.36%	14.04%	14.04%	14.04%	14.04%	18.86%	22.22%
Average (% feas. solutions found)					98.96%	99.15%	99.15%	99.47%	81.76%	80.79%	98.96%	99.15%	99.15%	99.47%	80.97%	80.80%	98.96%	99.15%	99.19%	99.47%	83.99%	83.68%

is less than 2600s, although this figure can vary significantly according to the instance. Since feasible solutions are found for 65% of the instances, of which about 50% lead to solutions of proven optimality (32% of the total), we can reasonably infer that instances not solved to optimality have an average gap still to close at the time limit of around 5%. Performances can be in some cases very good, with feasible solutions regularly found with c up to around 20, and in some cases even more if m is within 2 and 4, although a general overview reveals that model $\mathcal{M}^{PSC}$ can struggle in finding solutions for instances of medium to large size, as e.g. for buxey-1, buxey-2, sawyer-1 and sawyer-2 instances. This trend strengthens as $r_{\max}$ grows, which is normal as the time horizon, hence the size of the time-indexed model, grows significantly with the number of resource options for the workstations. Running times also suggest this trend. It can be shown that the best case is represented by $R_{\max} = m$, as the average results, all instances and $r_{\max}$ values considered, are all better than the average general values: 1.82% for $\overline{\%}_{\text{LB}}$, 1910.7 for $\overline{\text{cpu}}$, 78.43% for $\overline{\text{feas}}(\%)$ and 54.90% for $\overline{\text{opt}}(\%)$. All these consideration lead us to conclude that an heuristic approach is indeed advisable.
Table 3 allows an overview of the performances of MS×LS^{PSC} as regards the power peak. The Table reports the average gaps $\overline{\%}_{\text{LB}}$ and $\overline{\%}_{\text{BK}}$ obtained by MS×LS^{PSC}, computed as explained previously; for the second, the average standard deviation $\overline{\sigma}(\%_{\text{BK}})$ is also reported. Moreover, a comparison between the results of the matheuristic and a set of simpler decoders is provided: terms $\overline{\%}^{\text{ESD}}_{\text{BK}}$, $\overline{\%}^{\text{HD}}_{\text{BK}}$ and $\overline{\%}^{\text{ILPD}}_{\text{BK}}$ concern the average gap achieved for a randomly generated feasible coding by, respectively:

- a naive decoder, consisting in assigning the production tasks to the workstations and scheduling them at the earliest possible starting date;
- a greedy heuristic decoder, i.e. the routine **HeurDecoder()** alone;
- an ILP-based decoder, i.e. the routine **ExactDecoder()** alone, with a warmstart solution obtained by means of **HeurDecoder()**.

The three figures represent average gaps w.r.t. the best known solution, computed similarly to $\overline{\%}_{\text{BK}}$.
MS×LS^{PSC} has an average gap of 1.52% w.r.t. the best known LB, and of 0.66% w.r.t.the best known solution, which seems to suggest that the proposed heuristic has very good performances. Indeed, $\overline{\%}_{\text{LB}}$ represents an upper bound on the average gap between the solution yielded by MS×LS^{PSC}w.r.t. the optimal solution; $\overline{\%}_{\text{BK}}$ is a lower bound on the percentage distance w.r.t. the optimum (which by definition has a value less than or equal to the known solutions). Moreover, a 0.66% gap w.r.t. the best known solution could suggest that the behaviour of MS×LS^{PSC} is very stable, as the quality of the solution does not change significantly among the replications. In more detail, the higher gaps are found for instances with more than 30 tasks (with the exception of mansoor-2): the solution quality offered by MS×LS^{PSC} seems to worsen with the increasing instance size, which is to be expected since many performance aspects of MS×LS^{PSC} depend on the instance size, e.g. the size of the reduced ILP model, whereas the stopping criterion is a fixed time, and the same for all instances. Lastly, on Table 3, the fact that both $\overline{\%}_{\text{LB}}$ and $\overline{\%}_{\text{BK}}$ are most often either both 0 or both greater than 0 could mean that when the algorithm performance decreases, it is at least partially due to the worsening ILP lower bound.
The rightmost part of Table 3 shows the advantage of using MS×LS^{PSC} to achieve a power peak reduction. The value of 49.89% for gap $\overline{\%}^{\text{ESD}}_{\text{BK}}$ means that the naive solution, which could be adopted by a practitioner not supported by any optimisation algorithm, represents the average percentage power peak augmentation w.r.t. the best solution, meaning that by adopting MS×LS^{PSC}, the peak of the power consumption can by reduced by 33%. About 65% of this reduction can be obtained by using the heuristic decoder **HeurDecoder()**, and almost 90% can be achieved by using the heuristic and ILP-based decoders combined. It is noteworthy to point out that such reductions can be obtained all other production criteria being equal, namely the takt time and the number of workstations, which seems to suggest that using the proposed methods, and particularly the MS×LS^{PSC} matheuristic, can be very advantageous in an MS.
Table 4 shows the relation between the term $\overline{R}_{\max} = R_{\max} - m$, i.e. the difference between the maximum authorised number of resources for the line and the number of workstations, and the average number of starts of a replication of MS×LS^{PSC}, denoted as $\overline{\#}_{\text{st}}$. $\overline{R}_{\max}$ essentially represents how far we can get w.r.t. the minimum required number of resources for the line. The term $\#_{\text{repl}}$ represents the number of replications of PSCALB3PM instances characterised by a given value of $\overline{R}_{\max}$.
The Table shows that $\overline{\#}_{\text{st}}$, the average number of solutions yielded by MS×LS^{PSC} during a replication, drops dramatically as soon as $R_{\max}$ increases over m. When this happens, even for more constrained instances, the heuristic is more likely to find feasible solutions and invoke the ILP solving, but is also more likely to execute the part (lines 6-16, Figure 8) in which remaining available resources are

Table 2. Results of ILP model $\mathcal{M}^{PSC}$.

