A new Decoder for Permutation-based Heuristics to Minimize Power Peak in the Assembly Line Balancing

Xavier Delorme, Paolo Gianessi, Damien Lamy

Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 56-2 (2023) 3704–3709

A new Decoder for Permutation-based Heuristics to Minimize Power Peak in the Assembly Line Balancing

Xavier Delorme *, **Paolo Gianessi** * **Damien Lamy** *

* *Mines Saint-Etienne, Univ Clermont Auvergne, INP Clermont Auvergne, CNRS, UMR 6158 LIMOS, F-42023 Saint-Etienne, France (e-mail: (delorme, paolo.gianessi, damien.lamy)@emse.fr)*

Abstract: We consider the Simple Assembly Line Balancing Problem with Power Peak Minimization and Earliest Starting Dates (SALB3PM-ESD), a problem of balancing an assembly line and suitably sequencing its tasks so as to minimize the peak of the electric power consumption associated with them. We propose an ILP-based decoder to optimally split (w.r.t. the power peak) a sequence of tasks over the workstations of the line. The decoder is plugged into a simple local search algorithm to test its effectiveness in quickly computing the optimal split for the solutions encountered in the space of task sequences. Preliminary tests on instances from literature show that the decoder is efficient, and that it seems indeed promising to use it to take advantage of the numerous sequence-based optimization algorithms in the scheduling literature to develop more competitive methods to efficiently tackle the SALB3PM-ESD.

Keywords: Assembly Line Balancing, Energy Efficiency, Sustainable Manufacturing, Power Peak, Integer Linear Programming, Advanced Planning and Scheduling, Operations Research.

1. INTRODUCTION

The level attained in recent years by the main climate change indicators has made the transition to renewable energy sources more and more urgent. This, along with the international context, is causing the prices of energy to grow considerably, and pushing organizational changes towards energy conservation (International Energy Agency, 2022). Industry is strongly concerned, being accountable for more than 50% of the end-use energy consumption (Energy Information Administration, 2019), with undeniable consequences in terms of environmental impact, hence companies are more and more asked to be energy-efficient. This is also true specifically for what concerns electricity: according to the International Energy Agency (International Energy Agency, 2021), 22% of the total final electricity consumption in 2020 is due to the industrial sector, and this figure is expected to grow to 46% by 2050 due to the electrification of several industrial processes. Energy efficiency in electricity usage affects industrial systems from design through to management (Giret et al., 2015), and is generally addressed through various objectives and/or constraints, such as total energy consumption (Zhang and Chiong, 2016), total energy cost w.r.t. Time-Of-Use (TOU) pricing (Luo et al., 2013) and power peak limitation (Bruzzone et al., 2012). If on the one hand the literature accounts for a large number of scientific efforts in this sense (Akbar and Irohara, 2018), few are still the works that tackle the minimization of the peak of the electric power consumption. Such an objective allows to smoothen the energy consumption of an industrial system, hence to reduce operating costs, avoid penalties for exceeding a maximum allowed peak of power usage, or face the more and more volatile availability of electric energy, mainly due to the growing focus on renewable energy sources (Battaïa et al., 2020). Minimizing the power peak is even more important in paced production lines where such a peak is repeated at each cycle. It is then of great importance to consider electric power peak since the design stage of a production system – a purpose that seemingly very few papers have tried to address. Gianessi et al. (2019) defines the Simple Assembly Line Balancing Problem with Power Peak Minimization (SALB3PM), to the best of our knowledge the first example of Line Balancing (LB) problem with minimization of peak power. In order to consider power peak minimization in the design of paced production lines, the SALB3PM integrates the scheduling of tasks on workstations, other than classical balancing constraints. Lamy et al. (2020) studies a special case, which we refer to in the following as SALB3PM with Earliest Starting Dates (SALB3PM-ESD), where tasks must be executed as early as possible. This better fits manual or semi-automated contexts, in which idle times between tasks are difficult to manage. This paper focuses on the SALB3PM-ESD.

Figure 1 shows on an example how the scheduling decisions of the SALB3PM-ESD can considerably impact the power peak. In it, x-axes represents time; the instance has $n = 9$ tasks, depicted as boxes whose width, height and horizontal position represent, respectively, the processing time, the power consumption and the starting date. Each subfigure shows the schedule of the $m = 3$ workstations: the left y-axes represent the power consumed by each workstation, while thick blue lines and dashed red lines, respectively, show the overall power consumption profile and highlight the power peak w.r.t. the rightmost y-axis. In the left subfigure, only balancing decisions have been taken, i.e.

