Sedding, Helmut A.

**Article — Published Version**

# Mixed-model moving assembly line material placement optimization for a shorter time-dependent worker walking time

Journal of Scheduling

*Suggested Citation:* Sedding, Helmut A. (2023) : Mixed-model moving assembly line material placement optimization for a shorter time-dependent worker walking time, Journal of Scheduling, ISSN 1099-1425, Springer US, New York, Vol. 27, Iss. 3, pp. 257-275, https://doi.org/10.1007/s10951-023-00787-5

This Version is available at:
https://hdl.handle.net/10419/317847

Journal of Scheduling (2024) 27:257–275
https://doi.org/10.1007/s10951-023-00787-5

# Mixed-model moving assembly line material placement optimization for a shorter time-dependent worker walking time

**Helmut A. Sedding**[1,2]

Accepted: 25 May 2023 / Published online: 2 September 2023


**Abstract**
Car mass production commonly involves a moving assembly line that mixes several car models. This requires plenty of material supplies at the line side, but available space is scarce. Thus, material is placed apart from ideal positions. Then, picking it up involves walking along the line. This time is non-productive and can encompass 10–15% of total production time. Thus, it is important to estimate and minimize it during production planning. However, the calculations are difficult because the conveyor continuously moves. Therefore, most literature bounds walking time by a constant, but this discards valuable potential. To better approximate it, we use a time-dependent V-shaped function. A comparison indicates that for a majority of instances, constant walking time estimates of 95% confidence are at least 51% higher. Then, we introduce a model to optimize material positions such that the model-mix walking time is minimized. This poses an NP-hard sequencing problem with a recursive and nonlinear objective function. Our key discovery is a lower bound on the objective of partial solutions, established by a Lagrangian relaxation that can be solved in quadratic time. Resulting branch and bound based algorithms allow to quickly and reliably optimize up to the largest real-world sized instances.

**Keywords** Scheduling · Moving assembly line · Walking time · Material placement · Mixed-model production

## 1 Introduction

Mass-production of cars was particularly made possible by the moving assembly line. It continuously transports the work pieces from worker to worker. Productivity is highest if the assembly operations are divided equally between all workers. This is an NP-hard bin packing type optimization problem (Álvarez-Miranda and Pereira, 2019) that is called assembly line balancing (Salveson, 1955); surveys are found in Battaïa and Dolgui (2013, 2022), Becker and Scholl (2006), Boysen et al. (2022). Solutions can improve with shorter time buffers given more accurate time estimates, e.g., for a worker's workload when assigning operations.

A significant time sink can be a worker's walking time to fetch parts from the line side, amounting for 10–15% of total production time at a major German car manufacturer (Scholl et al., 2013). This time can hardly be estimated with a constant: it is highly variable because the distance is time-dependent. A method to estimate it is introduced in Sedding (2020a) in production of a single model. Walking time is then minimized by placing material supplies at optimal positions along the line side (the operation sequence is fixed). This optimized and more exact walking time estimate allows to better gage the worker's workload during assembly line balancing, potentially increasing overall line utilization. To employ the estimate in interactive planning software, fast compute times are essential, which are provided by heuristics in Sedding (2020a) with a median runtime of 0.002s. In this paper, we adapt this approach to a model-mix production, which is common in car assembly (Sternatz, 2014).

An assembly worker fetches needed material for an assembly operation by walking to the respective material box. In the typical moving car assembly line, boxes are located at the line side, along which the workpiece moves. Ideally, each box is located close to the workpiece's position at access time, as this minimizes walking time. However, if the line side space is scarce (Bautista and Pereira, 2007; Bukchin and Meller, 2005; Boysen et al., 2015), it is usually necessary to place a box apart from its ideal location along the line.

✉ Helmut A. Sedding
helmut.sedding@zhaw.ch

[1] Institute of Theoretical Computer Science, Ulm University, Ulm, Germany

[2] Institute of Data Analysis and Process Design, ZHAW, Winterthur, Switzerland

The moving workpiece's distance to a box and the resulting two-way walking time can be modeled by a convex function of time (Sedding, 2020a). However, most models for assembly line planning in the literature are restricted to constant processing time estimates, just like in the classic assembly line balancing problem (Salveson, 1955). Although sequence-dependent nonproductive processing times are included in Andrés et al. (2008), Scholl et al. (2013), Esmaeilbeigi et al. (2016), they cannot be applied to precisely plan walking times at assembly lines with a moving workpiece: Overly large safety factors are needed to upper bound the walking time with a constant. This gives away potential to increase the utilization of an assembly line. In this study, we test the potential of time-dependent walking time estimates.

When assigning operations to a worker, a realistic walking time estimate needs to take possible local optimizations into account. A minimization of a worker's walking time can be approached in two major ways for a given set of assembly operations. On the one hand, it is possible to optimize the operation sequence (Jaehn and Sedding, 2016; Sedding, 2020b). On the other, the positioning of the respective material boxes can be adjusted (Klampfl et al., 2006; Finnsgård et al., 2011; Sedding, 2017, 2020a). This requires to fix the operation sequence to determine a walking time optimized box placement. Finnsgård et al. (2011) describe a manual optimization and report a walking distance reduction by 52%. Schmid et al. (2021) optimize the placement with a mixed-integer programming approach that considers variable walking costs during material placement, but ignores the workpiece's continued movement while the worker walks. Klampfl et al. (2006) take this movement into account, using Euclidean distances to calculate time-dependent walking times. In three approaches, they consider placement of boxes along the line. First with overlaps, then without overlaps, and finally with stacking atop each other. Nonlinear programming is used to heuristically find placements. However, they record rather long compute times already for a small instance of five material boxes. A one-dimensional walking time model yields a significantly quicker optimization in Sedding (2017, 2020a, 2020c) for the case of single model placement.

In this paper, we adapt the material placement optimization in Sedding (2020a) to the mixed-model moving assembly line. A mixed-model assembly shares the same line for several product models (Thomopoulos, 1967). Then, the production can better adapt to varying demands of each model. This production method is standard in car assembly (Sternatz, 2014). Line side space can be even more scarce if further material is required for each model (Boysen et al., 2015). In the mixed-model setting, the objective is to minimize total processing time weighted by model share. This allows for longer processing times on some product models, especially rarer ones, provided this can be compensated for on other product models. Allowance is provided by the worker floating up- or downstream the line. However, an overload over the course of several cycles cannot be compensated anymore as increasing walking times exacerbate the overload. Such situations must be prevented during planning, in particular of the production sequence, cf. Boysen et al. (2009c) for a survey.

Our paper's contribution lies in placing material boxes at the line side for a worker's assembly operations over a mix of product models such that the model-mix weighted time-dependent walking times to gather material are minimized:

- First, we display all assumptions and introduce an optimization model in Sect. 2. In comparison to Sedding (2020a, 2020c), it permits multiple models and an offset for the placement area (or indirectly the start time).
- We observe that the problem entails a recursive, nonlinear objective function, which impedes evaluation of partial solutions and renders incremental solution procedures difficult. We provide a proof of strong NP-hardness for any number of product models (Sect. 3).
- A mixed-integer program is derived from the single-model case in Sedding (2020a, 2020c).
- The main technical contribution of our paper is a Lagrangian relaxation of the mathematical program, for which we find a partition into two independent subproblems, each of which is solved in quadratic time. This results in a fast lower bound for partial solutions (Sect. 5). This lower bound is the cornerstone of a branch and bound algorithm that incrementally places the boxes. A truncated branch and bound search provides a fast heuristic (Sect. 6).
- Finally, a numerical experiment shows the effectiveness of the approaches (Sect. 7). The total runtime of the best heuristic is negligibly small: for the largest instances, the median runtime is 0.071 s. Such a runtime is suitable for a use in interactive planning software. Potential savings are high in comparison to constant walking time estimates as well as intuitive placements (Sect. 7.5).
- Our model is the first in the literature to efficiently consider and minimize a time-dependent model of walking times for material placement at the mixed-model moving assembly line. As this production mode is standard for car production, our approach has a far-reaching applicability.

Preliminary versions of this paper are available in this special issue's workshop proceedings (Sedding, 2021) and in the author's dissertation (Sedding, 2020c, Chapter 6).

## 2 Modeling

This section presents the studied optimization problem $\mathscr{P}m$ for placing boxes for a model-mix of $m$ products. After a

formal definition of all parameters in a $\mathscr{P}m$ instance, its prerequisites are introduced. The definition of $\mathscr{P}m$ follows.

Note that $\mathscr{P}m$ builds upon the optimization problem in Sedding (2020a, 2020c) for a single product assembly $m = 1$. While the single-model case is extended to a mixed-model production, all other assumptions remain unchanged. A list of assumptions made for $\mathscr{P}m$ is gathered in Table 1.

**Definition 1** An instance of $\mathscr{P}m$ is given by

- $a \in \mathbb{Q} \cap [0, 1]$ as walking time slope (box ahead),
- $b \in \mathbb{Q} \cap [0, \infty)$ as walking time slope (box behind),
- $I$ as set of product models,
- $m = |I|$ as number of product models,
- $r_i \in \mathbb{Q} \cap [0, 1]$ s.t. $\sum_{i \in I} r_i = 1$, as production rate of model $i \in I$
- $J = \{(i, j) \mid i \in I, j \in \{1, \ldots, n_i\}\}$ as set of all jobs,
- $n_i \in \mathbb{N}$ as number of jobs of model $i \in I$, which remain in the fixed sequence $(i, 1), (i, 2), \ldots, (i, n_i)$,
- $n = |J| = \sum_{i \in I} n_i$ as total number of jobs,
- $\ell_{i,j} \in \mathbb{Q} \cap [0, \infty)$ as assembly time of job $(i, j) \in J$,
- $w_{i,j} \in \mathbb{N}$ as width of the box of job $(i, j) \in J$,
- $V \in \mathbb{Q}$ as start of the placement interval,
- $W = V + \sum_{(i,j) \in J} w_{i,j}$ as the placement interval end.

Before the optimization problem $\mathscr{P}m$ can be defined, we specify how boxes can be placed, see how walking time for fetching part of a boxes can be modeled, and define how to calculate the makespan (completion time) of a production cycle encompassing walking and assembly time. The result is visualized for an example instance in Fig. 1.

## 2.1 Box placement

Necessary material of an assembly operation is provided in a dedicated *box* $(i, j) \in J$. It takes up a certain *box width* $w_{i,j}$ along the line side, expressed as a share of the available width (in the dimension along the line). We may assume $w_{i,j} \in \mathbb{N}$ by scaling the coordinate system accordingly.

Let set $J$ denote the set of boxes for the the worker. The boxes are placed side-by-side without overlaps. Given a scarce line side space, we assume there are no gaps between boxes. Then, the boxes are placed in a contiguous section at the line side: the space between $V$ and $W = V + \sum_{(i,j) \in J} w_{i,j}$.

Then, a *box placement*

$$\{\pi_{i,j}\}_{(i,j) \in J} \tag{1}$$

**Table 1** The model assumptions equal the study of the single-model case in Sedding (2020a, 2020c) except for A3 switching to a model-mix and adding A16

| | |
|---|---|
| *A1* | *Single station* Focus on one assembly station and work piece |
| *A2* | *Single worker* Focus on one worker. Interference is ruled out |
| *A3* | *Multiple product variants* We consider a model-mix of several product variants, each with a different production rate and its own set of assembly operations and containers. While a production sequence of the product variants is not yet determined, it is assumed that the sequence fulfills the production rates on average |
| *A4* | *Fixed operation sequence* A worker has an immutable sequence of operations albeit it may be changed in practice |
| *A5* | *Single cycle* A production cycle begins when the workpiece enters the worker's zone, and it ends when all operations have been finished on this workpiece. If that takes shorter or longer than the average cycle time, the worker can start the next cycle early or late. This may cause a different starting point, highly depending on the production sequence. However, the sequence is not available during material placement. Thus, we assume an average cycle time and disregard floating |
| *A6* | *Single work point* The work point is at one location at the workpiece. We assume that other work points are assigned to different workers on assembly line balancing (Becker and Scholl, 2009) |
| *A7* | *One box per operation* One single box aggregates all material containers of an operation, e.g., as a stack or a shelf |
| *A8* | *One-dimensional box placement* Boxes are placed along a line parallel to the assembly line. Hence, the box width along this dimension is the decisive size to measure a box's occupied interval at the line side |
| *A9* | *No space between boxes* There are no gaps between adjacent boxes in our model. This assumption is realistic as space at the line side is typically scarce (Sternatz, 2015). Note that more elaborate logistics operations like material sequencing or kitting may reduce required space at the line side (Boysen et al., 2015; Schmid and Limère, 2019) |
| *A10* | *Stationary box positions* Box positions are fixed during production and optimized beforehand |
| *A11* | *Uniform walking velocity* Walking velocity is constant and the same for all operations |
| *A12* | *One-dimensional walking* Only walking in parallel to the assembly line is counted: Boxes are located in close tangency the conveyor to reduce the orthogonal margin to a gripping distance |
| *A13* | *One walk per operation* The worker fetches necessary parts all and only before the start of an assembly operation from a single box as in A7, An operation can aggregate several sub-operations, for all of which material is gathered in one single run |
| *A14* | *No pick time* Pick times are neglected. Finnsgård et al. (2011) reports that pick times encompass just 6% of nonproductive time |
| *A15* | *Picking at upstream side* We simplistically assume picking at the upstream (left) edge, albeit several pick points may exist for a box |
| *A16* | *Placement area offset* The box placement interval start can be given with a shift off the station start along the line |