instance	n	m	c	$r_{\max}=1$				$r_{\max}=2$				$r_{\max}=3$				*Total*			
				$\overline{\%_{\text{LB}}}$	$\overline{\text{cpu}}$	$\overline{\text{feas}}(\%)$	$\overline{\text{opt}}(\%)$	$\overline{\%_{\text{LB}}}$	$\overline{\text{cpu}}$	$\overline{\text{feas}}(\%)$	$\overline{\text{opt}}(\%)$	$\overline{\%_{\text{LB}}}$	$\overline{\text{cpu}}$	$\overline{\text{feas}}(\%)$	$\overline{\text{opt}}(\%)$	$\overline{\%_{\text{LB}}}$	$\overline{\text{cpu}}$	$\overline{\text{feas}}(\%)$	$\overline{\text{opt}}(\%)$
bowman-1	8	5	20	0.00%	0.1	100.00%	100.00%	0.00%	303.0	100.00%	100.00%	6.06%	3520.1	100.00%	9.09%	3.71%	2252.1	100.00%	44.44%
buxey-1	29	14	25	0.00%	2663.6	100.00%	100.00%	15.45%	3600.0	6.67%	0.00%	na	3601.2	0.00%	0.00%	7.73%	3580.0	4.44%	2.22%
buxey-2	29	7	47	na	3600.0	0.00%	0.00%	na	3600.0	0.00%	0.00%	11.39%	3600.6	13.33%	0.00%	11.39%	3600.4	8.33%	0.00%
jackson-1	11	8	7	0.00%	0.0	100.00%	100.00%	0.00%	186.0	100.00%	100.00%	0.04%	246.2	100.00%	94.12%	0.02%	217.0	100.00%	96.30%
jackson-2	11	3	21	0.00%	4.1	100.00%	100.00%	0.00%	1737.1	100.00%	100.00%	3.83%	3567.7	100.00%	14.29%	2.24%	2660.5	100.00%	50.00%
jaeschke-1	9	8	6	0.00%	0.0	100.00%	100.00%	0.00%	7.0	100.00%	100.00%	0.00%	47.2	100.00%	100.00%	0.00%	32.0	100.00%	100.00%
jaeschke-2	9	3	18	0.00%	0.9	100.00%	100.00%	0.00%	168.3	100.00%	100.00%	0.00%	1595.0	100.00%	100.00%	0.00%	986.6	100.00%	100.00%
mansoor-1	11	4	48	0.00%	0.6	100.00%	100.00%	8.00%	3598.8	100.00%	0.00%	7.80%	3600.2	100.00%	0.00%	7.35%	3359.7	100.00%	6.67%
mansoor-2	11	2	94	0.00%	3.1	100.00%	100.00%	4.76%	1511.8	100.00%	66.67%	8.57%	3122.6	100.00%	40.00%	6.35%	2239.1	100.00%	55.56%
mertens-1	7	6	6	0.00%	0.0	100.00%	100.00%	0.00%	1.1	100.00%	100.00%	0.00%	1.4	100.00%	100.00%	0.00%	1.3	100.00%	100.00%
mertens-2	7	2	18	0.00%	0.2	100.00%	100.00%	0.00%	1.4	100.00%	100.00%	0.00%	2.7	100.00%	100.00%	0.00%	2.0	100.00%	100.00%
mitchell-1	21	8	14	0.00%	12.7	100.00%	100.00%	1.48%	3600.1	100.00%	0.00%	1.34%	3600.4	100.00%	0.00%	1.34%	3467.4	100.00%	3.70%
mitchell-2	21	3	39	0.00%	1867.3	100.00%	100.00%	5.24%	3600.0	100.00%	0.00%	8.03%	3600.7	100.00%	0.00%	6.43%	3456.0	100.00%	8.33%
roszieg-1	25	10	14	0.00%	221.1	100.00%	100.00%	1.38%	3599.4	100.00%	0.00%	1.24%	3600.1	100.00%	0.00%	1.25%	3497.5	100.00%	3.03%
roszieg-2	25	4	32	0.00%	1719.3	100.00%	100.00%	5.42%	3600.0	100.00%	0.00%	5.71%	3599.0	88.89%	0.00%	5.20%	3474.1	93.33%	6.67%
sawyer-1	30	14	25	1.53%	3591.6	100.00%	0.00%	na	3600.0	0.00%	0.00%	4.49%	3600.8	3.45%	0.00%	3.01%	3600.3	4.44%	0.00%
sawyer-2	30	7	47	na	3600.0	0.00%	0.00%	na	3600.0	0.00%	0.00%	51.96%	3601.1	13.33%	0.00%	51.96%	3600.7	8.33%	0.00%
Average				0.10%	1016.8	88.24%	82.35%	1.83%	2429.7	64.00%	35.20%	3.28%	2743.3	63.95%	26.61%	2.61%	2560.5	65.07%	32.00%

Table 3. Average performances of MS×LS^{PSC} w.r.t. the best known lower and upper bound, and comparison as to power peak with alternative decoders.