10.1016/j.ifacol.2023.10.1537

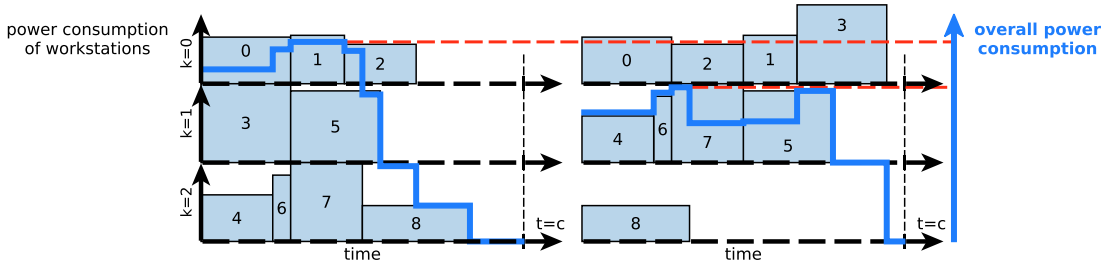

Fig. 1. Simple schedule (left) vs SALB3PM-ESD optimal solution (right) for a benchmark instance.

only precedence constraints are complied with, and at a later time tasks have been scheduled at the earliest possible date. The right subfigure is the SALB3PM-ESD optimum, for which the power peak decrease is considerable (-22%). In this paper we design an Integer Linear Program (ILP) to evaluate a *permutation* of the production tasks of a balancing line in terms of the induced electric power peak in the SALB3PM-ESD case. Several works exist in which scheduling problems are tackled by representing solutions as permutations of the tasks; permutation are transformed into assignments of tasks to machines by a *decoder*, which then computes the corresponding objective function value. The choice of modeling a feasible assignment via a permutation can be induced either by the specific problem, e.g. the permutation flowshop problem (Ashour, 1970), or by a particular approach, e.g. a local search (LS) metaheuristic or genetic algorithm (GA), (e.g. Li and Gao, 2016). Thus, using such a coding for the SALB3PM-ESD would allow to take advantage of the various optimization algorithms proposed in the scheduling literature. The related combinatorics is hopefully simple enough to be quickly dealt with by ILP. The resulting decoder could then be plugged into metaheuristics for the SALB3PM-ESD based on a permutation-based representation to efficiently evaluate a solution e.g. during a LS, or a chromosome during a GA. The rest of the paper is organized as follows. Section 2 provides a short literature review. Section 3 describes the ILP model of the proposed decoder for the SALB3PM-ESD. Section 4 presents an illustrative metaheuristic using the decoder and some preliminary results. Section 5 concludes this work and presents future research perspectives.

2. LITERATURE REVIEW

Since the introduction of the Simple Assembly Line Balancing Problem (SALBP) (Baybars, 1986), Line Balancing (LB) problems represent the class of optimization problems devoted to the design of production systems. The SALBP is the archetypal such problem: the production system is a paced, single-product line, and tasks have deterministic processing times, are independent of workstations, and subject to precedence constraints. Depending on whether the goal is the minimum number m of workstations given the takt time c, or the inverse, we have the two main variants, SALBP-1 and SALBP-2. SALBP is NP-hard and still a challenging problem for which best known approaches date back to less than a decade (Cerqueus and Delorme, 2019; Pape, 2015; Sewell and Jacobson, 2012). Many generalizations and rich versions have been considered to meet realistic applications, giving rise to a huge literature of LB problems (Boysen et al., 2022). However, to the best of our knowledge, few works jointly consider LB and sequencing, most often to take into account sequence-dependent setup times, e.g. Özcan (2019) addresses the balancing and scheduling of parallel assembly lines, providing an exact and a simulated annealing (SA) algorithm. Energy efficiency in LB problems is another aspect that, as far as the authors are aware of, have not received the deserved attention from the research community. Indeed, most existing papers on the subject seemingly focus on assembly line design problems, where special equipment must be assigned to workstations in order to perform production tasks, and the minimization of the associated energy consumption is sought, possibly along with other objectives. Robotic ALB problems (RALBPs, Borba et al., 2018) represent the most prominent example of this stream. For instance, Zhang et al. (2019) tackles the multi-objective problem of designing energy-efficient U-shaped robotic assembly lines, while considering productivity. A recent example of LB problem integrating energy-related criteria, other than RALBPs, is Wang et al. (2020), which studies the problem of a disassembly line for waste electrical/electronic equipment disposal, showing that disassembly profit can improve while reducing energy consumption. The only problems dealing with power peak are the SALB3PM and the SALB3PM-ESD, mentioned in Section 1. In Gianessi et al. (2019) a first time-indexed ILP formulation is introduced for the SALB3PM to tackle 38 benchmark SALBP-1 and SALBP-2 instances (Scholl, 1995) enhanced with randomly generated task power consumption values. The ILP model can solve to optimality instances with up to 25 tasks, 10 workstations and a takt time of 40 within a one hour time limit, but struggles to find optimal, or even feasible, solutions for bigger instances. Lamy et al. (2020) proposes a Multi-Start Evolutionary Local Search (MS×ELS) for the SALB3PM-ESD, tested on a larger instance set than Gianessi et al. (2019). Nevertheless, power peak constraints and/or objectives can be found more frequently in research works that tackle other production-related problems, e.g. in scheduling. Bruzzone et al. (2012) proposes a MILP and a heuristic approach to minimize power peak in a flexible flowshop. A jobshop with a variable power threshold is addressed by Kemmoe et al. (2017) and several metaheuristics are designed. Masmoudi et al. (2019) minimizes energy cost w.r.t. to TOU energy prices in a jobshop problem under makespan and maximum allowed power peak constraints. To conclude, we mention some works on scheduling problems in which a permutation-based representation of solutions is used. Ruiz and Maroto (2005) provides an extended review of heuristic and metaheuristic approaches for the permutation flowshop scheduling problem (PFSP), many of which make use of a solution coding as a per-