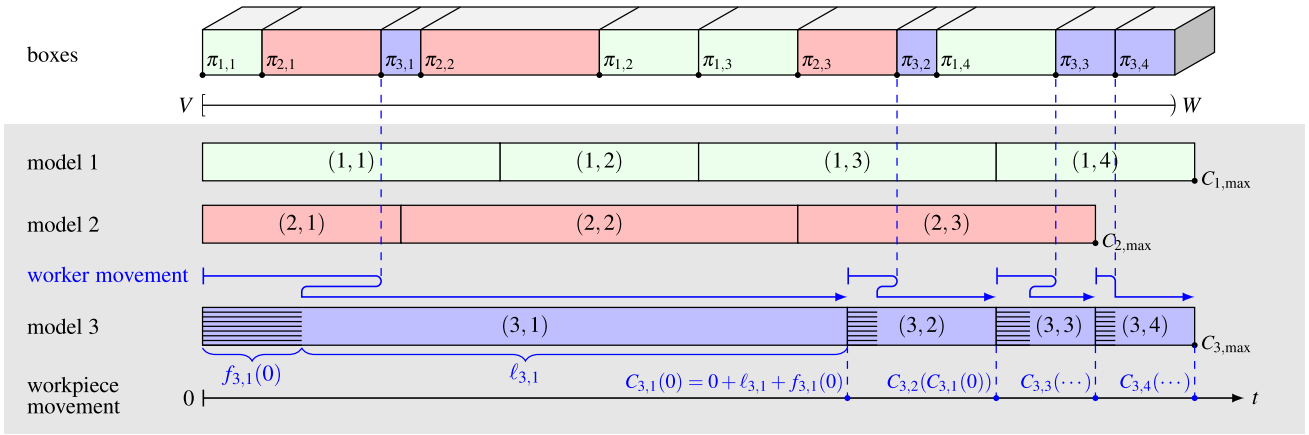


**Fig. 1** On planning the line side placement, boxes are positioned in a sequence at the line side in $[V, W)$, displayed in the figure's top row. Shown below are assembly operations (jobs) in fixed sequences for three product models. The horizontal axis represents the time; it shows assembly operations performed by the worker while a workpiece moves along the line over time $t$. Note that the time scale equals the space scale. At the start of a production cycle, the worker is at time and spatial position 0. Blue arrow lines indicate, exemplary for model 3, the worker's longitudinal movement along the line during one production cycle, alternating between fetching material (walking the curved line) and assembling, during both of which the workpiece moves. Job (3, 1) starts at time 0; with walking time $f_{3,1}(0)$ (indicated by the striped area, to&from the box at $\pi_{3,1}$) and assembly time $\ell_{3,1}$, the job completes at $C_{3,1}(0)$. Job completion times, calculated recursively, are labeled in blue color. A production cycle of model 3 is completed at time (and position) $C_{3,\max}$. The objective is to minimize the average completion time over all models by placing the boxes such that walking times are minimal.

states, for each box $(i, j) \in J$, a rational-valued position $V \le \pi_{i,j} \le W - w_{i,j}$ to place it on interval $[\pi_{i,j}, \pi_{i,j} + w_{i,j})$ at the line side such that $\bigcup_{(i,j)\in J} [\pi_{i,j}, \pi_{i,j} + w_{i,j}) = [V, W)$. Note that attaining a box placement is equivalent to finding a sequence for the boxes along the line side.

## 2.2 Walking time

The walking time calculation in Sedding (2020a) is briefly described in the following. In this model, the worker only walks along the moving assembly line; movement orthogonal to the line can be ignored. Three walking strategies are covered: Always walking on the fixed floor, atop the conveyor's moving floor, or whichever is better in the current movement direction.

For walking up to a box $(i, j) \in J$, the worker leaves the workpiece at a certain time $t$, arrives at the box's position $\pi_{i,j}$, and then returns to the workpiece. All the while, the workpiece continues to move. This gives several movement equations, which are solved with a closed formula. Then, the walking time is represented by the piecewise-linear function

$$f_{i,j}(t) = \max\{-a \cdot (t - \pi_{i,j}),\ b \cdot (t - \pi_{i,j})\} \tag{2}$$

where $\pi_{i,j}$ is the box position encoded in time units, and $0 \le a \le 1$ and $b \ge 0$ are two slopes that depend on the worker's velocity and walking strategy. See also Fig. 2.

If the current time equals the box position (case $t = \pi_{i,j}$), then the walking time is minimum, which occurs if the workpiece just passes by the box position. Otherwise, the walking time increases linearly. If the workpiece has not yet passed the box position (case $t < \pi_{i,j}$), then the walking time corresponds to $-a \cdot (t - \pi_{i,j})$, otherwise it is $b \cdot (t - \pi_{i,j})$.

The two slopes relate the workpiece's velocity $v_{\text{conveyor}}$ to the worker's walking velocity $v_{\text{worker}} > v_{\text{conveyor}}$. For example, if the worker walks on a non-moving floor beside the workpiece, then $a = 2/(v + 1)$ and $b = 2/(v - 1)$ where $v = v_{\text{worker}}/v_{\text{conveyor}}$. The same slope values are attained if the worker walks on floor plates that move together with the workpiece. If a worker can always choose the best of both options, then the slopes are $a = (2v + 1)/(1 + v)^2$ and $b = (2v + 1)/v^2$. Note that this walking strategy yields, if $v = 13.6$ as in Klampfl et al. (2006), a walking time reduction by 3.5% if $t < \pi_{i,j}$, else 4.1%, see Sedding (2020a).

## 2.3 Makespan calculation

Before an assembly operation can be processed, a walking time occurs to its distinct box $(i, j) \in J$. Together, these two constitute a *job*, which is also denoted by $(i, j)$. Then, job $(i, j)$ consists of two parts: the nonproductive walking time function $f_{i,j}$ in (2), and after that, a productive assembly time $\ell_{i,j} \ge 0$ that is a constant nonnegative rational number. Together, they form the job's processing time $p_{i,j}(t) = f_{i,j}(t) + \ell_{i,j}$. Starting at time $t$, the job completes at $C_{i,j}(t) = t + p_{i,j}(t)$. Substituting all components,

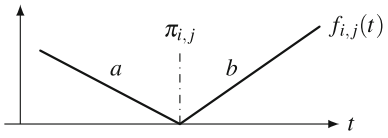


**Fig. 2** Visualization of the walking time function $f_{i,j}$ of job $(i, j) \in J$ in (2) as a function of start time $t$

the job's completion time is expressed by

$$C_{i,j}(t) = t + \ell_{i,j} + \max\{-a \cdot (t - \pi_{i,j}),\ b \cdot (t - \pi_{i,j})\}. \tag{3}$$

Each product model $i \in I$ requires processing of a certain number $n_i$ of jobs in a fixed sequence denoted by

$$(i, 1), (i, 2), \ldots, (i, n_i)$$

where $n_i$ is the number of jobs in model $i$. Then, the last completion time $C_i^{\max}$ in a model $i$ is the composition of job completion times (3), starting the first job $(i, 1)$ at time 0:

$$C_i^{\max} = C_{i,n_i}(\cdots C_{i,2}(C_{i,1}(0)))\cdots). \tag{4}$$

Note that a start of the first job at $t_{\min} \neq 0$ is attained by shifting the box placement interval by $-t_{\min}$.

***Remark 1*** The job sequence is fixed here. As mentioned in the introduction, it is also possible to optimize walking time by permuting the job sequence (Sedding, 2020b, 2020c). This belongs to the field of time-dependent scheduling (Gawiejnowicz, 2020b, 2020a). Related piecewise-linear convex $C_{i,j}$ functions are described in Farahani and Hosseini (2013), Kawase et al. (2018), Kononov (1998), while a recent survey is found in Sedding (2020b).

### 2.4 Optimization problem

Minimizing the overall walking time in the mixed-model setting is equal to minimizing the total weighted makespan (completion time) over all product models (Klampfl et al., 2006). This can be attained by a sum of each model $i$'s last completion time $C_i^{\max}$ weighted by production share $r_i$. This objective is minimized in the studied optimization problem.

**Definition 2** (Problem $\mathscr{P}m$) Given a $\mathscr{P}m$ instance (see Definition 1), minimize the weighted average makespan

$$\phi = \sum_{i \in I} r_i C_i^{\max}$$

by determining a box placement $\{\pi_{i,j}\}_{(i,j)\in J}$, which places the boxes, each represented by box width $w_{i,j}$, in a sequence in $[V, W)$, see (1). This yields, for model $i \in I$, makespan

$$C_i^{\max} = (C_{i,n_i} \circ \cdots \circ C_{i,2} \circ C_{i,1})(0),$$

which composes the completion times of model $i$'s job sequence according to (4). By (3), the completion time $C_{i,j}$ of job $(i, j) \in J$ as a function of job start time $t$ is given by

$$C_{i,j}(t) = t + \ell_{i,j} + \max\{-a \cdot (t - \pi_{i,j}),\ b \cdot (t - \pi_{i,j})\}.$$

In the single-model case, the makespan cannot be larger than the cycle time. Every cycle, a new workpiece arrives, hence a larger makespan would require a line stoppage (or a work overload situation). Herein lies a key advantage of model-mix production: If the weighted average makespan $\phi$ is not above the cycle time, then there is no work overload (on average). Models with a higher makespan let the worker float downstream, others upstream.

Hence, some models may be allowed a makespan longer than the cycle time. However, note that such an overload situation would affect the next production cycle's start time. Then, walking times are affected. If it occurs repeatedly, however, walking times can grow quickly as the worker is driven away from the boxes. If the worker has floated downstream behind the corresponding boxes, then any delay increases the completion time by a factor of $(1 + b)^n$ for $n$ remaining jobs, which follows from Sedding (2020b, Corollary 2). In excess, the worker needs assistance and/or the line needs to stop. Such overload situations must prevented in planning of the production sequence.

Although the makespan should equal the cycle time, it can well be different to the placement interval's end $W$. Moreover, the model allows a nonzero placement interval start $V$. In both cases, the placement area is incongruent to the assembly station, covering only a part, or extending out of it. This models that the placement area is offset to the assembly station. This is required, e.g., when subdividing the line-side space into different sized regions, which can benefit the overall assembly line balance. A nonzero start time of the first job can depict floating of the worker up- or downstream the line. This is represented in our model by shifting the placement interval back by the same amount.

## 3 Computational complexity

The main difficulty of optimizing a box placement is caused by the recursive and nonlinear nature of the objective. For example, if the first job's box is moved, then its walking time changes. As a consequence, succeeding jobs start at a different point in time. This time might be earlier or later

than before. As each walking time function is nonlinear, the objective value changes nonlinearly. Moreover, the respective ideal box location of succeeding jobs changes. Then, the placement of these boxes needs further reoptimization.

To highlight the complexity of $\mathscr{P}m$, we prove that it is NP-hard in the strong sense for an arbitrary number of product models $m \geq 1$. For this, we perform a reduction from 3-Partition as defined in Garey and Johnson (1978), which is NP-complete in the strong sense (Garey and Johnson, 1975).

**Definition 3** (3-Partition) Given a bound $B \in \mathbb{N}$ and $3z$ elements in multiset $X = \{x_1, \ldots, x_{3z}\} \subset \mathbb{N}$ with $B/4 < x < B/2$ for $x \in X$, and $\sum_{x \in X} x = zB$. The question is: does there exist a partition of $X$ into $z$ disjoint multisets $A_i$, $i = 1, \ldots, z$ with $\sum_{x \in A_i} x = B$?

**Theorem 1** *$\mathscr{P}m$ is* NP*-hard in the strong sense for arbitrary $m \geq 1$.*

***Proof*** For $m = 2$, we perform a reduction from 3-Partition as of Definition 3. This requires a decision version of $\mathscr{P}m$ that specifies a threshold value $\Phi$ and asks if a solution exists with objective value $\phi \leq \Phi$.