instance	n	m	c	$\overline{\%}_{\mathrm{LB}}$	$\overline{\%}_{\mathrm{BK}}$	$\overline{\sigma}(\%_{\mathrm{BK}})$	$\overline{\%}_{\mathrm{BK}}^{\mathrm{ESD}}$	$\overline{\%}_{\mathrm{BK}}^{\mathrm{HD}}$	$\overline{\%}_{\mathrm{BK}}^{\mathrm{ILPD}}$
bowman-1	8	5	20	0.00%	0.00%	0.00%	50.82%	18.33%	3.73%
buxey-1	29	14	25	0.85%	0.57%	0.28%	49.54%	13.61%	2.45%
buxey-2	29	7	47	3.49%	1.93%	0.65%	59.38%	19.75%	6.47%
jackson-1	11	8	7	0.03%	0.00%	0.07%	50.50%	19.11%	8.24%
jackson-2	11	3	21	1.72%	0.00%	0.00%	69.13%	37.46%	17.32%
jaeschke-1	9	8	6	0.00%	0.00%	0.00%	37.30%	23.68%	16.97%
jaeschke-2	9	3	18	0.00%	0.00%	0.00%	61.33%	29.84%	9.91%
mansoor-1	11	4	48	6.41%	0.02%	0.13%	34.72%	9.10%	1.97%
mansoor-2	11	2	94	0.00%	0.00%	0.00%	30.06%	11.92%	5.34%
mertens-1	7	6	6	0.00%	0.00%	0.00%	28.34%	5.94%	3.05%
mertens-2	7	2	18	0.00%	0.00%	0.00%	60.84%	34.63%	18.94%
mitchell-1	21	8	14	1.02%	0.51%	0.28%	47.92%	13.79%	3.17%
mitchell-2	21	3	39	5.72%	2.79%	1.70%	66.69%	33.62%	12.80%
roszieg-1	25	10	14	0.86%	0.45%	0.32%	49.04%	11.44%	1.82%
roszieg-2	25	4	32	4.32%	1.97%	1.11%	54.41%	23.44%	8.24%
sawyer-1	30	14	25	1.19%	0.66%	0.29%	52.88%	12.26%	1.95%
sawyer-2	30	7	47	3.82%	2.42%	0.75%	58.06%	19.68%	6.33%
Average				1.52%	0.66%	0.96%	49.89%	17.51%	6.12%

added iteratively, giving rise to further ILP calls. Hence, for $\overline{R}_{\max} > 0$, the time spent in ILP solving presumably represents the largest portion of the running time, which mechanically pushes MS×LS^{PSC} to explore a far lower number of solutions. However, this also means that MS×LS^{PSC} has an inherently adaptive behaviour: when the instance and/or the reduced margin in terms of overall number of available resources induce a higher likelihood of infeasible solutions, a higher number of solutions is generated and explored, which is a form of diversification strategy, while when feasible solutions are found, the algorithm switches to an intensification behaviour.

Table 4 also reports how the average gaps $\overline{\%}_{\mathrm{BK}}$ and $\overline{\%}_{\mathrm{LB}}$ (here averaged w.r.t. $\overline{R}_{\max}$) between MS×LS^{PSC} solutions and the best known solution and lower bounds, but also the average percentage of used resources and workstations, respectively $\overline{\mathrm{res}}_{\%\mathrm{u}}$ and $\overline{\mathrm{wst}}_{\%\mathrm{u}}$, evolve as the relative number of available resources increases. $\overline{\mathrm{res}}_{\%\mathrm{u}}$ and $\overline{\mathrm{wst}}_{\%\mathrm{u}}$ descend from the two terms expressed in (27), of which they represent the average over all the replications associated with a given value of $\overline{R}_{\max}$, all families of instances considered:

$$\mathrm{res}_{\%\mathrm{u}} = \frac{\sum_k r_k}{R_{\max}} \qquad \mathrm{wst}_{\%\mathrm{u}} = \frac{|\{k : r_k > 0\}|}{m} \tag{27}$$

The quality of the solutions offered by the heuristic improves as $\overline{R}_{\max}$ grows: in particular for $R_{\max} \geq m+1$, and following the same reasoning as Table 3, the gap w.r.t. the optimum is always in the interval $[0.75\%, 2.00\%]$, which progressively narrows with $\overline{R}_{\max}$. Since Table 4 has shown that, when $\overline{R}_{\max}$ increases, a growing amount of time is allocated to solving the ILP model iteratively, we could possibly infer that the ILP-based solution refinement of routine **Decoder()** is effective. We also notice that when $\overline{R}_{\max}$ increases, the average proportion of resources actually used by MS×LS^{PSC}, $\overline{\mathrm{res}}_{\%\mathrm{u}}$, progressively decreases; this could lead to conclude that adding resources is in general not a convenient choice, except for increasing a few units. However, the overall average value of $\overline{\mathrm{wst}}_{\%\mathrm{u}}$ is 86.51%, remaining in general around 90% for $\overline{R}_{\max}$ from 4 to 16, and approaching 100% only when the number of available resources grow very strongly. This seems to suggest that the heuristic exploits the possibility of adding parallel resources. Incidentally, $\overline{R}_{\max} = 0$ is the only case for which the best solution found by the heuristic can be infeasible, which happens in 3.92% of the cases, all for $r_{\max} = 1$. This seems to indicate that MS×LS^{PSC} is not well suited for PSCALB3PM instances where the number of available resources equals m and $r_{\max} = 1$ (i.e. the SALB3PM), and that tailored heuristic algorithms for the SALB3PM could be preferable.

Finally, Table 5 allows us to make some considerations more related to the problem than to the performances of the proposed algorithms. The Table reports, averaged for each group of PSCALB3PM instances derived from the same SALB3PM benchmark and having $\overline{R}_{\max} = 0$: the best optimality gap achieved for $r_{\max} = 1, 2$ and 3, $\%(r_{\max})$; the gap improvements $\%\mathrm{BK}(r_{\max})$ and $\%\mathrm{LB}(r_{\max})$ of, respectively, the best known upper and lower bounds when increasing $r_{\max}$ from 1 to 2 and from 2 to

Table 4. Behavior, performance and solution analysis of MS×LS^{PSC} depending on the relative number $\overline{R}_{\max}$ of available resources.