mutation of the production tasks. Lange and Werner (2019) tackles the blocking jobshop scheduling problem with total tardiness minimization: a SA framework is proposed which uses, among others, a permutation-based coding of the solution and some related neighborhood structures. Pan et al. (2022) tackles a distributed lot-streaming PFSP to minimize the makespan, and propose a range of population-based metaheuristics which consider a solution as a permutation of jobs. Finally, we cite two works featuring many elements in common with the work presented here as both concern ALB problems and make use of a permutation-based coding of solutions, specifically because both deal with sequence-dependent setup times. Lahrichi et al. (2022) tackles the SALBP-2 variant of the RALBP, and delves into the special case in which a unique sequence of tasks is given and split via a polynomial optimal algorithm, in order to explore the space of such sequences; Borisovsky et al. (2013) studies the balancing of reconfigurable machining lines and proposes a GA in which chromosomes are task permutations, mapped into feasible solutions by either a greedy heuristic decoder or a MILP-based one. Both works, however, differ from the LB problem studied here, in which decisions about task sequencing are needed to minimize the power peak.

3. MATHEMATICAL MODEL OF THE DECODER

In this section, after defining suitable notations, we provide an Integer Linear Programming model as a decoder for the SALB3PM-ESD. Such a decoder will allow to transform a permutation of the n tasks of a production process into an SALB3PM-ESD solution, i.e. an assignment and sequencing of the tasks over the m workstations.
Let $\mathcal{O}$ be a set of n production tasks and $\mathcal{M} = \{0, ..., m-1\}$ the set of available workstations. The set of tasks comes with a given *sequence* $(p(0), ..., p(n-1))$: p is a permutation $\{0, ..., n-1\} \rightarrow \mathcal{O}$ that defines how the tasks must be sequenced on the sequence of the workstations, and cannot be changed. By abuse of notation, we use symbol $\mathcal{O} = (0, ..., n-1)$ to refer also to the given sequence. As in the SALB3PM-ESD, tasks have processing times t_j and power consumption values w_j, and precedence constraints exist among them. However, the sequence $\mathcal{O}$ is supposed to be precedence-compliant: a task i is a predecessor (direct or not) of another task j (noted $i \prec j$ if direct) if and only if $i < j$, i.e. $\mathcal{O}$ is a valid topological ordering of the tasks. The takt time of the line, c, is also known, so that tasks must all be run within a time horizon $\mathcal{T} = \{0, ..., c-1\}$. The objective is to decide how to *split* the sequence $\mathcal{O}$ into as many subsequences $\mathcal{O}_0 = (0, ..., i_0), \mathcal{O}_1 = (i_0 + 1, ..., i_1), ..., \mathcal{O}_{m-1} = (i_{m-2}+1, ..., i_{m-1})$ as the number m of workstations, in such a way that:

- the assignment $\forall k \in \mathcal{M}$ of the tasks of subsequence $\mathcal{O}_k$ to workstation k according to the order defined by $\mathcal{O}_k$ (hence by $\mathcal{O}$), and the schedule of each task at the earliest possible date, gives rise to a feasible SALB3PM-ESD solution, i.e. $(\forall k \in \mathcal{M}) \sum_{j \in \mathcal{O}_k} t_j \leq c$
- the peak of the overall power profile is minimized

The decisions can be modeled by two sets of binary decisions variables and $c+1$ integer variables:

- *split* variables L_i, $i \in \mathcal{O}$, $L_i = 1 \Leftrightarrow$ task i is the last one processed on a workstation;
- *subsequence* variables $L'_{i,j}$, $i, j \in \mathcal{O}$, $L'_{i,j} = 1 \Leftrightarrow$ task i is the last processed on its workstation and j is assigned to the following one but not as the last. We define $L'_{i,j}$ only if $\sum_{h=i+1}^{j+1} t_h \leq c$, otherwise having $L_i = 1$ and $L_j = 0$ is not possible. We note $\mathcal{P}$ the set of pairs of tasks i, j for which $L'_{i,j}$ is defined;
- *slot-power* variables $\overline{W}_t$, $t \in \mathcal{T}$, the overall power consumption at date t;
- *power-peak* variable W_M.