A corresponding instance is constructed as follows. First, we freely choose any allowed nonzero slope $0 < a \leq 1, b > 0$ value. For the two models $I = \{1, 2\}$, we set production share $r_1 = 0$ and $r_2 = 1$. Hence, model 1 incurs no walking time in the objective function. However, it occupies space at the line side for its $n_1 = 3z$ jobs for which we let $w_{1,j} = x_j$, $j = 1, \ldots, 3z$, and choose an arbitrary $\ell_{1,j}$ value. The second model has $n_2 = z+1$ jobs. For each $j = 1, \ldots, z+1$, we set $w_{2,j} = 1$ and $\ell_{2,j} = B + 1$. Finally, we set threshold value

$$\Phi = \sum_{j=1,\ldots,z+1} l_{2,j} = (B+1)\,(z+1)\,.$$

This instance's objective value is, given the unilateral production shares, $\phi = C_{2,z+1}$. As $C_{2,z+1} \leq \Phi$, there is $\phi \leq \Phi \iff \phi = \Phi$. Let us assume that $\phi = \Phi$. This requires in the second product model for each $j = 1, \ldots, z+1$ that $p_{2,j} = \ell_{2,j}$. This is the case if and only if the corresponding box is precisely positioned at $(B+1) \cdot j$. These boxes leave gaps of width $B$. Each gap is closed by the first product model's boxes if and only if the 3-Partition instance can be solved. Hence, $\mathscr{P}m$ is NP-hard for $m = 2$.

We generalize this reduction to $m > 2$ by extending the instance with $m - 2$ models $I'$. For each $i \in I'$, let $r_i = 0$. Then, the last completion times of models $I'$ yield no impact on the objective function. Moreover, we introduce an arbitrary number of jobs for each added model $i \in I'$, each with the same box width $B + 1$ and an arbitrary assembly time. Because these boxes are too wide to be placed between two adjacent boxes of the second product model for $\phi \leq \Phi$, they must be placed after the last one. Hence, they assert no effect on the box placement of the first and the second product model.

For $m = 1$, strong NP-hardness is shown in Sedding (2020a). Concluding, a pseudopolynomial reduction of 3-Partition to $\mathscr{P}m$ exists for $m \geq 1$. □

## 4 Mathematical programming

We describe $\mathscr{P}m$ solutions using mathematical programming, adapting the special case $m = 1$ in Sedding (2020a, 2020c) to $m \geq 1$. This includes a makespan calculation for each product model and changes the objective function to a sum of makespans weighted by production share. Then, we derive a mixed-integer program, reformulating box placement constraints.

### 4.1 Mathematical program

We introduce continuous variables $\pi_{i,j}$ as box position and $C_{i,j}$ as completion time of job $(i, j) \in J$. Then, a mathematical program for $\mathscr{P}m$ can be stated as:

$$\text{minimize} \quad \sum_{i \in I} r_i \, C_{i,n_i}$$

subject to

$$C_{i,0} = 0, \qquad i \in I, \tag{5a}$$

$$C_{i,j} \geq C_{i,j-1} + \ell_{i,j} - a\big(C_{i,j-1} - \pi_{i,k}\big),\ (i,j) \in J, \tag{5b}$$

$$C_{i,j} \geq C_{i,j-1} + \ell_{i,j} + b\big(C_{i,j-1} - \pi_{i,k}\big),\ (i,j) \in J, \tag{5c}$$

$$\{\pi_{i,j}\}_{(i,j) \in J} \text{ being a box placement.} \tag{5d}$$

Completion times are set recursively in constraints (5b) and (5c) starting with (5a). Constraint (5d) restricts the box position variables to a valid box placement as defined in (1).

We observe that attaining a valid box placement corresponds to a job sequencing problem on a single machine, which determines each job's processing interval from start time to completion time. This is alike to the interval a box is placed upon; the difference being that the job's interval is typically denoted by its end (the completion time), while we instead describe a box position by the interval start.

On a side note, it is known from job sequencing that idle times add a further layer of complexity, e.g., when optimizing non-regular objectives like earliness and tardiness penalties (Garey et al., 1988). Similarly, we presume that the assumption of placing boxes without gaps provides, besides the practical reason, a computational benefit.

### 4.2 Mixed-integer program

Model (5) is restricted to a mixed-integer program by substituting the placement constraints (5d). There exists a variety of possible formulations in the related domain of job sequencing, cf. Queyranne and Schulz (1994), Keha et al. (2009), Baker and Keller (2010) for reviews. We use disjunctive sequencing constraints to ensure consistency and comparability with the mixed-integer program and the numerical study in Sedding (2020c) for the single-model case $m = 1$. For job sequencing, this method is treated, e.g., in Manne (1960), Balas (1985), Queyranne (1993).

Disjunctive sequencing constraints yield a total order on the boxes to decide the position of each. This is established by disjunctive constraints between each pair of jobs.

Let us ease the notation by introducing $(i, j) \prec (h, k)$, a relation between jobs $(i, j), (h, k) \in J$ that holds if and only if $i < h$, and in case of $i = h$, if $j < k$. Moreover, we abbreviate a job's pair notation by a single letter, i.e., by writing $x = (i, j)$ or $y = (h, k)$ in this section.

Then, (5d) is substituted by

$$\pi_x + w_x \le \pi_y \quad \vee \quad \pi_y + w_y \le \pi_x, x, y \in J, \; x \prec y, \tag{6}$$

$$V \le \pi_x \le W - w_x, \qquad x \in J. \tag{7}$$

This can be reformulated as a mixed-integer program using the 'big-M' method. This relaxes either of the inequalities in (6) by adding $W$, because $\pi_x + w_x \le W$ for all $x \in J$. Let $u_{x,y}$ denote a binary variable for each job pair $x, y \in J$ with $x \prec y$. Then, while (7) remains, (6) is replaced with

$$\pi_x + w_x \le \pi_y + W\big(1 - u_{x,y}\big), \qquad x, y \in J, \; x \prec y, \tag{8a}$$

$$\pi_y + w_y \le \pi_x + W\, u_{x,y}, \qquad x, y \in J, \; x \prec y, \tag{8b}$$

$$u_{x,y} \in \{0, 1\}, \qquad x, y \in J, \; x \prec y. \tag{8c}$$

The resulting mixed-integer program (MIP) encompasses constraints (5a)–(5c), (7), (8a)–(8c), with $n\,(n-1)/2$ binary variables.

## 5 Lower bound

A lower bound on the minimum objective value $\phi^*$ of a $\mathscr{P}m$ instance is introduced in the following. We consider a Lagrangian relaxation of the mathematical program (5) and show that it is possible to solve it with a quadratic time algorithm.

The lower bound also accepts a partially solved instance for use within a branch and bound search. A solution can be constructed by placing the boxes in the placement interval $[V, W)$. It starts at $V$ with the first box, and places the next box besides. This is repeated until all boxes are placed. An intermediate, partial solution can be subsumed as follows.

**Definition 4** A partial solution provides the box position $\pi_{i,j}, (i, j) \in J_\mathrm{F}$, for a set of *fixed jobs* $J_\mathrm{F} \subseteq J$ such that these boxes are placed in $[V, F)$ where $F = V + \sum_{(i,j)\in J_\mathrm{F}} w_{i,j}$, i.e., there is $V \le \pi_{i,j} \le F - w_{i,j}$ for $(i, j) \in J_\mathrm{F}$. Then, we call $\{\pi_{i,j}\}_{(i,j)\in J_\mathrm{F}}$ a *partial box placement*. It is completed by placing the boxes of the remaining *open jobs* $J_\mathrm{O} = J \setminus J_\mathrm{F}$ between $F$ and $W$.

### 5.1 Recurrence solving

In (5), we solve the recurrence relation of the completion time variables to a closed form. Let us introduce, for each job $(i, j) \in J$, a continuous variable walking time $\omega_{i,j}$ and deviation $\delta_{i,j}$ of the job's start time from its ideal start time (i.e., $\pi_{i,j}$). Each completion time variable is replaced by a sum of all assembly and walking times until then. This replaces constraints (5a) to (5c) with

$$C_{i,j} = \sum_{k=1,\dots,j} \ell_{i,k} + \omega_{i,k}, \qquad (i, j) \in J, \tag{9a}$$

$$\omega_{i,j} \ge -a\delta_{i,j}, \qquad (i, j) \in J, \tag{9b}$$

$$\omega_{i,j} \ge b\delta_{i,j}, \qquad (i, j) \in J, \tag{9c}$$

$$\delta_{i,j} = -\pi_{i,j} + \sum_{k=1,\dots,j-1} \ell_{i,k} + \omega_{i,k}, \qquad (i, j) \in J. \tag{9d}$$

Walking time is piecewise linear and, because it is minimized, limited from below in constraints (9b) and (9c). It depends on the deviation of a job's start time to its position, set in constraints (9d).

The walking time variables' domain can be limited to strengthen the model: the domain is not less than zero, and at most, it corresponds to the walking time in forwards direction to at most the box position $W-1$, and in the backwards direction, to the lowest possible position $V$ (and back). Hence,

$$0 \le \omega_{i,j} \le \max\big\{-a\,\big(C_{i,j-1} - (W-1)\big),\; b\,\big(C_{i,j-1} - V\big)\big\}$$

for $(i, j) \in J$. Substituting the completion times variables with the sum in (9a) gives, for $(i, j) \in J$, the closed formula

$$0 \le \omega_{i,j} \le \max\left\{-a\left(1 - W + \sum_{k=1,\dots,j-1} \ell_{i,k} + \omega_{i,k}\right),\; b\left(-V + \sum_{k=1,\dots,j-1} \ell_{i,k} + \omega_{i,k}\right)\right\}. \tag{9e}$$

### 5.2 Lagrangian relaxation

With the model modifications, we perform a Lagrangian relaxation of constraints (9b) and (9c). This introduces the corresponding Lagrangian multipliers $\lambda_{i,j} \geq 0$, $\lambda'_{i,j} \geq 0$ for $(i,j) \in J$.

Then, the Lagrangian problem is

$$\phi^*_{\text{Lagr}}(L) = \min \phi_{\text{Lagr}}$$

with

$$\phi_{\text{Lagr}} = \sum_{i \in I} r_i\, C_{i,n_i} + \sum_{(i,j)\in J} \lambda_{i,j}\left(-a\delta_{i,j} - \omega_{i,j}\right) + \lambda'_{i,j}\left(b\delta_{i,j} - \omega_{i,j}\right) \tag{10}$$

subject to

$$L = \left(\left(\lambda_{i,j}, \lambda'_{i,j}\right)\right)_{(i,j)\in J} \geq 0,$$

as well as constraints (9a) to (9e), and constraint (5d).

The set of multipliers $L$ can be optimized using a standard subgradient optimization, see Fisher (2004). Note that the lower bound inequality $\phi^*_{\text{Lagr}}(L) \leq \phi^*$ holds for any $L$.

Substituting the completion time variables in (10) according to (9a) yields

$$\begin{aligned}\phi_{\text{Lagr}} &= \sum_{(i,j)\in J} r_i(\ell_{i,j}+\omega_{i,j}) + (b\lambda'_{i,j} - a\lambda_{i,j})\delta_{i,j} - (\lambda_{i,j}+\lambda'_{i,j})\omega_{i,j} \\ &= \sum_{(i,j)\in J} \ell_{i,j}\zeta_{i,j} + \underbrace{\sum_{(i,j)\in J} \omega_{i,j}\theta_{i,j}}_{\Omega} + \underbrace{\sum_{(i,j)\in J} \left(a\lambda_{i,j} - b\lambda'_{i,j}\right)\pi_{i,j}}_{\Pi}\end{aligned}$$

with constants

$$\zeta_{i,j} = r_i + \sum_{k=j+1,\ldots,n_i} \left(b\lambda'_{i,k} - a\lambda_{i,k}\right), \quad (i,j) \in J,$$

$$\text{and} \quad \theta_{i,j} = \zeta_{i,j} - \left(\lambda_{i,j} + \lambda'_{i,j}\right), \quad (i,j) \in J.$$

Observe that the walking time and box placement variables occur only in different constraints.

***Property 1*** In the Lagrangian problem, the walking time variables $\omega_{i,j}$, $(i,j) \in J$, and box position variables $\pi_{i,j}$, $(i,j) \in J_{\text{O}}$ (from Definition 4), are independent.

Thus, it is possible to separately optimize walking times and box positions. This gives us the partial objective

$$\Omega = \sum_{(i,j)\in J} \omega_{i,j}\theta_{i,j}$$

for the walking times, and

$$\Pi = \sum_{(i,j)\in J} \left(a\lambda_{i,j} - b\lambda'_{i,j}\right)\pi_{i,j}$$

for the box positions.

### 5.3 Optimizing box position values

The boxes of the open jobs $J_{\text{O}}$ (cf. Definition 4) are to be placed within a box sequence between $F$ and $W$. In partial objective $\Pi$, each box $(i,j) \in J_{\text{O}}$ adds term $\left(a\lambda_{i,j} - b\lambda'_{i,j}\right)\pi_{i,j}$. Hence to minimize $\Pi$, we get a classic total weighted completion time scheduling problem of the boxes (as jobs), which is solved in polynomial time by sorting them (Smith, 1956).