$\overline{R}_{\max}$	$\#_{\text{repl}}$	$\overline{\#}_{\text{st}}$	$\overline{\%}_{\text{LB}}$	$\overline{\%}_{\text{BK}}$	$\#\text{inf}_{\text{inst}}(\%)$	$\overline{\text{res}}_{\%\text{u}}$	$\overline{\text{wst}}_{\%\text{u}}$
0	1020	28701.82	1.36%	0.35%	3.92%	99.83%	76.53%
1	680	276.42	2.00%	0.75%	0.00%	98.66%	74.96%
2	680	61.90	1.89%	0.76%	0.00%	95.60%	80.76%
3	640	49.24	1.95%	0.84%	0.00%	91.18%	85.10%
4	580	50.69	1.86%	0.75%	0.00%	88.36%	88.20%
5	500	56.46	1.68%	0.74%	0.00%	86.09%	89.27%
6	480	57.49	1.76%	0.75%	0.00%	82.01%	90.18%
7	400	21.30	1.66%	0.75%	0.00%	83.00%	92.41%
8	360	23.53	1.43%	0.58%	0.00%	78.14%	91.62%
9	260	17.67	0.99%	0.62%	0.00%	76.31%	90.11%
10	260	17.79	1.01%	0.63%	0.00%	72.66%	90.39%
11	220	20.84	1.13%	0.72%	0.00%	70.69%	90.71%
12	220	20.76	1.13%	0.72%	0.00%	67.99%	91.05%
13	200	3.84	1.22%	0.77%	0.00%	67.88%	94.65%
14	200	3.89	1.23%	0.79%	0.00%	65.46%	94.67%
15	120	5.38	0.61%	0.36%	0.00%	61.61%	92.24%
16	120	5.26	0.60%	0.35%	0.00%	59.85%	92.99%
17	60	1.07	0.86%	0.54%	0.00%	64.21%	97.19%
18	60	1.07	0.87%	0.55%	0.00%	61.77%	97.31%
19	60	1.07	0.90%	0.58%	0.00%	60.22%	97.19%
20	60	1.07	0.87%	0.55%	0.00%	58.10%	97.31%
21	40	1.10	0.93%	0.64%	0.00%	59.50%	96.96%
22	40	1.10	0.93%	0.64%	0.00%	58.68%	97.50%
23	40	1.10	0.94%	0.65%	0.00%	56.69%	96.79%
24	40	1.10	0.93%	0.64%	0.00%	55.39%	96.96%
25	40	1.10	0.94%	0.66%	0.00%	53.59%	96.79%
26	40	1.10	0.91%	0.62%	0.00%	52.00%	96.96%
27	40	1.10	0.94%	0.65%	0.00%	50.98%	96.79%
28	40	1.10	0.90%	0.61%	0.00%	50.06%	96.96%
Average			1.52%	0.66%	0.53%	83.98%	86.51%

3; the same gap $\%\text{BK}(r_{\max})$ when the solution improved by increasing $r_{\max}$ was proven optimal. This means that here we do not consider the possibility of augmenting the number of resources, but instead of switching from a serial setting to a parallel-serial configuration.

Table 5. Solution improvements when augmenting the number of possible parallel resources, without changing their overall number.

instance	$\%(r_{\max})$			$\%\text{BK}(r_{\max})$		$\%\text{LB}(r_{\max})$		$\%\text{BK}(r_{\max})$ (opt.only)	
	1	2	3	$1 \Rightarrow 2$	$2 \Rightarrow 3$	$1 \Rightarrow 2$	$2 \Rightarrow 3$	$1 \Rightarrow 2$	$2 \Rightarrow 3$
bowman-1	0.00%	0.00%	0.00%	6.19%	0.00%	6.19%	0.00%	6.19%	0.00%
buxey-1	0.00%	0.80%	0.53%	5.06%	0.27%	5.82%	0.00%	5.06%	
buxey-2	$+\infty$	2.94%	2.46%	99.80%	0.49%	0.00%	0.00%		
jackson-1	0.00%	0.00%	0.61%	12.23%	1.21%	12.23%	1.82%	12.23%	1.21%
jackson-2	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
jaeschke-1	0.00%	0.00%	0.00%	20.48%	0.00%	20.48%	0.00%	20.48%	0.00%
jaeschke-2	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
mansoor-1	0.00%	0.00%	9.77%	0.00%	0.00%	0.00%	9.77%	0.00%	0.00%
mansoor-2	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
mertens-1	0.00%	0.00%	0.00%	5.85%	0.00%	5.85%	0.00%	5.85%	0.00%
mertens-2	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
mitchell-1	0.00%	1.00%	1.00%	4.74%	0.00%	5.69%	0.00%	4.74%	
mitchell-2	0.00%	5.26%	4.00%	0.00%	1.32%	5.26%	0.00%	0.00%	
roszieg-1	0.00%	0.78%	0.78%	0.39%	0.00%	1.17%	0.00%	0.39%	
roszieg-2	0.00%	5.13%	5.13%	1.68%	0.00%	6.72%	0.00%	1.68%	
sawyer-1	1.53%	0.93%	0.93%	1.23%	0.00%	0.62%	0.00%		
sawyer-2	$+\infty$	3.41%	2.86%	99.82%	0.57%	0.00%	0.00%		
Average	0.10%	1.19%	1.65%	3.86%	0.23%	4.12%	0.68%	4.05%	0.13%

As it was expected, the average optimality gap $\%(r_{\max})$ increases with $r_{\max}$, due to the increased size of the search space and of the model $\mathcal{M}^{PSC}$, although the average value 1.65% achieved for $r_{\max} = 3$ shows

that a good approximation of the optimal solution can be achieved in most cases. Equally expected is the fact that peak improvements are more consistent when going from $r_{\max} = 1$ to $r_{\max} = 2$, whereas it seems that only very poor further improvements can be achieved with $r_{\max} = 3$: if we can consider only the cases for which the peak being improved corresponds to an optimal solution, the average values are, respectively, 4.05% and 0.13%. Therefore, it seems that from a managerial viewpoint it is considerably more interesting, in order to reduce the power peak, to rearrange the existing resources to increase the number of workstations with parallel resources, rather than the overall number of resources. This seems to suggest, also from an algorithmic point of view, that the search for solutions with a higher number of resources could be stopped at earlier stages, in order to make a more efficient use of the computational time.