The proposed ILP model is then the following:

$$\min W_M \tag{1}$$

$$\text{s.t.} \sum_{i \in \mathcal{O}} L_i = m \quad ; \quad L_{n-1} = 1 \tag{2}$$

$$\sum_{i \leq j < \min\{h \in \mathcal{O} : \sum_{g=i}^{h} t_g > c\}} L_j \geq 1 \qquad \forall i \in \mathcal{O} \tag{3}$$

$$L'_{i,j} \leq L_i \qquad \forall (i,j) \in \mathcal{P} \tag{4}$$

$$L'_{i,j} \leq 1 - \sum_{(i+1)\,\%\,n \leq h \leq j} L_h \qquad \forall (i,j) \in \mathcal{P} \tag{5}$$

$$L'_{i,j} \geq L_i - \sum_{(i+1)\,\%\,n \leq h \leq j} L_h \qquad \forall (i,j) \in \mathcal{P} \tag{6}$$

$$\overline{W}_t \leq W_M \qquad \forall t \in \mathcal{T} \tag{7}$$

$$\overline{W}_0 = \sum_{i \in \mathcal{O}} w_{((i+1)\,\%\,n)} \cdot L_i \tag{8}$$

$$\begin{aligned} \overline{W}_t = \overline{W}_{t-1} &+ \sum_{(i,j) \in \mathcal{P} : \sum_{h=(i+1)\%n}^{j} t_h = t} w_{j+1} \cdot L'_{i,j} \\ &- \sum_{(i,j) \in \mathcal{P} : \sum_{h=(i+1)\%n}^{j+1} t_h = t} w_{j+1} \cdot L'_{i,j} \\ &- \sum_{i \in \mathcal{O} : t_{((i+1)\,\%\,n)} = t} w_{((i+1)\,\%\,n)} \cdot L_i \quad \forall t \in \mathcal{T} \setminus \{0\} \end{aligned} \tag{9}$$

$$L_i, L'_{i,j} \in \{0,1\}, \overline{W}_t, W_M \in \mathbb{Z}_+$$

Split decisions are thoroughly modeled by split variables, as their name suggest; slot-power variables and W_M allow to model the power profile following the split decisions and to smoothen it. As for subsequence variables, the value they take based on L_i variables allow to have complete information on how tasks that are not the last of the respective workstations are arranged on them. This, combined with the earliest starting date assumption of SALB3PM-ESD, allows to know at each date $t \in \mathcal{T}$ whether there are tasks that are triggered or ended, hence to know when and by how much the power profile changes.
Indeed, having $L'_{i,j} = 1$ not only means that i is the last task of the workstation k it is assigned to, i.e. the sequence $\mathcal{O}$ is split between i and $i+1$, but also that all tasks from $i+1$ to $j+1$ are on workstation $k+1$, since $(\forall h = i+1, ..., j)\ L_h = 0$ (see (4)-(6), in which % denotes the modulo operator). Hence, if $L'_{i,j} = 1$, the starting dates of tasks $i+1, i+2, i+3, ..., j+1$ are known and equal to, respectively, $t{=}0$, $t{=}t_{i+1}$, $t{=}t_{i+1}{+}t_{i+2}$, ..., $t{=}\sum_{h=i+1}^{j} t_h$. Clearly this can change with the split decisions, e.g. if $L_i{=}L_{i+1}{=}1$, then $i{+}1$ is the only task on workstation $k{+}1$, whose consumption is w_{i+1} for $t < t_{i+1}$, 0 for $t \geq t_{i+1}$.
Once these aspects related to subsequence variables are clear, understanding how model (1)-(9) describes the decoding of a sequence and the determination of the opti-