**Lemma 1** *Partial objective $\Pi$ is minimum if $\pi_{i,j}, (i,j) \in J_{\text{O}}$, are obtained by sequencing $J_{\text{O}}$'s boxes nonincreasingly by*

$$\frac{a\lambda_{i,j} - b\lambda'_{i,j}}{w_{i,j}}.$$

Thus for $n_{\text{O}} = |J_{\text{O}}|$, optimal box position values are attained in $\mathcal{O}(n_{\text{O}} \log n_{\text{O}})$ time.

### 5.4 Optimizing walking time values

The walking time variables $\omega_{i,j}$, $(i,j) \in J$, are optimized by minimizing $\Omega$.

For each $(i,j) \in J$, the value range of $\omega_{i,j}$ is limited by constraints (9e). It can be transformed with constants $q_{i,j} = -V + \sum_{k=1,\ldots,j-1} \ell_{i,k}$ and $q'_{i,j} = 1 - W + q_{i,j} + V$ to the range

$$0 \leq \omega_{i,j} \leq \max\left\{\underbrace{-a\left(q'_{i,j} + \sum_{k=1,\ldots,j-1} \omega_{i,k}\right)}_{\alpha_{i,j}}, \underbrace{b\left(q_{i,j} + \sum_{k=1,\ldots,j-1} \omega_{i,k}\right)}_{\beta_{i,j}}\right\}. \tag{11}$$

We observe for any $i \in I$, and with increasing $j$ from 1 to $n_i$ that term $\alpha_{i,j}$ (as defined in (11)) is nonincreasing and term $\beta_{i,j}$ (as in (11)) is nondecreasing. Hence, we can find some $\kappa_i \in \{0,\ldots,n_i\}$ such that $\alpha_{i,j} > \beta_{i,j}$ for all $j = \kappa_i + 1,\ldots,n_i$. Given such a $\kappa_i$, we can replace constraints (11) by

$$\begin{aligned} 0 \leq \omega_{i,j} &\leq \alpha_{i,j} \quad \text{if } j \leq \kappa_i, \\ 0 \leq \omega_{i,j} &\leq \beta_{i,j} \quad \text{if } j > \kappa_i. \end{aligned}$$

Hence, depending on the value of $\kappa_i$, either of the two range constraints is active for job $(i, j) \in J$.

We replace the upper bound by an equality with slack variable $0 \le y_{i,j} \le 1$. Then,

$$\begin{aligned} 0 \le \omega_{i,j} &= y_{i,j}\alpha_{i,j} \quad \text{if } j \le \kappa_i, \\ 0 \le \omega_{i,j} &= y_{i,j}\beta_{i,j} \quad \text{if } j > \kappa_i. \end{aligned} \tag{12}$$

***Property 2*** Given $\kappa_i$, $i \in I$, of an optimal solution. If $\alpha_{i,j} < 0$ for any job $(i, j) \in J$ with $j \le \kappa_i$, then it is possible to decrease $\kappa_i$ without changing the objective $\Omega$ such that $\alpha_{i,j} \ge 0$ for each job $(i, j) \in J$ with $j \le \kappa_i$.

***Proof*** Given the described case, then $\kappa_i \ge 1$ and $\alpha_{i,\kappa_i} < 0$ because $\alpha_{i,j}$ is decreasing with $j$. Hence, $y_{i,\kappa_i} = \omega_{i,\kappa_i} = 0$ to fulfill constraints (12).

Let us decrease $\kappa_i$ by one. Then, it is possible to leave $y_{i,\kappa_i+1} = \omega_{i,\kappa_i+1} = 0$ in the solution. Thus, $\Omega$ remains unchanged. We repeat this step until $\alpha_{i,\kappa_i} \ge 0$. □

**Corollary 1** *An optimum solution exists where $\alpha_{i,j} \ge 0$ holds for each $(i, j) \in J$ with $j \le \kappa_i$.*

**Lemma 2** *For each $i \in I$, let $\kappa_i^*$ be the minimum value for $\kappa_i$ that permits an optimum solution. Then, there exists such a solution where for each job $(i, j) \in J$ there is*

$$y_{i,j} = \begin{cases} 0, & \text{if } \theta_{i,j} + \displaystyle\sum_{k=j+1,\dots,n_i} y_{i,k}c_{i,k}\theta_{i,k} > 0, \\ 1, & \text{else} \end{cases}$$

$$\text{with } c_{i,k} = \begin{cases} -a, & \text{if } k \le \kappa_i^*, \\ b, & \text{else.} \end{cases} \tag{13}$$

***Proof*** For each model $i \in I$, we show this by induction for $j = n_i, \dots, 1$. Then,

$$\omega_{i,j} = y_{i,j} \cdot \underbrace{c_{i,j}\left(d_{i,j} + \sum_{k=1,\dots,j-1} \omega_{i,k}\right)}_{\ge 0}$$

$$\text{with } c_{i,j} = \begin{cases} -a, & \text{if } j \le \kappa_i^*, \\ b, & \text{if } j > \kappa_i^*, \end{cases} \quad d_{i,j} = \begin{cases} q'_{i,j}, & \text{if } j \le \kappa_i^*, \\ q_{i,j}, & \text{if } j > \kappa_i^*. \end{cases}$$

By choice of $\kappa_i$ and use of Property 2, the slack variable $y_{i,j}$ multiplies a nonnegative value. Hence, $\omega_{i,j}$ is nonnegative. Moreover, $\omega_{i,j}$ influences $\omega_{i,k}$ for each $k = j + 1, \dots, n_i$ unless $y_{i,k} = 0$. Thus, $\omega_{i,j}$ contributes to the objective $\Omega$ not only with factor $\theta_{i,j}$, but moreover via $\omega_{i,k}$ with factor $y_{i,k}c_{i,k}\theta_{i,k}$. The total contribution of $\omega_{i,j}$ to $\Omega$ is thus with factor $\theta_{i,j} + \sum_{k=j+1,\dots,n_i} y_{i,k}c_{i,k}\theta_{i,k}$. If this factor is positive, then the lowest slack value $y_{i,j} = 0$ minimizes $\Omega$. If it is negative, then the highest slack value $y_{i,j} = 1$ is optimal. If the factor is zero, any value for $y_{i,j}$ is optimal. □

Let $\overline{\kappa_i} \in \{0, \dots, n_i\}$ for each $i \in I$ be the maximum $\kappa_i$ for which $-aq'_{i,\overline{\kappa_i}} \ge 0$ holds.

***Property 3*** An optimum solution exists where $\kappa_i \le \overline{\kappa_i}$ for each $i \in I$.

***Proof*** Assume we are given an optimum solution. Naturally, all walking time variables $\omega_{i,j}$, $(i, j) \in J$, are nonnegative. For each $i \in I$, both $\sum_{k=1,\dots,j-1} \omega_{i,k}$ and $q'_{i,j}$ are nondecreasing with respect to $j$, while $-aq'_{i,j}$ is nonincreasing. Hence if $-aq'_{i,\kappa'_i} < 0$ for any $(i, \kappa'_i) \in J$ with $\kappa'_i \le \kappa_i$, then $\alpha_{i,j} < 0$ for $j = \kappa'_i, \dots, \kappa_i$. However it is, according to Property 2, possible to set $\kappa_i < \kappa'_i$ such that the solution is optimum and $-aq'_{i,j} < 0$ as well as $\alpha_{i,j} \ge 0$ hold for any $(i, j) \in J$ with $j \le \kappa_i$. □

The presented results allow us to describe an algorithm to minimize the walking time variables.

**Lemma 3** *In an outer loop, set $\kappa_i = 0, \dots, \overline{\kappa_i}$ for $i \in I$. Given $\kappa_i$, Lemma 2's recurrence (13) is used to set $y_{i,j}$ for each $(i, j) \in J$, which also sets $\omega_{i,j}$ and objective value $\Omega$. Then, the smallest obtained objective value is optimal.*

This algorithm takes quadratic time in terms of $n_i$. Over all $m$ models, the worst case runtime is $\mathcal{O}\left(\sum_{i \in I} n_i^2\right) \le \mathcal{O}\left(n^2\right)$.

Combining the algorithm in Lemma 3 with the box sequencing procedure in Lemma 1, we are able to find a solution for the whole Lagrangian problem.

**Theorem 2** *An optimum solution to $\phi^*_{Lagr}(L)$ is computed in $\mathcal{O}\left(n^2\right)$ time.*

## 6 Branch and bound methods

For solving $\mathcal{P}m$ instances, we describe a branch and bound search (B&B) and a heuristic version in the following. The upper bound is initialized with basic heuristics. Bounding is performed with the above Lagrangian based lower bound and an additional combinatorial lower bound. The heuristic version introduces a heuristic dominance rule and limits the number of visited descending nodes.

### 6.1 Basic heuristics

To construct a good upper bound for the branch and bound search we use a constructive heuristic, a local search, and a simulated annealing metaheuristic.

*Weighted nearest identity (WNID)* As a construction heuristic, an intuitive way to place the boxes is in the same sequence as the jobs, which is already reported in Ford and Crowther (1922, p. 80). For a single model, this strategy is described in Sedding (2020a); it can be called identity sequence placement

heuristic. Extending this to multiple models, our strategy is to intersperse the identity sequences of all models $I$ such that

$$\pi_{i,j} < \pi_{i,j'} \text{ for all } (i,j),(i,j') \in J \text{ with } j < j'. \tag{14}$$

Our approach is greedy, it places all boxes iteratively along the line, starting at position 0. In an iteration step, we select the next box to place. To fulfill (14), there are at most $m$ possible boxes to choose from. We rank the boxes by the resulting walking time weighted by the reciprocal of the corresponding model's production share. Accordingly, we call this method *weighted nearest identity* (WNID) sequence placement heuristic, see Algorithm 6.1.

**Algorithm 1** WNID: Weighted nearest identity sequence placement heuristic

$(j_i)_{i\in I} \leftarrow 1$ ▷ set the next box of each model
$(t_i)_{i\in I} \leftarrow 0$ ▷ set the current time in each model
$F \leftarrow V$ ▷ set the next position to place a box
**loop** $|J|$ **times**
  $N \leftarrow \{(i,j_i) \in J \mid i \in I\}$ ▷ get possible next boxes
  $(i,j) \leftarrow \operatorname{argmin}_{(i,j)\in N} \frac{1}{r_i}\max\{-a\,(t_i - F),\ b\,(t_i - F)\}$
  $\pi_{i,j} \leftarrow F$ ▷ place the selected box $(i,j)$ at $F$
  $t_i \leftarrow C_{i,j}(t_i)$ ▷ advance model $i$'s time
  $F \leftarrow F + w_{i,j}$ ▷ increase $F$ to the next free space
  $j_i \leftarrow j_i + 1$ ▷ select the next box in model $i$
**return** $\{\pi_{i,j}\}_{(i,j)\in J}$

*Hill Climbing (HC)* A local search typically improves a constructive heuristic's solution. For this, we use the transpose neighborhood that comprises all possible swaps of neighboring boxes. Hence, the number of neighbors is $|J| - 1$ for $|J|$ boxes. The local search procedure is initialized with the WNID solution. Of the current solution's neighborhood, this procedure searches for the neighbor with highest improvement of the objective value. If such a neighbor exists, it is selected as the current solution. This is known as *hill climbing* (HC) search.

*Simulated Annealing (SA)* is a metaheuristic that escapes local minima (Kirkpatrick et al., 1983; Černý, 1985). In comparison, the local search procedure stops at some local minimum. It might occasionally correspond to a global minimum, but it is usually sensible to apply SA: Given certain conditions, SA converges to a global optimum (Hajek, 1988). To cross the solution space, SA permits worse solutions with a decreasing probability. In our case, we use the basic and widespread procedure of Press et al. (1992).

### 6.2 Branch and bound algorithm

An exact solution for $\mathscr{P}m$ can be computed by a branch and bound search (B&B). We use the above described heuristics' objective value as an initial upper bound.

*Branching* Our B&B performs a depth first search, discarding nodes that do not lead to an optimal solution. Any node corresponds to a partial box placement $\{\pi_j\}_{j\in J_F}$ of some fixed jobs $J_F \subseteq J$, as of Definition 4. The root node initializes fixed job set $J_F = \emptyset$ and open job set $J_O = J$. A leaf node is reached if $J_F = J$ and $J_O = \emptyset$. A descending node is created by fixing an open job $j \in J_O = J \setminus J_F$, placing it at $\pi_j = V + \sum_{k\in J_F} w_k$. A node is discarded if its lower bound is not less than the upper bound. The latter corresponds to the currently best known box placement's objective value, which is initialized with the described HC or SA heuristics and updated whenever a better solution is reached at a leaf node.