6. Conclusions and Perspectives

Motivated by the growing interest in Reconfigurable Manufacturing Systems as a lever to pursue energy efficiency, in this work we have studied the problem of balancing a particular type of single-product RMS, a paced Parallel-Serial line with Crossover, in order to minimise the electric power peak related to processing the production tasks, while complying with a target takt time. The Parallel-Serial-with-Crossover Assembly Line Balancing Problem with Power Peak Minimization (PSCALB3PM) is a new problem as far as we are aware, and therefore as a first contribution we defined it thoroughly. Since it generalises another recent problem, the SALB3PM, which is NP-hard, the PSCALB3PM is NP-hard, too. The second and most important contribution is the definition of a time-indexed Integer Linear Program, $\mathcal{M}^{PSC}$, for the PSCALB3PM. We then defined a matheuristic, MS×LS^{PSC}, in order to allow practitioners to deal with large instances.
Both the ILP model and the matheuristic have been tested on an extended set of instances, generated from benchmark SALB3PM instances. The results for $\mathcal{M}^{PSC}$ with a 3600s time limit show that optimal solution are regularly found for small instances and, more generally, an average optimality gap of less than 2.7% can be achieved, even though for larger instances optimal, and even feasible, solutions seem much harder to find. For MS×LS^{PSC} the results are very interesting, with very stable behaviour and a gap w.r.t. the optimal solution on average under 1.6%, a good usage of additional resources, but also an inherently adaptive capability, with a higher number of generated solutions (diversification) for harder instances, and an intensification strategy when feasible solutions are found.
As to some managerial insights, three levers were considered to reduce power peak reduction in a Manufacturing System.
Resorting to optimisation tools is clearly the most important one, as both the exact, and most of all the matheuristic approach, seem promising. In particular, it has been shown that the power peak can be reduced by around 33% by adopting the proposed matheuristic MS×LS^{PSC}, all other production costs being constant – namely, production pace and number of workstations, although this reduction could require to have intermediate idle times between tasks, hence to trigger the processing of tasks with greater precision. It is also interesting to note that about 66% of the aforementioned power peak reduction could be achieved by the greedy heuristic decoder used by MS×LS^{PSC}, and around 90% by the other, ILP-based, decoder of MS×LS^{PSC}, ultimately confirming the convenience of adopting optimisation tools. The second lever is the adoption of parallel-serial configurations with crossover, which allows significant power peak reductions to be achieved in an existing MSs without requiring additional resources. The last lever is to actually invest in additional resources, even though the achieved power peak reduction could not be worth the associated extra-cost.
This work opens up some very interesting and promising research paths. Testing alternative ILP formulations, e.g. inspired by disjunctive formulations for production scheduling problems, and comparing the results obtained to those presented here is certainly of interest, in order to assess the more suitable modelling technique. From an algorithmic perspective, it would certainly be interesting to develop more advanced exact approaches, based on the study of problem-tailored data preprocessing routines, valid inequalities or branching rules, to be able to solve larger-sized instances exactly. Moreover, to conduct a deeper analysis of the behaviour of the proposed matheuristic, it would be interesting to test a version in which the stopping criterion is not a fixed number of iteration but instead an adaptive one, i.e. depending on the status of the search and the size of the instance.
The matheuristic can certainly also benefit from various improvements, from the reduction of the time spent in adding resources – the option that seems less viable from an economic viewpoint, to the refinement consisting in repairing, instead of penalising, infeasible solutions.
Finally, the study of the PSCALB3PM also seems to suggest some possible generalisations, i.e. the

consideration of variable task power consumption profiles; a more general setting, e.g. a line in which tasks on parallel resources can be scheduled differently, or a mixed/multi-model MS; a multi-objective extension of the problem, in order to find trade-offs between power peak and some production-related objective, for instance the takt time.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work is supported by the project ANR-21-CE10-0019 "RECONFIDURABLE".

Data Availability statement

The datasets that support the findings of this study are available from the corresponding author, [PG], upon reasonable request.

References

Artigues, Christian, Pierre Lopez, and Alain Haït. 2013. "The energy scheduling problem: Industrial case-study and constraint propagation techniques." International Journal of Production Economics 143 (1): 13–23. https://doi.org/10.1016/j.ijpe.2010.09.030.

Bänsch, Kristian, Jan Busse, Frank Meisel, Julia Rieck, Sebastian Scholz, Thomas Volling, and Matthias G. Wichmann. 2021. "Energy-aware decision support models in production environments: A systematic literature review." Computers & Industrial Engineering 159: 107456. https://doi.org/10.1016/j.cie.2021.107456.

Battaïa, Olga, Lyes Benyoucef, Xavier Delorme, Alexandre Dolgui, and Simon Thevenin. 2020. "Sustainable and Energy Efficient Reconfigurable Manufacturing Systems." In Reconfigurable Manufacturing Systems: From Design to Implementation, 179–191. Cham, Switzerland: Springer. https://doi.org/10.1007/978-3-030-28782-5_9.

Battaïa, Olga, and Alexandre Dolgui. 2013. "A taxonomy of line balancing problems and their solution approaches." International Journal of Production Economics 142: 259–277. https://doi.org/10.1016/j.ijpe.2012.10.020.

Borba, Leonardo, Marcus Ritt, and Cristóbal Miralles. 2018. "Exact and heuristic methods for solving the Robotic Assembly Line Balancing Problem." European Journal of Operational Research 270: 146–156. https://doi.org/10.1016/j.ejor.2018.03.011.

Borisovsky, Pavel A., Xavier Delorme, and Alexandre Dolgui. 2014. "Balancing reconfigurable machining lines via a set partitioning model." International Journal of Production Research 52 (13): 4026–4036. https://doi.org/10.1080/00207543.2013.849857.

Bortolini, Marco, Francesco Gabriele Galizia, and Cristina Mora. 2018. "Reconfigurable manufacturing systems: Literature review and research trend." Journal of manufacturing systems 49: 93–106. https://doi.org/10.1016/j.jmsy.2018.09.005.

Bowman, Edward H. 1959. "The schedule-sequencing problem." Operations Research 7 (5): 621–624. https://doi.org/10.1287/opre.7.5.621.

Boysen, Nils, Philipp Schulze, and Armin Scholl. 2022. "Assembly line balancing: What happened in the last fifteen years?" European Journal of Operational Research 301 (3): 797–814. https://doi.org/10.1016/j.ejor.2021.11.043.