mal split is straightforward. Due to (2), the tasks that are the last of their workstations are exactly m, $n-1 \in \mathcal{O}$ being one of them, while (3) imposes the feasibility of a split, as it forbids the subsequence $(i+1, ..., i')$ defined by two successive tasks i and i' having $L_i = L_{i'} = 1$, if it exceeds the takt time, i.e. if $\sum_{h=i+1}^{i'} t_h > c$. Constraint (8) determines the power consumption at $t = 0$, $\overline{W}_0$, based on the fact that if $L_i = 1$, then i ends a subsequence, and $i+1$ is scheduled at $t = 0$ on the following workstation. Notice that modulo allows $i = n-1$ to act as if it was the predecessor of $i = 0$ in the sequence, hence to account for consumption w_0 on workstation $k = 0$ at date $t = 0$. Constraints (9) determine the power consumption at date $t > 0$ in incremental fashion based on that at date $t-1$: the difference $\overline{W}_t - \overline{W}_{t-1}$ is computed based on subsequence variables as explained before. The second term on the right-hand side is greater than 0 if : $\exists i, j \in \mathcal{O}$ s.t. $L_i = 1; (\forall\, h = i+1, ..., j+1)\, L_h = 0$ i.e. if j is on the subsequence after that closed by i; the considered t equals the total processing time of tasks $h = i+1, ..., j$, i.e. t is the starting date of $j+1$. Similar reasonings allow to see that the third term is only greater than 0 when a task ends, and the fourth compensate the first-task power consumption terms introduced by constraints (8). Again, the modulo operator allows to use task $n-1$ as the splitting task giving rise to subsequence $\mathcal{O}_0$. For instance in the right subfigure of Figure 1 the sequence $(0, 2, 1, 3, 4, 6, 7, 5, 8)$ is split into $(0, 2, 1, 3)$, $(4, 6, 7, 5)$, (8): we have $L_3 = L_7 = L_8 = 1$ (i.e. tasks 3, 5 and 8 in the figure) and $L'_{3,4} = L'_{3,5} = L'_{3,6} = 1$, allowing (8) to account w_4 (power consumption of task 4 in the figure) at $t = 0$, constraint (9) for $t = t_4$ to consider the increment $w_6 - w_4$ thank to $L'_{3,4} = 1$, and so on. Finally, (7) and the objective function (1) allow power peak minimization. From constraints (9), it is clearer that subsequence variables actually allow to linearize some products among split variables.

4. COMPUTATIONAL EXPERIMENTS

To have a preliminary assessment of the effectiveness of the decoder, we plug it into a simple metaheuristic algorithm, which we test on a small set of SALB3PM-ESD instances. A neighborhood operator N is defined as follows. A neighbor of sequence $s = (p(0), ..., p(n-1))$ is obtained by:

- choosing $j \in \{0, ..., n-1\}$
- finding h and k s.t. $p(h)$ and $p(k)$ are, respectively, the closest direct predecessor and direct successor of $p(j)$ in the sequence, i.e. $h = \max\{g : 0 \leq g < j, p(g) \prec p(j)\}$, $k = \min\{g : j < g \leq n-1, p(j) \prec p(g)\}$
- removing task $p(j)$ from its current position, j, and inserting it in position $g \in \{h+1, ..., j-1, j+1, ..., k-1\}$
- shifting forward tasks $p(g), ..., p(j-1)$ of one position if $g < j$, or shifting backward tasks $p(j+1), ..., p(g)$ of one position if $j < g$

If s complies with precedence constraints (as we supposed in Section 3), then the compliance of the obtained neighbor sequence is guaranteed. Figure 2 gives an example of neighborhood move in the case of the benchmark SALB3PM-ESD instance bowman-1 and a possible task permutation. In both subfigures, tasks are numbered according to both the original instance task indices and the corresponding permutation indices (in smaller, dark blue boxes), e.g. on the right subfigure $p(3) = 4$ and $p(4) = 6$. Hence, the permutation on the left is $(0, 1, 2, 3, 4, 6, 5, 7)$. By picking $j = 3$ in it, task $p(3) = 3$ (green) is chosen whose closest direct predecessor and successor (orange) are, respectively, $p(1) = 1$ and $p(6) = 5$, so we have to choose a new position $g \in \{2, 4, 5\}$. The right subfigure corresponds to $g = 5$, which leads to the new sequence $(0, 1, 2, 4, 6, 3, 5, 7)$.

Figure 2 also shows how going from one sequence to a neighbor one impacts the power peak. In the left subfigure, tasks are assigned to workstations according to split (0),(1),(2,3),(4,6),(5,7) which is optimal w.r.t. sequence $(0, 1, 2, 3, 4, 6, 5, 7)$: the peak occurs at the overlap of tasks (according to the original indices) 0, 1, 3, 6 and 5, respectively on workstations $k = 0$ to $k = 4$. The new sequence $(0, 1, 2, 4, 6, 3, 5, 7)$ can lead to different possible splits, though not all feasible, e.g. (0),(1),(2),(4,6,3),(5,7) violates takt c on $k = 3$. Of the remaining possible splits, (0),(1),(2,4),(6),(3,5,7) (right subfigure) has its peak at the overlap of 0, 1, 4, 6 and 5 and a peak variation $w_4 - w_3 < 0$; split (0),(1),(2),(4,6),(3,5,7) gives a peak at $t = 0$ (all workstations having a decreasing profile) and a peak change $w_2 + w_4 - (w_5 + w_6) > 0$; split (0),(1),(2,4),(6,3),(5,7) has a peak at the overlap of 0, 1, 4, 3 and 5, the increase is $w_4 - w_6 > 0$. The first case is optimal.