*Combinatorial lower bound* When our B&B search visits a new node, it is first evaluated by a quickly obtained lower bound. This allows to potentially discard it before computing the more elaborate Lagrangian lower bound. Let us first consider a trivial lower bound: It places each box exactly at its ideal time: the corresponding job's start time. In a partial solution, this bound can be increased by using the fixed $\pi_{i,j}$ values of all fixed jobs $(i,j) \in J_F$. Although the position of the open boxes is not yet known, we know that they will be placed in the unoccupied interval $[F, W)$. Hence, if the start time $t_{i,j}$ of an open job $(i,j) \in J_O$ lies within $[F, W - w_{i,j}]$, then we still place the box exactly at this start time. Otherwise, the box can be placed at $F$ or at $W - w_{i,j}$, whichever is closer to $t_{i,j}$. This increases the lower bound further. Concluding, we calculate the start time for each job in a model iteratively. If we encounter an open job $(i,j) \in J_O$, then its box position is temporarily set to

$$\pi_{i,j} = \max\{F,\ \min\{t,\ W - w_{i,j}\}\}.$$

*Lagrangian lower bound* Secondly, our B&B evaluates a partial solution with the Lagrangian based lower bound of Sect. 5. While the lower bound is calculated in quadratic time for a given set of Lagrangian multipliers $L$, the multipliers are iteratively improved using a subgradient optimization based on Fisher (2004), Held et al. (1974) as follows.

– In our case, the iteration step's updated set of multipliers $\hat{L}$ for step size $s > 0$ is

$$\hat{\lambda}_{i,j} = \max\left\{\lambda_{i,j} + s\left(-a\delta_{i,j} - \omega_{i,j}\right),\ 0\right\},$$
$$\hat{\lambda}'_{i,j} = \max\left\{\lambda'_{i,j} + s\left(b\delta_{i,j} - \omega_{i,j}\right),\ 0\right\},$$

for each $i, j \in J$. For the step size, we employ the common

$$s = \frac{v \cdot \left(UB - \phi^*_{\text{Lagr}}(L)\right)}{\sum_{(i,j)\in J} \left(-a\delta_{i,j} - \omega_{i,j}\right)^2 + \left(b\delta_{i,j} - \omega_{i,j}\right)^2}$$

where $UB \geq \phi^*$ is an upper bound value for $\phi^*$, and $v \in (0, 2]$ is a step size factor. Our initial value for the latter is $v = 1$. Then, we divide $v$ by two after 10 iterations of no improvement on the Lagrangian objective $\phi^*_{\text{Lagr}}$.

– We reduce the number of iterations by reusing multiplier values during the B&B search like in Fisher (2004). This avoids optimizing $L$ from scratch for each partial solution. Exactly one set of $L$ values is memorized. Hence, the last obtained multiplier values $L$ provide a warm-start in the next bound calculation.
– Our B&B performs a higher number of iterations earlier in the search tree. A better bound has a greater utility there. We iterate at most

$$\max\left\{1, \left\lfloor 4 \cdot \sqrt{|J_{\text{O}}|} \right\rfloor\right\}$$

times; except at the root node (with $|J_{\text{O}}| = n$), where we allow for up to $10n$ iterations, to initialize $L$ from scratch.

*Traversing order* The Lagrangian lower bound calculation is additionally used in our B&B search to set a traversing order. This determines, in each node, the sequence for visiting the descending nodes. More promising descending nodes should be visited first. Each descending node places a different box at $F$. Lemma 1's artificial position value provides a hint for good posititions of the open boxes $J_{\text{O}}$. This motivates our use of $\pi_{i,j}$, $(i, j) \in J_{\text{O}}$, as a traversing order value. We use the nondecreasing order of $\pi_{i,j}$ for ranking the open boxes and visit the descending nodes accordingly.

### 6.3 Truncated branch and bound algorithm

Our *truncated branch and bound* (TrB&B) leaves out some nodes in the B&B search, which yields a heuristic. Moreover, a node is evaluated with a heuristic dominance rule that compares a currently visited node to previously visited nodes.

Adapting the approach in Sedding (2020a), we limit the number of descending nodes to the maximum branching factor

$$\text{BF}_{\max} = \min\left\{|J_{\text{O}}|, \max\left\{\lceil\psi\rceil, \left\lfloor \frac{|J_{\text{O}}|}{\sigma} \right\rfloor\right\}\right\}$$

for constant $\psi > 0$ and $\sigma > 0$. While the number of descending nodes is restricted by $\sigma$ near the root of the search tree, it is restricted by $\psi$ when being deeper in the search tree.

With our heuristic dominance rule, the current node is discarded if the partial solution is (probably) dominated by any previous partial solution. For this, we adapt the approach in Sedding (2020a). Then, the heuristic dominance rule works as follows. Based on a partial box placement, it creates two complete artificial placements:

(a) All open jobs $j \in J_{\text{O}}$ are placed such that $\pi_j = F$,
(b) all open jobs $j \in J_{\text{O}}$ are placed such that $\pi_j = W - w_j$.

This yields two heuristic objective values $\phi_{(\text{a})}$ and $\phi_{(\text{b})}$. Then, a partial solution is heuristically dominated (and can be discarded) unless at least one of the two values is less than a previously found value, respectively, for the same set $J_{\text{F}}$ of fixed jobs.

## 7 Numerical results

A quantitative evaluation of the described solution methods is performed in the following numerical study. We create artificial instances and perform a full factorial evaluation with the exact approaches and, secondly, with the heuristics.

### 7.1 Instance generation

We generate random instances in a variety of parameter settings. Comparability to Sedding (2020c) is ensured by using the same variants for assembly times, box widths, and slopes.

(1) The number of product models is set to $m \in \{2, 4, 8\}$.
(2) The production shares of all models $I$ need to be positive and sum up to one. We generate these shares randomly analogous to Boysen et al. (2008, 2009a, 2009b) as follows. Let $PC$ denote the total production cycle count. Initialize each model's demand to one unit. Then, select a model with uniform probability, increase the model's demand by one unit. Repeat this step until the total demand equals $PC$. This method is equivalent to a uniform sample (with replacement) of $(PC - m)$ values from $I$. Then, the frequency of each model plus one corresponds to the model's demand. Finally, dividing the demand of each model by $PC$ gives its production share. We let $PC = 1000m$ to avoid quantization. Note that in the limit for $PC \to \infty$, the expected value for each share is $1/m$.
(3) The number of jobs is set to $n \in \{8, 12, \ldots, 28\}$. A job's model is selected with uniform probability. Hence, each model's expected number of jobs is $n/m$.

**Table 2** Exact algorithms' runtime in seconds, grouped by $n$ and $m$

| $n$ | $m$ | MIP | | | B&B | | |
|---|---|---|---|---|---|---|---|
| | | Md | Q75 | solved (%) | Md | Q75 | solved (%) |
| ∗ | ∗ | 35.60 | ≥ 3600 | 71 | 0.59 | 159.02 | 87 |
| 8 | ∗ | 0.02 | 0.09 | 100 | 0.00 | 0.00 | 100 |
| 12 | ∗ | 0.33 | 1.84 | 100 | 0.00 | 0.01 | 100 |
| 16 | ∗ | 10.59 | 105.84 | 97 | 0.12 | 0.29 | 100 |
| 20 | ∗ | 470.75 | ≥ 3600 | 64 | 5.08 | 13.96 | 100 |
| 24 | ∗ | ≥ 3600 | ≥ 3600 | 38 | 195.22 | 560.50 | 93 |
| 28 | ∗ | ≥ 3600 | ≥ 3600 | 26 | ≥ 3600 | ≥ 3600 | 28 |
| ∗ | 2 | 11.27 | 3194.58 | 75 | 0.33 | 86.90 | 88 |
| ∗ | 4 | 30.40 | ≥ 3600 | 70 | 0.56 | 136.04 | 87 |
| ∗ | 8 | 102.84 | ≥ 3600 | 66 | 1.40 | 299.77 | 85 |
| 8 | 2 | 0.02 | 0.02 | 100 | 0.00 | 0.00 | 100 |
| 8 | 4 | 0.02 | 0.03 | 100 | 0.00 | 0.00 | 100 |
| 8 | 8 | 0.25 | 0.51 | 100 | 0.00 | 0.00 | 100 |
| 12 | 2 | 0.14 | 0.39 | 100 | 0.00 | 0.00 | 100 |
| 12 | 4 | 0.26 | 0.83 | 100 | 0.00 | 0.00 | 100 |
| 12 | 8 | 3.16 | 17.73 | 100 | 0.01 | 0.01 | 100 |
| 16 | 2 | 3.73 | 16.58 | 100 | 0.08 | 0.21 | 100 |
| 16 | 4 | 7.42 | 58.39 | 99 | 0.10 | 0.22 | 100 |
| 16 | 8 | 105.42 | 441.26 | 91 | 0.21 | 0.48 | 100 |
| 20 | 2 | 129.71 | 1761.67 | 81 | 3.16 | 11.05 | 100 |
| 20 | 4 | 490.70 | ≥ 3600 | 63 | 4.19 | 9.68 | 100 |
| 20 | 8 | ≥ 3600 | ≥ 3600 | 47 | 8.68 | 22.56 | 100 |
| 24 | 2 | ≥ 3600 | ≥ 3600 | 44 | 136.06 | 580.62 | 92 |
| 24 | 4 | ≥ 3600 | ≥ 3600 | 37 | 152.39 | 413.69 | 96 |
| 24 | 8 | ≥ 3600 | ≥ 3600 | 33 | 296.49 | 1033.29 | 92 |
| 28 | 2 | ≥ 3600 | ≥ 3600 | 28 | ≥ 3600 | ≥ 3600 | 37 |
| 28 | 4 | ≥ 3600 | ≥ 3600 | 22 | ≥ 3600 | ≥ 3600 | 28 |
| 28 | 8 | ≥ 3600 | ≥ 3600 | 28 | ≥ 3600 | ≥ 3600 | 18 |

Md, median runtime in seconds; Q75, upper quartile in seconds; solved, percentage of instances solved in 1 h
∗ aggregate of all variants of the respective parameter

(4) The jobs' assembly times are generated in four variants (Jaehn and Sedding, 2016; Sedding, 2020a):

(L1) all equal (unitary value 1),
(L2) all distinct (a random permutation of $(1, 2, \ldots, n)$),
(L3) uniform random variates from $\{1, 2, \ldots, 10\}$,
(L4) geometric random variates with mean $\lambda = 2$.

(5) The box widths of the jobs are drawn in either of four variants, and normalized by scaling and rounding to obtain a uniform total width $10 \cdot n$ (Sedding, 2020a):

(W1) all equal box widths (unitary value 1),
(W2) all distinct (a random permutation of $(1, 2, \ldots, n)$),
(W3) uniform random variates from $\{2^0, \ldots, 2^3\} \cup \{3 \cdot 2^0, \ldots, 3 \cdot 2^2\}$, which represent the ISO1-pallet divisions of Euro stacking boxes,
(W4) rounded up gamma variates with shape 1.25 and unit scale.

(6) The slopes $a, b$ are determined by the walking velocity $v$, which is expressed in multiples of the conveyor velocity, and the walking strategy. Like in Sedding (2020a), we vary the walking velocity $v \in \{2, 4, 8, 16\}$ while the conveyor velocity is kept unitary. Then, the slope values follow from the chosen strategy:

(S1) $a = 2/(v + 1)$ and $b = 2/(v - 1)$ (walking either besides or atop the moving conveyor),
(S2) $a = (2v + 1)/(v + 1)^2$ and $b = (2v + 1)/v^2$ (walking both besides and atop, always choosing the faster option).

(7) Finally, processing times are harmonized with the box widths with respect to the walking velocity. In the test, we consider the case of a zero box placement interval start $V = 0$ and that its end $W$ is approximately equal or larger than the mixed-model weighted average makespan,

i.e., the objective value $\phi$. Otherwise, the problem is likely easier, as many start times will be after the box position (if $V \ll 0$ or $W \ll \phi^*$, cf. Sedding (2020b)) or before (if $V \gg 0$ or $W \gg \phi^*$). Note it is not necessary to compute the optimum $\phi^*$ during harmonization because strict equality of $W$ and $\phi^*$ is not required. Thus, we can use the rather intuitive solution of the WNID heuristic. Then, the deviation $|\phi - W|$ is minimized by linearly scaling all assembly times uniformly with the same factor, using the univariate optimization method in Brent (1971). This procedure is repeated at most 10 times because the WNID's solution can change due to the scaling.

The parameter setting variants yield a product of $3 \cdot 6 \cdot 4 \cdot 4 \cdot 4 \cdot 2 = 2304$ settings. For each setting, ten instances are generated, which yields 23040 instances total.

### 7.2 Test setup

Performance comparablity to the single-model study in Sedding (2020c) is ensured by, besides using the same mixed-integer programming approach, using equal hardware and software.