Bruzzone, Alessandro A.G., Davide Anghinolfi, Massimo Paolucci, and Flavio Tonelli. 2012. "Energy-aware scheduling for improving manufacturing process sustainability: A mathematical model for flexible flow shops." CIRP Annals - Manufacturing Technology 61 (1): 459–462. https://doi.org/10.1016/j.cirp.2012.03.084.

Buxey, G.M. 1974. "Assembly line balancing with multiple stations." Management science 20 (6): 1010–1021. https://doi.org/10.1287/mnsc.20.6.1010.

Carlucci, Daniela, Paolo Renna, and Sergio Materi. 2021. "A Job-Shop Scheduling Decision-Making Model for Sustainable Production Planning With Power Constraint." IEEE Transactions on Engineering Management https://doi.org/10.1109/TEM.2021.3103108.

Cerqueus, Audrey, and Xavier Delorme. 2019. "A branch-and-bound method for the bi-objective simple line assembly balancing problem." International Journal of Production Research 57 (18): 5640–5659. https://doi.org/10.1080/00207543.2018.1539266.

Cerqueus, Audrey, and Xavier Delorme. 2023. "Evaluating the scalability of reconfigurable manufacturing systems at the design phase." International Journal of Production Research https://doi.org/10.1080/00207543.2022.2164374.

Dahmani, Abdelhak, Lyes Benyoucef, and Jean-Marc Mercantini. 2022. "Toward Sustainable Reconfigurable Manufacturing Systems (SRMS): Past, Present, and Future." Procedia Computer Science 200: 1605–1614. https://doi.org/10.1016/j.procs.2022.01.361.

Delorme, Xavier, Audrey Cerqueus, Paolo Gianessi, and Damien Lamy. 2023. "RMS balancing and planning under uncertain demand and energy cost considerations." International Journal of Production Economics 261: 108873. https://doi.org/10.1016/j.ijpe.2023.108873.

Delorme, Xavier, and Paolo Gianessi. 2022. "Designing Reconfigurable Manufacturing Systems to Minimize Power Peak." IFAC-PapersOnLine 55 (10): 1296–1301. https://doi.org/10.1016/j.ifacol.2022.09.569.

Dubey, Rameshwar, Angappa Gunasekaran, Petri Helo, Thanos Papadopoulos, Stephen J. Childe, and B.S. Sahay. 2017. "Explaining the impact of reconfigurable manufacturing systems on environmental performance: The role of top management and organizational culture." Journal of cleaner production 141: 56–66. https://doi.org/10.1016/j.jclepro.2016.09.035.

Essafi, Mohamed, Xavier Delorme, Alexandre Dolgui, and Olga Guschinskaya. 2010. "A MIP approach for balancing transfer line with complex industrial constraints." Computers & Industrial Engineering 58 (3): 393–400. https://doi.org/10.1016/j.cie.2009.04.009.

European Central Bank. 2022. "The impact of the war in Ukraine on euro area energy markets." https://www.ecb.europa.eu/pub/economic-bulletin/focus/html/index.en.html.

European Commission. 2021. SET Plan Action 6 on Energy Efficiency in Industry. Technical Report. https://setis.ec.europa.eu/publications_en.

Fakih, Ali, Pascal Ghazalian, and Nancy Ghazzawi. 2020. "The effects of power outages on the performance of manufacturing firms in the MENA region." Review of Middle East Economics and Finance 16 (3). https://doi.org/10.1515/rmeef-2020-0011.

Fang, Kan, Nelson A. Uhan, Fu Zhao, and John W. Sutherland. 2013. "Flow shop scheduling with peak power consumption constraints." Annals of Operations Research 206 (1): 115–145. https://doi.org/10.1007/s10479-012-1294-z.

Fang, Yilin, Quan Liu, Miqing Li, Yuanjun Laili, and Duc Truong Pham. 2019. "Evolutionary many-objective optimization for mixed-model disassembly line balancing with multi-robotic workstations." European Journal of Operational Research 276 (1): 160–174. https://doi.org/10.1016/j.ejor.2018.12.035.

Fernandes, João M.R.C., Seyed Mahdi Homayouni, and Dalila Benedita M. Martins Fontes. 2022. "Energy-Efficient Scheduling in Job Shop Manufacturing Systems: A Literature Review." Sustainability 14 (10): 6264. https://doi.org/10.3390/su14106264.

Freiheit, Theodor, Moshe Shpitalni, and S. Jack Hu. 2004. "Productivity of paced parallel-serial manufacturing lines with and without crossover." Journal of Manufacturing Science and Engineering 126 (2): 361–367. https://doi.org/10.1115/1.1688372.

Ghanei, Shima, and Tarek AlGeddawy. 2020. "An integrated multi-period layout planning and scheduling model for sustainable reconfigurable manufacturing systems." Journal of Advanced Manufacturing Systems 19 (01): 31–64. https://doi.org/10.1142/S0219686720500031.

Ghobakhloo, Morteza, and Masood Fathi. 2021. "Industry 4.0 and opportunities for energy sustainability." Journal of Cleaner Production 295: 126427. https://doi.org/10.1016/j.jclepro.2021.126427.

Gianessi, Paolo, Xavier Delorme, and Oussama Masmoudi. 2019. "Simple Assembly Line Balancing Problem with Power Peak Minimization." In Advances in Production Management Systems, edited by Farhad Ameri, Kathryn E. Stecke, Gregor von Cieminski, and Dimitris Kiritsis, Vol. 566, 239–247. Cham: Springer Int. Publishing. https://doi.org/10.1007/978-3-030-30000-5_31.

Glover, Fred, and Gary A. Kochenberger. 2003. Handbook of Metaheuristics. New York, NY: Springer US. https://doi.org/10.1007/b101874.

Gondran, Matthieu, Sylverin Kemmoe, Damien Lamy, and Nikolay Tchernev. 2020. "Bi-objective optimisation approaches to Job-shop problem with power requirements." Expert Systems with Applications 162: 113753. https://doi.org/10.1016/j.eswa.2020.113753.