We use neighborhood operator N to implement a Simulated Annealing (SA) algorithm (Aarts et al., 2005). Based on local search, SA consists in a random walk in the solution space, during which deteriorating solutions are accepted with a probability given by a temperature parameter. Such acceptance probability decreases with the temperature, becoming 0 in the last stages of the algorithm. A typical stop criterion is given by a maximum number of iterations. Starting from an SALB3PM-ESD instance, the SA generates a random, precedence-compliant sequence $s_0 = (p(0), ..., p(n-1))$, evaluated by the decoder. At each iteration, a random neighbor $s_{i+1} \in N(s_i)$ of the current sequence s_i is picked and optimally split by the decoder. The temperature is initialized based on an upper bound U of the power peak, then updated every 20 iterations by a factor 0.98. The SA runs for 5000 iterations, so that for each instance the decoder is invoked a number of times large enough to evaluate its behavior. Although sequences are guaranteed to be feasible w.r.t. precedence, a sequence can be met during the random walk that cannot be split without exceeding c. To such a sequence we assign an upper bound on the power peak to penalize it.

Six instances are considered to test the SA: to have a wide range of number n of tasks, 4 are taken from the benchmark set of Lamy et al. (2020), and two are derived from as many SALBP-1 instances of Otto et al. (2013), these latter featuring higher c values. All instances are enhanced with random task power consumption values uniformly generated in $\{5, ..., 50\}$. The MS×ELS algorithm of (Lamy et al., 2020) is also run on them for comparison. Table 1 describes the overall performances of the SA. In it, for each instance, n is reported, along with: the total SA runtime, T_{SA}; the portion of time spent to optimally split the encountered task sequences, T_{d}; the average time per iteration, $\overline{T}_{\mathrm{it}}$; the percentage of infeasible sequences, *inf*; the peak decrease w.r.t. the initial sequence, Δ ($-\infty$ meaning an infeasible initial sequence); the gap w.r.t. the MS×ELS solution. As for the MS×ELS, 20 runs have been launched, each with a time limit $\lceil \frac{T_{SA}}{20} \rceil$, and the best

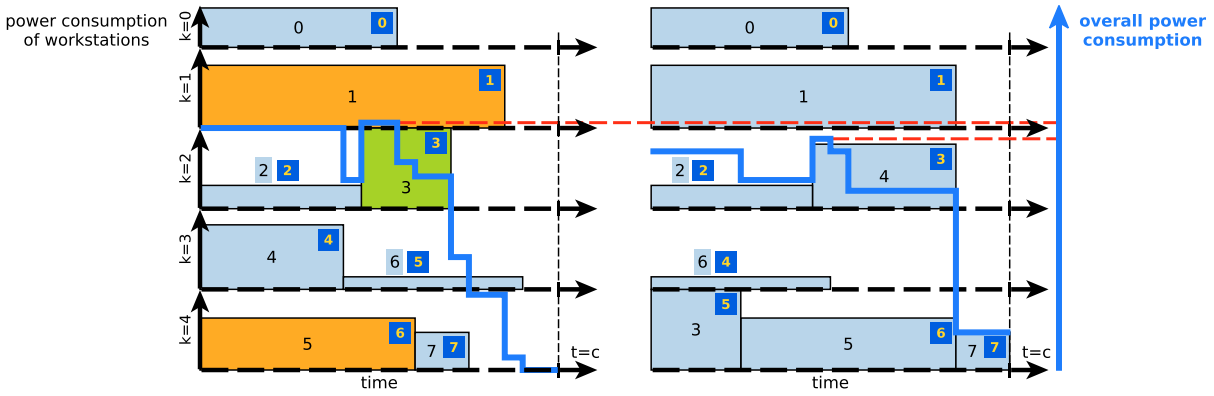

Fig. 2. Two sequences for instance bowman-1, their optimal splits, and a neighbor move from one to the other.

found solution is taken. A ($\star$) symbol in the *opt* column means that the optimal solution is known, and the SA has managed to find it, whereas u means that the optimal solution is not known to date. Table 2 focuses on the average features of the ILPs that the decoder has had to solve, in terms of: percentage of sequences for which the ILP has been solved to optimality at the root node, $solv_{root}$; average optimality gap at the root node, $\overline{gap}_{root}$; average number of variables, $\overline{\#var}$, and constraints, $\overline{\#cns}$.

Table 1. Detailed results of the SA algorithm.

instance	n	T_{SA} (s)	T_{d} (%)	$\overline{T}_{\mathrm{it}}$ (ms)	inf (%)	Δ (%)	MS×ELS (%)	opt
bowman-1	8	9.0	95.3	1.8	0.0	-15.1	0.0	$\star$
jackson-2	11	61.2	99.0	12.2	0.0	-5.6	0.0	$\star$
otto-n20-125	20	643.5	99.1	128.7	1.9	-21.6	3.1	u
heskiaoff-2	28	824.6	96.8	164.9	70.1	$-\infty$	7.9	u
kilbridge-1	45	333.1	59.8	66.7	37.9	-13.0	3.2	u
otto-n50-100	50	4199.5	95.9	839.9	9.1	$-\infty$	7.4	u

Table 2. Features of the decoder ILP models.