The algorithms are all implemented in the C++11 programming language. C++ STL containers are used for the data structures. Computations remain single-threaded. The code is compiled by GCC to x86-64 binaries. Mixed-integer programs are given to the Gurobi 7.5 C++ API and solved with it. For a fair comparison, the Gurobi computation is limited to one thread. For Simulated Annealing, the reference implementation of Press et al. (1992, pp. 448–451) with default parameters is used. The TrB&B heuristic is paramerized with $\psi = 5, \sigma = 7$.

All programs are ran on Ubuntu Linux. Each instance gets a dedicated CPU core of an Intel Xeon E5-2680.

A computation is terminated after a 1 h time limit. In this case, although the final runtime is unknown, it is lower bounded to at least 1 h. Thus, runtime percentiles (and median) can still be calculated if the respective share of instances finished within the time limit.

### 7.3 Exact algorithms

Tested exact approaches are the mixed-integer program (MIP, Sect. 4.2) and the branch and bound (B&B, Sect. 6.2). Resulting runtimes are aggregated in Table 2, in groupings by $n$ and $m$.

Let us first analyze the MIP and the B&B together:

- For both, we can observe an exponential growth of runtime with increasing $n$, and some increase along with $m$. This happens irrespective of the number of models $m$. With the strong NP-hardness proof for $\mathscr{P}m$ in Theorem 1, this behavior is not unexpected. Both approaches exhibit high quartile deviations. The decrease in the quartile deviation in groups of higher $n$ might be explained by the smaller number of solved instances in such a group. The correlation between B&B and MIP runtimes is 0.25, which is weakly positive. Hence both approaches experience some similar difficulty with the instances.
- The B&B is consistently faster than the MIP. The B&B is faster in 18718 (81.24%) of all 20200 solved instances. While B&B is able to solve a clear majority of instances until $n = 24$, the MIP can solve the majority only until $n = 16$. On an aggregate level, the median speed advantage of the B&B is 66-fold.

**Table 3** Median B&B runtime in seconds for $n = 24$ and $m = 4$ grouped in walking velocity and walking strategy pairs

| | $v = 2$ | $v = 4$ | $v = 8$ | $v = 16$ |
|---|---|---|---|---|
| S1 | 210.55 | 271.23 | 131.73 | 67.08 |
| S2 | 216.01 | 262.37 | 108.52 | 87.39 |

**Table 4** Median B&B runtime in seconds for $n = 24$ and $m = 4$ grouped in pairs of assembly time and box width settings

| | L1 | L2 | L3 | L4 |
|---|---|---|---|---|
| W1 | 106.52 | 118.78 | 103.84 | 76.35 |
| W2 | 102.37 | 180.76 | 136.40 | 154.03 |
| W3 | 148.55 | 124.23 | 295.41 | 220.50 |
| W4 | 202.12 | 201.55 | 265.69 | 237.31 |

Further analysis of B&B performance shows peak difficulty for walking velocity $v = 4$ and for box width setting W4. It has lowest difficulty for W1, but remains inconclusive for the different assembly time settings:

- Table 3 lists median runtimes grouped by each ($v$, S) pair. Both walking strategies yield similar runtimes. The runtime is smallest for $v = 16$, rises highest at $v = 4$, and falls again for $v = 2$. Note that this picture is different from the single-model case in Sedding (2020a): their B&B for $m = 1$ had the highest runtime for $v = 16$, and the lowest for $v = 2$.
- Table 4 groups each (L, W) pair. Compared to the other box width settings W4 is highest with L1, L2, and L4; W3 is highest with L3 (with W4 close up). Hence, W4 is likely the most difficult case.
- Of the assembly time settings in Table 4, L2 runtime is highest for W1 and W2; while L3 is highest with W3 and W4. L4 is lowest for W1, L2 for W2, L3 for W3 and W4. With this diverse ranking it remains unclear which case has the highest or lowest performance.

Concluding, the B&B is clearly the best performing exact approach for solving the $\mathscr{P}m$ with a median 66-fold speed advantage over the MIP. The B&B exhibits relatively uniform performance in all box width variants, is slower with the more realistic assembly times W3 and W4, and is faster for instances of the more realistic velocities $v = 8$ and $v = 16$.

### 7.4 Proportion of walking time to total work time

The exact solutions allow us to analyze the share of walking time compared to the total work time in our instances. Scholl et al. (2013) reports that material fetching amounts to about 10–15% of total work time at an assembly line of a large German automobile manufacturer. Our most realistic worker velocity cases are $v = 8$ and $v = 16$ (Klampfl et al. (2006) document a similar assumption of $v = 13.6$). Table 5 shows those cases attain 5.7–11% median walking time for walking strategy S1. It is slightly lower 5.2–11% in strategy S2. Note this is less than the 9–15% mean walking time in Sedding (2020a) for a single product model $m = 1$.

### 7.5 Comparison with constant walking time estimates

In most of the literature, it is assumed that the time to gather material at the line side of a moving conveyor can be estimated by constant estimates. This includes methods time measurements (MTM) and most scientific literature. A compromise might be to represent walking time as costs (Limère et al., 2015; Schmid et al., 2021; Müllerklein et al., 2022). However, this can be imprecise if the space at the line side is scarce and high walking times occur.

In Table 5, a high variability of the minimum walking time can be observed. Thus, using the mean walking time as a constant estimate poses the risk of misguided planning. However, a conservative estimate can be far off. For example for $v = 8$ in S1, the 95% percentile is 16.8% walking time. Taking this as a conservative estimate yields an overestimation of walking time in a quarter of the instances (Q25) by at least 75%, and in half of the instances (the median is 11.1%) by at least 51%. Similarly for $v = 16$ in S1: here, the 95% percentile is 9.6%; hence the overestimation is over 108% compared to the lower quartile and 68% compared to the median. Therefore, we conclude that constant estimates of walking time are either misguiding or overly conservative because of the observed variability in the instances' minimum walking times.

An accurate depiction of walking times would henceforth improve planning accuracy, for example, in assembly line balancing for an even distribution of work to workers (Boysen et al., 2022), or in product sequencing, which determines the order of product models at the mixed-model line (Becker and Scholl, 2006; Boysen et al., 2009c, 2012). Both problems would ideally be considered simultaneously (Boysen et al., 2022). An accurate, time-dependent walking time estimate aids to reduce overload situations and minimize idle time. As elucidated in Sect. 2.4, the worker may start a cycle at a varying position due to floating, depending on the previous cycle's completion time. Hence, walking times are highly dynamic. To minimize them, it may be useful to continuously reorder the worker's operations where possible, e.g., with methods described in Jaehn and Sedding (2016), Sedding (2020b).

### 7.6 Heuristics

Tested basic heuristic approaches (cf. Sect. 6.1) are: the weighted nearest identity sequence (WNID) placement heuristic that provides the start for a best improvement local search (HC) and a simulated annealing (SA) metaheuristic. Also, we tested the truncated branch and bound (TrB&B, Sect. 6.3), initialized with the WNID upper bound (denoted TrB&B$_{UB\,\mathrm{HC}}$) and also the SA upper bound (TrB&B$_{UB\,\mathrm{SA}}$). For a comparison, we let the MIP terminate on time limits of 10, 60, and 3600 s.

We measure solution quality by the percentage increase of walking time, which divides additional walking time by minimum walking time. This is captured by the percentage walking time error

$$\mathrm{PE}(\phi, \phi') = \frac{\phi - \phi'}{\phi' - \sum_{(i,j)\in J} r_i \ell_{i,j}} \cdot 100\% \tag{15}$$

for two objective values.

We perform a subgroup analysis of the heuristics on the instance subset that is exactly solved by B&B within the 1 h time limit. Let MPE denote mean $\mathrm{PE}(\phi, \phi^*)$ of the attained objective $\phi$ and an instance's optimum $\phi^*$. Please refer to Table 6 to observe the share of optimally solved instances and MPE values grouped by $n$ and $m$ values, and $(n, m)$ pairs. The number of finished instances increases over time, which is plotted for several approaches in Fig. 3 for $n = 24, m = 4$. The resulting PE is shown as box plots in Fig. 4.

The WNID yields a weak performance, still it is able to achieve an optimum solution for some small instances. The HC much improves on this, attains an optimum very frequently in 39% of the cases. Its computation time is barely measurable. The SA improves on this by optimally solving 68% of the instances. For $n = 24$ and $m = 4$, its median runtime is measurable with 0.026 s.

The TrB&B greatly improves on both the HC and the SA's upper bound. In the former case, it finds the optimum solution for 46% of the instances. It reduces the MPE from 121% to 5.05%. For $n = 24$ and $m = 4$, the median runtime rises to 0.146 s.

**Table 5** Median (50% percentile), quartile (25%, 75% percentile), and 95% percentile minimum walking time percentage of total weighted work time for $n = 24$ and $m = 4$ instances with a known optimum, in walking velocity and walking strategy pairs

| $v$ | S1 | | | | S2 | | | |
|---|---|---|---|---|---|---|---|---|
| | Q25 (%) | Q50 (%) | Q75 (%) | Q95 (%) | Q25 (%) | Q50 (%) | Q75 (%) | Q95 (%) |
| 2 | 46 | 51 | 57 | 67 | 33 | 38 | 43 | 50 |
| 4 | 22 | 24 | 27 | 33 | 18 | 21 | 23 | 27 |
| 8 | 9.7 | 11 | 13 | 17 | 8.3 | 11 | 13 | 17 |
| 16 | 4.6 | 5.8 | 7.1 | 9.7 | 4.3 | 5.4 | 6.5 | 9.5 |

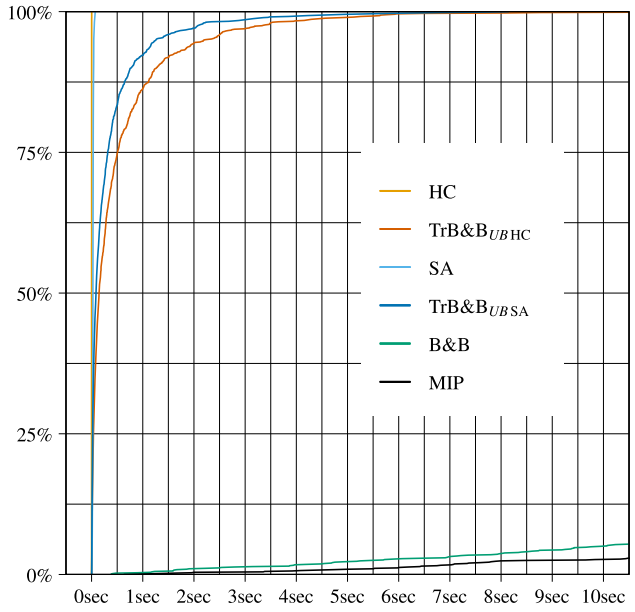


**Fig. 3** Percentage of finished instances with $n = 24$ and $m = 4$ in a line plot of algorithm runtime in seconds

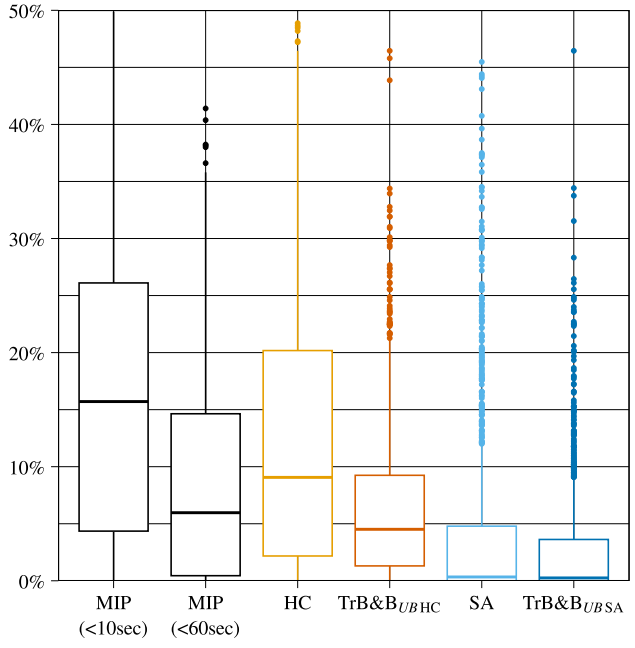


**Fig. 4** Box plots of percentage walking time error PE($\phi, \phi^*$) of a heuristic's $\phi$ for $n = 24$ and $m = 4$ instances solved by B&B with optimum $\phi^*$

With the SA upper bound, an optimum is found for 72% of the instances, further reducing the MPE to 1.77%. For $n = 24$ and $m = 4$, the median runtime is 0.140 s, which includes the SA runtime.

Except for small instances, the time-limited MIP at 10 or 60 s has worse PE and MPE values even though the runtime is much higher than of the other heuristics: The MIP's median runtime corresponds to the time limit because available compute time is mostly used up completely. With a long runtime, only a small PE remains.