Gourgand, Michel, Nathalie Grangeon, and Sylvie Norre. 2007. "Metaheuristics based on bin packing for the line balancing problem." RAIRO-Operations Research 41 (2): 193–211. https://doi.org/10.1051/ro:2007018.

Haapala, Karl R., Fu Zhao, Jaime Camelio, John W. Sutherland, Steven J. Skerlos, David A. Dornfeld, Ibrahim S. Jawahir, Andres F. Clarens, and Jeremy L. Rickli. 2013. "A review of engineering research in sustainable manufacturing." Journal of manufacturing science and engineering 135 (4). https://doi.org/10.1115/1.4024040.

Huang, Aihua, Fazleena Badurdeen, and Ibrahim S. Jawahir. 2018. "Towards developing sustainable reconfigurable manufacturing systems." Procedia manufacturing 17: 1136–1143. https://doi.org/10.1016/j.promfg.2018.10.024.

International Energy Agency. 2017. Tracking Clean Energy Progress 2017: Industry. Technical Report. https://www.iea.org/reports/tracking-clean-energy-progress-2017.

International Energy Agency. 2021. World Energy Outlook 2021. Technical Report. https://www.iea.org/reports/world-energy-outlook-2021.

International Energy Agency. 2022a. Electricity Market Report – July 2022 – Update. Technical Report.

https://www.iea.org/reports/electricity-market-report-july-2022.
International Energy Agency. 2022b. "The world's coal consumption is set to reach a new high in 2022 as the energy crisis shakes markets." https://www.iea.org/news/the-world-s-coal-consumption-is-set-to-reach-a-new-high-in-2022-as-the-energy-crisis-shakes-markets.
Jamwal, Anbesh, Rajeev Agrawal, Monica Sharma, and Antonio Giallanza. 2021. "Industry 4.0 Technologies for Manufacturing Sustainability: A Systematic Review and Future Research Directions." Applied Sciences 11 (12). https://doi.org/10.3390/app11125725.
Janardhanan, Mukund Nilakantan, George Guoquan Huang, and Sivalinga Govindarajan Ponnambalam. 2015. "An investigation on minimizing cycle time and total energy consumption in robotic assembly line systems." Journal of Cleaner Production 90 (2): 311–325. https://doi.org/10.1016/j.jclepro.2014.11.041.
Kawaguchi, Shuhei, and Yoshikazu Fukuyama. 2016. "Reactive Tabu Search for Job-shop scheduling problems considering peak shift of electric power energy consumption." In 2016 IEEE Region 10 Conference (TENCON), Singapore, 3406–3409. IEEE. https://doi.org/10.1109/TENCON.2016.7848686.
Kemmoe, Sylverin, Damien Lamy, and Nikolay Tchernev. 2017. "Job-shop like manufacturing system with variable power threshold and operations with power requirements." International Journal of Production Research 55 (20): 6011–6032. https://doi.org/10.1080/00207543.2017.1321801.
Khezri, Amirhossein, Hichem Haddou Benderbal, and Lyes Benyoucef. 2021. "Towards a sustainable reconfigurable manufacturing system (SRMS): multi-objective based approaches for process plan generation problem." International Journal of Production Research 59 (15): 4533–4558. https://doi.org/10.1080/00207543.2020.1766719.
Koren, Yoram, Xi Gu, Fazleena Badurdeen, and Ibrahim S. Jawahir. 2018. "Sustainable living factories for next generation manufacturing." Procedia Manufacturing 21: 26–36. https://doi.org/10.1016/j.promfg.2018.02.091.
Koren, Yoram, Uwe Heisel, Francesco Jovane, Toshimichi Moriwaki, Gumter Pritschow, Galip Ulsoy, and Hendrik Van Brussel. 1999. "Reconfigurable manufacturing systems." CIRP Annals 48: 2. https://doi.org/10.1016/S0007-8506(07)63232-6.
Koren, Yoram, Wencai Wang, and Xi Gu. 2017. "Value creation through design for scalability of reconfigurable manufacturing systems." International Journal of Production Research 55 (5): 1227–1242. https://doi.org/10.1080/00207543.2016.1145821.
Kovalev, Sergey, Xavier Delorme, Alexandre Dolgui, and Ammar Oulamara. 2017. "Minimizing the number of stations and station activation costs for a production line." Computers & Operations Research 79: 131–139. https://doi.org/10.1016/j.cor.2016.10.007.
Lahrichi, Youssef, Laurent Deroussi, Nathalie Grangeon, and Sylvie Norre. 2021. "A balance-first sequence-last algorithm to design RMS: a matheuristic with performance guaranty to balance reconfigurable manufacturing systems." Journal of Heuristics 27 (1): 107–132. https://doi.org/10.1007/s10732-021-09473-1.
Lamy, Damien, Xavier Delorme, and Paolo Gianessi. 2020. "Line Balancing and Sequencing for Peak Power Minimization." IFAC-PapersOnLine 53 (2): 10411–10416. https://doi.org/10.1016/j.ifacol.2020.12.2781.
Lawrence, Akvile, Patrik Thollander, Mariana Andrei, and Magnus Karlsson. 2019. "Specific Energy Consumption/Use (SEC) in Energy Management for Improving Energy Efficiency in Industry: Meaning, Usage and Differences." Energies 12 (2): 247. https://doi.org/10.3390/en12020247.
Li, Zixiang, Qiuhua Tang, and LiPing Zhang. 2016. "Minimizing energy consumption and cycle time in two-sided robotic assembly line systems using restarted simulated annealing algorithm." Journal of Cleaner Production 135: 508–522. https://doi.org/10.1016/j.jclepro.2016.06.131.
Liang, Junyong, Shunsheng Guo, Baigang Du, Yibing Li, Jun Guo, Zhijie Yang, and Shibao Pang. 2021. "Minimizing energy consumption in multi-objective two-sided disassembly line balancing problem with complex execution constraints using dual-individual simulated annealing algorithm." Journal of Cleaner Production 284: 125418. https://doi.org/10.1016/j.jclepro.2020.125418.
Liu, Rongfan, Ming Liu, Feng Chu, Feifeng Zheng, and Chengbin Chu. 2021. "Eco-friendly multi-skilled worker assignment and assembly line balancing problem." Computers & Industrial Engineering 151: 106944. https://doi.org/10.1016/j.cie.2020.106944.
Lourenço, Helena R., Olivier C. Martin, and Thomas Stützle. 2003. "Iterated Local Search." In Handbook of Metaheuristics, edited by Fred Glover and Gary A. Kochenberger, 320–353. New York, NY: Springer US. https://doi.org/10.1007/0-306-48056-5_11.
Masmoudi, Oussama, Xavier Delorme, and Paolo Gianessi. 2019. "Job-shop scheduling problem with energy consideration." International Journal of Production Economics 216: 12–22. https://doi.org/10.1016/j.ijpe.2019.03.021.
Masmoudi, Oussama, Alice Yalaoui, Yassine Ouazene, and Hicham Chehade. 2017. "Lot-sizing in a multi-stage flow line production system with energy consideration." International Journal of Production Research 55 (6): 1640–1663. https://doi.org/10.1080/00207543.2016.1206670.
Massimi, Elisa, Amirhossein Khezri, Hichem Haddou Benderbal, and Lyes Benyoucef. 2020. "A heuristic-based non-linear mixed integer approach for optimizing modularity and integrability in a sustainable reconfigurable manufacturing environment." The International Journal of Advanced Manufacturing Technology 108 (7):