instance	n	$\overline{\mathrm{solv}}_{\mathrm{root}}$ (%)	$\overline{\mathrm{gap}}_{\mathrm{root}}$ (%)	$\overline{\#\mathrm{var}}$	$\overline{\#\mathrm{cns}}$
bowman-1	8	100,00	0,00	33,67	61,96
jackson-2	11	78,78	0,92	66,11	146,42
otto-n20-125	20	83,44	0,41	1084,59	2205,41
heskiaoff-2	28	100,00	0,00	566,37	1283,06
kilbridge-1	45	95,16	0,09	238,18	558,70
otto-n50-100	50	77,24	0,52	1351,90	2942,57

Table 1 shows that the SA algorithm has, on average, a runtime of less than 1 second per iteration, and for the two smaller instances is able to find an optimal solution. However, the gap with the MS×ELS in terms of the best found solution with equal time can rise to +8%, which is normal for a simple metaheuristic as the developed SA compared to an algorithm designed to be competitive. T_{SA} predictably grows with n, even though the comparison of kilbridge-1 and heskiaoff-2 reveals that n is not the only factor impacting it. Indeed, a look at Table 2 suggests that T_{SA} presumably depends also on the average size of the ILP problems (which can be roughly estimated by the product of $\overline{\#\mathrm{var}}$ and $\overline{\#\mathrm{cns}}$) which is 5 to 6 times bigger for heskiaoff-2. By comparing now heskiaoff-2 with otto-n20-125 we see that the ILPs of the latter can be estimated being on average 3 to 4 times bigger than the former, yet otto-n20-125 has shorter runtime T_{SA}: this is possibly due to n and the number of infeasible sequences encountered during the random walk for heskiaoff-2, which is the highest among all instances. Such high percentage may be due to heskiaoff-2 being very tight as to the task processing times opposed to the takt time constraint, which gives rise to many infeasible task sequences. These latter are presumably those who take more time for the decoder, which could explain this figures. However, by comparing heskiaoff-2 and otto-n20-125 one can also see that the size of the ILPs of the latter is bigger even though n is smaller: this could be due to a higher number of subsequence variables, hence of split alternatives. Finally, we notice that in general the solving time of the ILP problems represent more than 95% of the overall solving time: the exception of kilbridge-1 could depend on the lower ratio between the ILP solving time and the time spent in neighborhood moves, the first being smaller due to the unusually reduced size of the models, the second depending on n. In general, the decoder in most cases finds the optimum at the root node, or the root gap is very low, which would suggest the effectiveness of model (1)-(9) on the considered instances.

5. CONCLUSIONS AND PERSPECTIVES

In this work, we have studied the SALB3PM-ESD, a problem of balancing an assembly line with consideration of the power peak induced by the production tasks to minimize it. An ILP-based decoder has been proposed, capable of splitting a sequence of the tasks over the sequence of workstations so as to minimize the peak. The decoder has been tested by plugging it in a simple Simulated Annealing (SA) algorithm, which is only meant to be a testbed and not a contribution. Preliminary tests have been run on a small set of SALB3PM-ESD instances from the literature. The results show that the decoder seems to be very effective, due to a tight ILP model, and capable to find optimal or near-optimal solution already at the root node. Some instances can make the decoding harder though, most notably those with larger takt time values. It is also noteworthy to mention that some instances generate a significant number of infeasible sequences.

As to future research perspectives, more extensive computational experiments must be conducted on a wider, more diversified instance set. It would also be interesting to generalize the model to consider production tasks with non-constant and/or machine-dependent power consumption

profiles. The results also suggest promising ideas to yield a more competitive metaheuristic, e.g. stopping the decoder at the root node to reduce the computational time while keeping a good enough approximation of the objective function value, or detecting infeasible sequences before decoding. Based on this, the decoder could be used to take advantage of many existing algorithms of the scheduling literature to develop more performing methods and efficiently tackle the SALB3PM-ESD, but also problems with similar features, such as scheduling problems on parallel machines with no-wait constraints.

REFERENCES

Aarts, E., Korst, J., and Michiels, W. (2005). Simulated annealing. In *Search methodologies*, 187–210. Springer.

Akbar, M. and Irohara, T. (2018). Scheduling for sustainable manufacturing: A review.*J Clean Prod*, 205,866–883.

Ashour, S. (1970). An experimental investigation and comparative evaluation of flow-shop scheduling techniques. *Oper Res*, 18(3), 541–549.

Battaïa, O., Benyoucef, L., Delorme, X., Dolgui, A., and Thevenin, S. (2020). Sustainable and energy efficient reconfigurable manufacturing systems. In *Reconfigurable Manufacturing Systems: From Design to Implementation*, 179–191. Springer.

Baybars, İ. (1986). A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem. *Manage Sci*, 32(8), 909–932.