An assessment of the full set of instances would require knowledge of an optimal solution of all instances. For 2463 instances, an optimum could not be attained within the 1 h time limit with neither exact algorithm (MIP, B&B). To study the set of all instances in a consistent way, we use the objective value $\phi^{\mathrm{H}}$ of the long running MIP (<1 h) as a reference, because it returns the smallest overall MPE in the preceding subgroup analysis. The heuristics' objective value $\phi$ can then be assessed with the percentage of instances that yield $\phi \leq \phi^{\mathrm{H}}$, and with the mean of the percentage walking time error PE($\phi, \phi^{\mathrm{H}}$) denoted by HMPE. We observe similar results for this test as in the subgroup analysis. In particular, executing the MIP for 60 s gives yields relatively high HMPE for large instances. Compared to the MIP runtime of 1 h, the HMPE of the TrB&B is noticeably less for high $n$ and $m$ values, while maintaining a much shorter runtime. See also Table 7.

Concluding, the TrB&B$_{UB\,SA}$ provides the best heuristic solutions, makes good use of SA's initial upper bound, and retains a low runtime. This suggests that the research and implementation effort for the TrB&B is worthwhile, especially in combination with SA. Both the TrB&B and the MIP can be parameterized regarding solution quality, although the latter admits greater walking time in the same runtime.

## 8 Conclusion

In this paper, we consider the mixed-model placement of boxes to minimize worker walking at the moving assembly line. The time-dependent walking time model allows for a much higher precision according to our numerical experiment: in most instances, constant walking time estimates that

**Table 6** Heuristic solutions on instances solved optimally by B&B, grouped by $n$, $m$, or both

| $n$ | $m$ | WNID | | MIP (<10 s) | | MIP (<60 s) | | MIP (<1 h) | | HC | | TrB&B$_{UB\,\mathrm{HC}}$ | | SA | | TrB&B$_{UB\,\mathrm{SA}}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | opt (%) | MPE (%) | opt (%) | MPE (%) | opt (%) | MPE (%) | opt (%) | MPE (%) | opt (%) | MPE (%) | opt (%) | MPE (%) | opt (%) | MPE (%) | opt (%) | MPE (%) |
| ∗ | ∗ | 1 | 121 | 57 | 4.3 | 67 | 2.5 | 85 | 0.6 | 39 | 8.7 | 46 | 5.0 | 68 | 3.3 | 72 | 1.8 |
| 8 | ∗ | 4 | 66 | 100 | 0.0 | 100 | 0.0 | 100 | 0.0 | 82 | 2.5 | 92 | 0.6 | 95 | 0.9 | 98 | 0.1 |
| 12 | ∗ | 0 | 101 | 96 | 0.0 | 100 | 0.0 | 100 | 0.0 | 49 | 6.7 | 63 | 3.0 | 80 | 3.3 | 86 | 0.9 |
| 16 | ∗ | 0 | 127 | 63 | 1.3 | 80 | 0.4 | 99 | 0.0 | 32 | 9.1 | 39 | 5.7 | 66 | 3.6 | 70 | 1.8 |
| 20 | ∗ | 0 | 144 | 25 | 5.9 | 43 | 3.4 | 80 | 0.5 | 20 | 11.6 | 25 | 8.0 | 56 | 4.2 | 59 | 2.7 |
| 24 | ∗ | 0 | 157 | 9 | 12.2 | 21 | 7.9 | 52 | 2.2 | 14 | 13.1 | 16 | 6.9 | 45 | 4.1 | 48 | 2.8 |
| 28 | ∗ | 0 | 164 | 11 | 12.8 | 24 | 7.5 | 57 | 2.1 | 20 | 12.5 | 22 | 9.6 | 51 | 6.0 | 55 | 3.7 |
| ∗ | 2 | 2 | 101 | 61 | 4.0 | 71 | 2.3 | 89 | 0.5 | 39 | 11.2 | 65 | 7.1 | 44 | 8.3 | 70 | 3.4 |
| ∗ | 4 | 0 | 149 | 56 | 4.9 | 66 | 2.9 | 84 | 0.8 | 33 | 10.2 | 66 | 2.4 | 42 | 5.3 | 69 | 1.5 |
| ∗ | 8 | 1 | 113 | 53 | 3.9 | 64 | 2.4 | 82 | 0.6 | 44 | 4.7 | 72 | 0.5 | 52 | 1.4 | 76 | 0.3 |
| 8 | 2 | 8 | 74 | 100 | 0.0 | 100 | 0.0 | 100 | 0.0 | 77 | 4.1 | 87 | 1.2 | 91 | 2.4 | 96 | 0.4 |
| 8 | 4 | 0 | 86 | 100 | 0.0 | 100 | 0.0 | 100 | 0.0 | 70 | 3.3 | 88 | 0.5 | 92 | 0.3 | 97 | 0.1 |
| 8 | 8 | 4 | 39 | 100 | 0.0 | 100 | 0.0 | 100 | 0.0 | 100 | 0.0 | 100 | 0.0 | 100 | 0.0 | 100 | 0.0 |
| 12 | 2 | 1 | 97 | 100 | 0.0 | 100 | 0.0 | 100 | 0.0 | 49 | 10.2 | 57 | 5.7 | 75 | 8.5 | 82 | 2.2 |
| 12 | 4 | 0 | 125 | 100 | 0.0 | 100 | 0.0 | 100 | 0.0 | 44 | 7.4 | 60 | 2.7 | 79 | 1.1 | 85 | 0.6 |
| 12 | 8 | 0 | 82 | 89 | 0.1 | 99 | 0.0 | 100 | 0.0 | 54 | 2.5 | 73 | 0.4 | 85 | 0.1 | 91 | 0.1 |
| 16 | 2 | 0 | 108 | 76 | 1.0 | 91 | 0.3 | 100 | 0.0 | 33 | 12.0 | 36 | 9.4 | 63 | 7.9 | 68 | 3.8 |
| 16 | 4 | 0 | 156 | 65 | 1.5 | 83 | 0.5 | 100 | 0.0 | 28 | 10.9 | 36 | 6.3 | 65 | 2.4 | 69 | 1.4 |
| 16 | 8 | 0 | 117 | 48 | 1.4 | 67 | 0.5 | 98 | 0.0 | 35 | 4.4 | 45 | 1.5 | 69 | 0.4 | 74 | 0.3 |
| 20 | 2 | 0 | 113 | 31 | 5.2 | 53 | 2.7 | 89 | 0.3 | 25 | 13.4 | 27 | 10.8 | 55 | 8.3 | 60 | 4.8 |
| 20 | 4 | 0 | 173 | 22 | 7.0 | 42 | 4.1 | 77 | 0.7 | 16 | 13.5 | 20 | 9.8 | 54 | 3.5 | 56 | 2.6 |
| 20 | 8 | 0 | 146 | 22 | 5.7 | 35 | 3.3 | 72 | 0.6 | 20 | 7.8 | 27 | 3.3 | 59 | 0.8 | 62 | 0.7 |
| 24 | 2 | 0 | 114 | 10 | 11.2 | 22 | 7.1 | 61 | 1.7 | 18 | 14.4 | 19 | 12.3 | 45 | 6.9 | 49 | 5.0 |
| 24 | 4 | 0 | 184 | 8 | 13.8 | 19 | 8.8 | 49 | 2.8 | 10 | 15.5 | 12 | 6.4 | 41 | 4.4 | 43 | 2.8 |
| 24 | 8 | 0 | 172 | 10 | 11.5 | 21 | 7.8 | 46 | 2.1 | 14 | 9.3 | 18 | 1.9 | 49 | 1.1 | 51 | 0.7 |
| 28 | 2 | 0 | 106 | 17 | 12.6 | 31 | 6.8 | 63 | 1.9 | 20 | 17.4 | 23 | 14.2 | 47 | 11.1 | 52 | 6.4 |
| 28 | 4 | 0 | 219 | 6 | 13.7 | 18 | 8.5 | 56 | 2.5 | 16 | 11.5 | 17 | 8.3 | 52 | 2.9 | 55 | 2.1 |
| 28 | 8 | 0 | 196 | 8 | 11.7 | 20 | 7.5 | 46 | 1.9 | 24 | 3.9 | 27 | 2.2 | 57 | 0.4 | 61 | 0.4 |

opt, percentage of instances solved optimally among those that B&B solved within 1 h; MPE, mean percentage walking time error to the instance's minimum walking time
∗ aggregate of all variants of the respective parameter

**Table 7** Heuristic solutions on instances compared to the heuristic MIP (<1 h), grouped by $n$, $m$, or both

| $n$ | $m$ | WNID | | MIP (<10 s) | | MIP (<60 s) | | HC | | TrB&B$_{UB\,\mathrm{HC}}$ | | SA | | TrB&B$_{UB\,\mathrm{SA}}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\leq \phi^{\mathrm{H}}$(%) | HMPE (%) | $\leq \phi^{\mathrm{H}}$(%) | HMPE (%) | $\leq \phi^{\mathrm{H}}$(%) | HMPE (%) | $\leq \phi^{\mathrm{H}}$(%) | HMPE (%) | $\leq \phi^{\mathrm{H}}$(%) | HMPE (%) | $\leq \phi^{\mathrm{H}}$(%) | HMPE (%) | $\leq \phi^{\mathrm{H}}$(%) | HMPE (%) |
| ∗ | ∗ | 1 | 124 | 50 | 5.0 | 60 | 2.8 | 39 | 8.2 | 49 | 4.6 | 72 | 2.3 | 76 | 0.9 |
| 8 | ∗ | 4 | 66 | 100 | 0.0 | 100 | 0.0 | 82 | 2.5 | 92 | 0.6 | 95 | 0.9 | 98 | 0.1 |
| 12 | ∗ | 0 | 101 | 96 | 0.0 | 100 | 0.0 | 49 | 6.7 | 63 | 3.0 | 80 | 3.3 | 86 | 0.9 |
| 16 | ∗ | 0 | 127 | 63 | 1.3 | 81 | 0.4 | 32 | 9.1 | 39 | 5.7 | 66 | 3.6 | 71 | 1.8 |
| 20 | ∗ | 0 | 143 | 25 | 5.4 | 45 | 2.8 | 23 | 11.0 | 29 | 7.4 | 62 | 3.7 | 66 | 2.1 |
| 24 | ∗ | 0 | 151 | 10 | 9.9 | 22 | 5.6 | 22 | 10.7 | 35 | 4.6 | 63 | 1.9 | 67 | 0.6 |
| 28 | ∗ | 0 | 153 | 6 | 13.2 | 13 | 7.8 | 28 | 9.3 | 34 | 6.5 | 66 | 0.8 | 69 | −0.5 |
| ∗ | 2 | 1 | 101 | 55 | 4.6 | 64 | 2.5 | 41 | 10.3 | 68 | 5.9 | 45 | 7.7 | 73 | 2.5 |
| ∗ | 4 | 0 | 148 | 50 | 5.5 | 60 | 3.1 | 33 | 9.6 | 70 | 1.5 | 44 | 5.0 | 74 | 0.6 |
| ∗ | 8 | 1 | 121 | 46 | 4.8 | 56 | 2.8 | 44 | 4.7 | 78 | −0.4 | 57 | 1.1 | 81 | −0.5 |
| 8 | 2 | 8 | 74 | 100 | 0.0 | 100 | 0.0 | 77 | 4.1 | 87 | 1.2 | 91 | 2.4 | 96 | 0.4 |
| 8 | 4 | 0 | 86 | 100 | 0.0 | 100 | 0.0 | 70 | 3.3 | 88 | 0.5 | 92 | 0.3 | 97 | 0.1 |
| 8 | 8 | 4 | 39 | 100 | 0.0 | 100 | 0.0 | 100 | 0.0 | 100 | 0.0 | 100 | 0.0 | 100 | 0.0 |
| 12 | 2 | 1 | 97 | 100 | 0.0 | 100 | 0.0 | 49 | 10.2 | 57 | 5.7 | 75 | 8.5 | 82 | 2.2 |
| 12 | 4 | 0 | 125 | 100 | 0.0 | 100 | 0.0 | 44 | 7.4 | 60 | 2.7 | 79 | 1.1 | 85 | 0.6 |
| 12 | 8 | 0 | 82 | 89 | 0.1 | 99 | 0.0 | 54 | 2.5 | 73 | 0.4 | 85 | 0.1 | 91 | 0.1 |
| 16 | 2 | 0 | 108 | 76 | 1.0 | 91 | 0.3 | 33 | 12.0 | 36 | 9.4 | 64 | 7.9 | 68 | 3.8 |
| 16 | 4 | 0 | 156 | 65 | 1.5 | 83 | 0.5 | 28 | 10.9 | 36 | 6.3 | 65 | 2.4 | 69 | 1.4 |
| 16 | 8 | 0 | 117 | 48 | 1.4 | 67 | 0.5 | 35 | 4.4 | 46 | 1.5 | 70 | 0.4 | 75 | 0.2 |
| 20 | 2 | 0 | 112 | 31 | 4.8 | 54 | 2.4 | 26 | 13.1 | 28 | 10.5 | 57 | 8.0 | 62 | 4.4 |
| 20 | 4 | 0 | 171 | 22 | 6.2 | 44 | 3.4 | 19 | 12.7 | 24 | 9.0 | 60 | 2.8 | 63 | 1.9 |
| 20 | 8 | 0 | 145 | 22 | 5.0 | 38 | 2.7 | 23 | 7.1 | 34 | 2.7 | 70 | 0.2 | 72 | 0.1 |
| 24 | 2 | 0 | 110 | 10 | 9.5 | 23 | 5.4 | 25 | 12.3 | 27 | 10.2 | 57 | 4.7 | 61 | 3.0 |
| 24 | 4 | 0 | 176 | 9 | 10.8 | 21 | 5.9 | 18 | 12.6 | 29 | 3.7 | 60 | 1.8 | 64 | 0.1 |
| 24 | 8 | 0 | 166 | 10 | 9.3 | 23 | 5.6 | 24 | 7.4 | 50 | −0.2 | 72 | −0.9 | 77 | −1.3 |
| 28 | 2 | 0 | 105 | 10 | 12.2 | 18 | 6.8 | 33 | 10.4 | 34 | 9.0 | 63 | 3.9 | 67 | 1.5 |
| 28 | 4 | 0 | 176 | 4 | 14.4 | 10 | 8.7 | 22 | 10.8 | 26 | 8.0 | 62 | 0.6 | 66 | −0.6 |
| 28 | 8 | 0 | 180 | 4 | 13.0 | 12 | 8.0 | 29 | 6.7 | 42 | 2.5 | 72 | −2.2 | 74 | −2.4 |