1997–2020. https://doi.org/10.1007/s00170-020-05366-y.
Menghi, Roberto, Alessandra Papetti, Michele Germani, and Marco Marconi. 2019. “Energy efficiency of manufacturing systems: A review of energy assessment methods and tools.” Journal of Cleaner Production 240: 118276. https://doi.org/10.1016/j.jclepro.2019.118276.
Módos, István, Přemysl Šucha, and Zdeněk Hanzálek. 2021. “On parallel dedicated machines scheduling under energy consumption limit.” Computers & Industrial Engineering 159: 107209. https://doi.org/10.1016/j.cie.2021.107209.
Mohamed, Nader, Jameela Al-Jaroodi, and Sanja Lazarova-Molnar. 2019. “Leveraging the Capabilities of Industry 4.0 for Improving Energy Efficiency in Smart Factories.” IEEE Access 7: 18008–18020. https://doi.org/10.1109/ACCESS.2019.2897045.
Morgan, Jeff, Mark Halton, Yuansong Qiao, and John G. Breslin. 2021. “Industry 4.0 smart reconfigurable manufacturing machines.” Journal of Manufacturing Systems 59: 481–506. https://doi.org/10.1016/j.jmsy.2021.03.001.
Nature. 2022. “What the war in Ukraine means for energy, climate and food.” https://www.nature.com/articles/d41586-022-00969-9.pdf.
Pape, Tom. 2015. “Heuristics and lower bounds for the simple assembly line balancing problem type 1: Overview, computational tests and improvements.” European Journal of Operational Research 240: 32–42. https://doi.org/10.1016/j.ejor.2014.06.023.
Pinto, Peter A., David G. Dannenbring, and Basheer M. Khumawala. 1981. “Branch and bound and heuristic procedures for assembly line balancing with paralleling of stations.” International Journal of Production Research 19 (5): 565–576. https://doi.org/10.1080/00207548108956687.
Putnik, Goran D., Alojzij Sluga, Hoda A. ElMaraghy, Roberto Teti, Yoram Koren, Tullio A.M. Tolio, and Bernard K.K. Hon. 2013. “Scalability in manufacturing systems design and operation: State-of-the-art and future developments roadmap.” CIRP Annals 62 (2): 751–774. https://doi.org/10.1016/j.cirp.2013.05.002.
Renna, Paolo, and Sergio Materi. 2021. “A Literature Review of Energy Efficiency and Sustainability in Manufacturing Systems.” Applied Sciences 11 (16). https://doi.org/10.3390/app11167366.
Scholl, Armin. 1999. Balancing and sequencing of assembly lines. Heidelberg, Germany: Physica-Verlag.
Sewell, Edward C., and Sheldon H. Jacobson. 2012. “A branch, bound, and remember algorithm for the simple assembly line balancing problem.” INFORMS Journal on Computing 24 (3): 433–442. https://doi.org/10.1287/ijoc.1110.0462.
Talbi, El-Ghazali. 2009. Metaheuristics: from design to implementation. Hoboken, NJ: John Wiley & Sons. https://doi.org/10.1002/9780470496916.
United Nations. 2022. Global impact of war in Ukraine: Energy crisis. Technical Report. https://unctad.org/publication/global-impact-war-ukraine-energy-crisis.
U.S. Energy Information Administration. 2019. International Energy Outlook 2019 with projections to 2050. Technical Report. https://www.eia.gov/outlooks/ieo/pdf/ieo2019.pdf.
Wang, Kaipu, Xinyu Li, Liang Gao, and Peigen Li. 2020. “Energy consumption and profit-oriented disassembly line balancing for waste electrical and electronic equipment.” Journal of Cleaner Production 265: 121829. https://doi.org/10.1016/j.jclepro.2020.121829.
World Bank. 2023. “Enterprise Surveys – Infrastructure.” https://www.enterprisesurveys.org/en/data/explore topics/infrastructure.
Zhang, Beikun, Liyun Xu, and Jian Zhang. 2020. “Developing mathematical model and optimization algorithm for designing energy efficient semi-automated assembly line.” Computers & Industrial Engineering 149: 106768. https://doi.org/10.1016/j.cie.2020.106768.
Zhang, Zikai, Qiuhua Tang, Zixiang Li, and Liping Zhang. 2019. “Modelling and optimisation of energy-efficient U-shaped robotic assembly line balancing problems.” International Journal of Production Research 57 (17): 5520–5537. https://doi.org/10.1080/00207543.2018.1530479.