Borba, L., Ritt, M., and Miralles, C. (2018). Exact and heuristic methods for solving the Robotic Assembly Line Balancing Problem. *Eur J Oper Res*, 270(1), 146–156.

Borisovsky, P.A., Delorme, X., and Dolgui, A. (2013). Genetic algorithm for balancing reconfigurable machining lines. *Comput Ind Eng*, 66(3), 541–547.

Boysen, N., Schulze, P., and Scholl, A. (2022). Assembly line balancing: What happened in the last fifteen years? *Eur J Oper Res*, 301(3), 797–814.

Bruzzone, A.A., Anghinolfi, D., Paolucci, M., and Tonelli, F. (2012). Energy-aware scheduling for improving manufacturing process sustainability: A mathematical model for flexible flow shops. *CIRP Ann-Manuf Techn*, 61(1), 459–462.

Cerqueus, A. and Delorme, X. (2019). A branch-and-bound method for the bi-objective simple line assembly balancing problem. *Int J Prod Res*, 57(18), 5640–5659.

Energy Information Administration (2019). International energy outlook 2019 with projections to 2050. `https://www.eia.gov/outlooks/ieo/pdf/ieo2019.pdf`.

Gianessi, P., Delorme, X., and Masmoudi, O. (2019). Simple Assembly Line Balancing Problem with Power Peak Minimization. In F. Ameri, K.E. Stecke, G. von Cieminski, and D. Kiritsis (eds.), *Advances in Production Management Systems*, volume 566, 239–247. Springer Int. Publishing, Cham.

Giret, A., Trentesaux, D., and Prabhu, V. (2015). Sustainability in manufacturing operations scheduling: A state of the art review. *J Manuf Syst*, 37, 126–140.

International Energy Agency (2021). World energy outlook 2021. `https://www.iea.org/reports/world-energy-outlook-2021`.

International Energy Agency (2022). Electricity Market Report. `https://iea.org/reports/electricity-market-report-july-2022`.

Kemmoe, S., Lamy, D., and Tchernev, N. (2017). Job-shop like manufacturing system with variable power threshold and operations with power requirements. *Int J Prod Res*, 55(20), 6011–6032.

Lahrichi, Y., Damand, D., Deroussi, L., Grangeon, N., and Norre, S. (2022). Investigating two variants of the sequence-dependent robotic assembly line balancing problem by means of a split-based approach. *Int J Prod Res*, 1–17.

Lamy, D., Delorme, X., and Gianessi, P. (2020). Line balancing and sequencing for peak power minimization. *IFAC-PapersOnLine*, 53(2), 10411–10416.

Lange, J. and Werner, F. (2019). A permutation-based heuristic method for the blocking job shop scheduling problem. *IFAC-PapersOnLine*, 52(13), 1403–1408.

Li, X. and Gao, L. (2016). An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *Int J Prod Econ*, 174, 93–110.

Luo, H., Du, B., Huang, G.Q., Chen, H., and Li, X. (2013). Hybrid flow shop scheduling considering machine electricity consumption cost.*Int J Prod Econ*,146(2),423–439.

Masmoudi, O., Delorme, X., and Gianessi, P. (2019). Job-shop scheduling problem with energy consideration. *Int J Prod Econ*, 216, 12–22.

Otto, A., Otto, C., and Scholl, A. (2013). Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing. *Eur J Oper Res*, 228(1), 33–45.

Özcan, U. (2019). Balancing and scheduling tasks in parallel assembly lines with sequence-dependent setup times. *Int J Prod Econ*, 213, 81–96.

Pan, Y., Gao, K., Li, Z., and Wu, N. (2022). Improved meta-heuristics for solving distributed lot-streaming permutation flow shop scheduling problems. *IEEE T Autom Sci Eng*.

Pape, T. (2015). Heuristics and lower bounds for the simple assembly line balancing problem type 1: Overview, computational tests and improvements. *Eur J Oper Res*, 240(1), 32–42.

Ruiz, R. and Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *Eur J Oper Res*, 165(2), 479–494.

Scholl, A. (1995). Data of assembly line balancing problems. Technical report, Darmstadt Technical University, Institute for Business Studies (BWL).

Sewell, E.C. and Jacobson, S.H. (2012). A Branch, Bound, and Remember Algorithm for the Simple Assembly Line Balancing Problem.*INFORMS J Comput*,24(3),433–442.

Wang, K., Li, X., Gao, L., and Li, P. (2020). Energy consumption and profit-oriented disassembly line balancing for waste electrical and electronic equipment. *J Clean Prod*, 265, 121829.

Zhang, R. and Chiong, R. (2016). Solving the energy-efficient job shop scheduling problem: a multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption. *J Clean Prod*, 112(4), 3361–3375.

Zhang, Z., Tang, Q., Li, Z., and Zhang, L. (2019). Modelling and optimisation of energy-efficient u-shaped robotic assembly line balancing problems. *Int J Prod Res*, 57(17), 5520–5537.