$\leq \phi^{\mathrm{H}}$, percentage of instances solved at least as well as the heuristic objective value $\phi^{\mathrm{H}}$ that MIP attained within 1 h; HMPE, mean percentage walking time error to the walking time attained by the MIP within 1 h (heuristically)

∗ aggregate of all variants of the respective parameter

include 95% of cases are at least 51% higher than with our time-dependent model (see Sect. 7.5).

We prove that this optimization problem is NP-hard in the strong sense for any number of product models. We observe that moderately sized instances with up to 16 jobs can be solved with a mixed-integer program that employs disjunctive sequencing constraints to avoid box overlapping. For larger instances, we construct branch and bound-based algorithms. A Lagrangian relaxation leads to a lower bound that is solved in quadratic time. This bound is employed in a branch and bound search, for which a truncated search tree yields a heuristic. Also, we describe an intuitive heuristic that is complemented with a local search and simulated annealing to find an initial upper bound.

The numerical results indicate that our exact branch and bound based algorithms provide superior runtime and quality compared to solving a mixed-integer program. The runtime of the best heuristic typically remains below 1 s, contributing to an interactive planning experience that is more accurate and permits less safety buffer time.

**Funding** Open access funding provided by ZHAW Zurich University of Applied Sciences.

## Declarations

**Conflict of interest** The author has no competing interests to declare that are relevant to the content of this article.



## References

Álvarez-Miranda, E., & Pereira, J. (2019). On the complexity of assembly line balancing problems. *Computers & Operations Research, 108*, 182–186. https://doi.org/10.1016/j.cor.2019.04.005

Andrés, C., Miralles, C., & Pastor, R. (2008). Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research, 187*(3), 1212–1223. https://doi.org/10.1016/j.ejor.2006.07.044

Baker, K. R., & Keller, B. (2010). Solving the single-machine sequencing problem using integer programming. *Computers & Industrial Engineering, 59*(4), 730–735. https://doi.org/10.1016/j.cie.2010.07.028

Balas, E. (1985). On the facial structure of scheduling polyhedra. *Mathematical Programming Study, 24*, 179–218. https://doi.org/10.1007/BFb0121051

Battaïa, O., & Dolgui, A. (2013). A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics, 142*(2), 259–277. https://doi.org/10.1016/j.ijpe.2012.10.020

Battaïa, O., & Dolgui, A. (2022). Hybridizations in line balancing problems: A comprehensive review on new trends and formulations. *International Journal of Production Economics, 250*, 108673. https://doi.org/10.1016/j.ijpe.2022.108673

Bautista, J., & Pereira, J. (2007). Ant algorithms for a time and space constrained assembly line balancing problem. *European Journal of Operational Research, 177*(3), 2016–2032. https://doi.org/10.1016/j.ejor.2005.12.017

Becker, C., & Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research, 168*(3), 694–715. https://doi.org/10.1016/j.ejor.2004.07.023

Becker, C., & Scholl, A. (2009). Balancing assembly lines with variable parallel workplaces: Problem definition and effective solution procedure. *European Journal of Operational Research, 199*(2), 359–374. https://doi.org/10.1016/j.ejor.2008.11.051

Boysen, N., Fliedner, M., & Scholl, A. (2008). Sequencing mixed-model assembly lines to minimize part inventory cost. *OR Spectrum, 30*(3), 611–633. https://doi.org/10.1007/s00291-007-0095-2

Boysen, N., Fliedner, M., & Scholl, A. (2009a). Level Scheduling for batched JIT supply. *Flexible Services and Manufacturing Journal, 21*(1–2), 31–50. https://doi.org/10.1007/s10696-009-9058-z

Boysen, N., Fliedner, M., & Scholl, A. (2009b). Level scheduling of mixed-model assembly lines under storage constraints. *International Journal of Production Research, 47*(10), 2669–2684. https://doi.org/10.1080/00207540701725067

Boysen, N., Fliedner, M., & Scholl, A. (2009c). Sequencing mixed-model assembly lines: Survey, classification and model critique. *European Journal of Operational Research, 192*(2), 25. https://doi.org/10.1016/j.ejor.2007.09.013

Boysen, N., Scholl, A., & Wopperer, N. (2012). Resequencing of mixed-model assembly lines: Survey and research agenda. *European Journal of Operational Research, 216*(3), 594–604. https://doi.org/10.1016/j.ejor.2011.08.009

Boysen, N., Emde, S., Hoeck, M., & Kauderer, M. (2015). Part logistics in the automotive industry: Decision problems, literature review and research agenda. *European Journal of Operational Research, 242*(1), 107–120. https://doi.org/10.1016/j.ejor.2014.09.065

Boysen, N., Schulze, P., & Scholl, A. (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

Brent, R. P. (1971). An algorithm with guaranteed convergence for finding a zero of a function. *The Computer Journal, 14*(4), 422–425. https://doi.org/10.1093/comjnl/14.4.422

Bukchin, Y., & Meller, R. D. (2005). A space allocation algorithm for assembly line components. *IIE Transactions, 37*(1), 51–61. https://doi.org/10.1080/07408170590516854

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications, 45*(1), 41–51. https://doi.org/10.1007/BF00940812

Esmaeilbeigi, R., Naderi, B., & Charkhgard, P. (2016). New formulations for the setup assembly line balancing and scheduling problem. *OR Spectrum, 38*(2), 493–518. https://doi.org/10.1007/s00291-016-0433-3

Farahani, M. H., & Hosseini, L. (2013). Minimizing cycle time in single machine scheduling with start time-dependent processing times. *The International Journal of Advanced Manufacturing*

*Technology, 64*(9), 1479–1486. https://doi.org/10.1007/s00170-012-4116-1
Finnsgård, C., Wänström, C., Medbo, L., & Neumann, W. P. (2011). Impact of materials exposure on assembly workstation performance. *International Journal of Production Research, 49*(24), 7253–7274. https://doi.org/10.1080/00207543.2010.503202
Fisher, M. L. (2004). The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science, 50*(12 Supplement), 1861–1871. https://doi.org/10.1287/mnsc.1040.0263
Ford, H., & Crowther, S. (1922). *My life and work*. Doubleday Page & Co.
Garey, M. R., & Johnson, D. S. (1975). Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing, 4*(4), 397–411. https://doi.org/10.1137/0204035
Garey, M. R., & Johnson, D. S. (1978). "Strong" NP-completeness results: Motivation, examples, and implications. *Journal of the ACM, 25*(3), 499–508. https://doi.org/10.1145/322077.322090
Garey, M. R., Tarjan, R. E., & Wilfong, G. T. (1988). One-processor scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research, 13*(2), 330–348. https://doi.org/10.2307/3689828
Gawiejnowicz S (2020a) Models and Algorithms of Time-Dependent Scheduling, 2nd edn. Monographs in Theoretical Computer Science, Springer, Berlin, Heidelberg, https://doi.org/10.1007/978-3-662-59362-2
Gawiejnowicz, S. (2020b). A review of four decades of time-dependent scheduling: Main results, new topics, and open problems. *Journal of Scheduling, 23*(1), 3–47. https://doi.org/10.1007/s10951-019-00630-w
Hajek, B. (1988). Cooling schedules for optimal annealing. *Mathematics of Operations Research, 13*(2), 311–329. https://doi.org/10.1287/moor.13.2.311
Held, M., Wolfe, P., & Crowder, H. P. (1974). Validation of subgradient optimization. *Mathematical Programming, 6*(1), 62–88. https://doi.org/10.1007/BF01580223
Jaehn, F., & Sedding, H. A. (2016). Scheduling with time-dependent discrepancy times. *Journal of Scheduling, 19*(6), 737–757. https://doi.org/10.1007/s10951-016-0472-2
Kawase, Y., Makino, K., & Seimi, K. (2018). Optimal composition ordering problems for piecewise linear functions. *Algorithmica, 80*(7), 2134–2159. https://doi.org/10.1007/s00453-017-0397-y
Keha, A. B., Khowala, K., & Fowler, J. W. (2009). Mixed integer programming formulations for single machine scheduling problems. *Computers & Industrial Engineering, 56*(1), 357–367. https://doi.org/10.1016/j.cie.2008.06.008
Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science, 220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671
Klampfl, E., Gusikhin, O., & Rossi, G. (2006). Optimization of workcell layouts in a mixed-model assembly line environment. *International Journal of Flexible Manufacturing Systems, 17*(4), 277–299. https://doi.org/10.1007/s10696-006-9029-6
Kononov, A. V. (1998). Problems in scheduling theory on a single machine with job durations proportional to an arbitrary function. *Diskretnyĭ Analiz i Issledovanie Operatsiĭ, 5*(3), 17–37.
Limère, V., Landeghem, H. V., & Goetschalckx, M. (2015). A decision model for kitting and line stocking with variable operator walking distances. *Assembly Automation, 35*(1), 47–56. https://doi.org/10.1108/AA-05-2014-043
Manne, A. S. (1960). On the job-shop scheduling problem. *Operations Research, 8*(2), 219–223. https://doi.org/10.1287/opre.8.2.219
Müllerklein, D., Fontaine, P., & Ostermeier, F. (2022). Integrated consideration of assembly line scheduling and feeding: A new model and case study from the automotive industry. *Computers & Industrial Engineering, 170*, 108288. https://doi.org/10.1016/j.cie.2022.108288
Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in C: The art of scientific computing*. Cambridge University Press.
Queyranne, M. (1993). Structure of a simple scheduling polyhedron. *Mathematical Programming, 58*(1–3), 263–285. https://doi.org/10.1007/BF01581271
Queyranne, M., & Schulz, A. S. (1994). Polyhedral approaches to machine scheduling. (p. 3). Tech. rep.: Technische Universität Berlin, Fachbereich.
Salveson, M. E. (1955). The assembly line balancing problem. *Journal of Industrial Engineering, 6*(3), 18–25.
Schmid, N. A., & Limère, V. (2019). A classification of tactical assembly line feeding problems. *International Journal of Production Research, 57*(24), 7586–7609. https://doi.org/10.1080/00207543.2019.1581957
Schmid, N. A., Limère, V., & Raa, B. (2021). Mixed model assembly line feeding with discrete location assignments and variable station space. *Omega, 102*, 102286. https://doi.org/10.1016/j.omega.2020.102286
Scholl, A., Boysen, N., & Fliedner, M. (2013). The assembly line balancing and scheduling problem with sequence-dependent setup times: Problem extension, model formulation and efficient heuristics. *OR Spectrum, 35*(1), 291–320. https://doi.org/10.1007/s00291-011-0265-0
Sedding, H. A. (2017) Box placement as time dependent scheduling to reduce automotive assembly line worker walk times. In *Proceedings of the 13th Workshop on Models and Algorithms for Planning and Scheduling Problems*, Seeon, Germany, pp 92–94.
Sedding, H. A. (2020a). Line side placement for shorter assembly line worker paths. *IISE Transactions, 52*(2), 181–198. https://doi.org/10.1080/24725854.2018.1508929
Sedding, H. A. (2020b). Scheduling jobs with a V-shaped time-dependent processing time. *Journal of Scheduling, 23*(6), 751–768. https://doi.org/10.1007/s10951-020-00665-4
Sedding, H. A. (2020c). *Time-dependent path scheduling: Algorithmic minimization of walking time at the moving assembly line*. Springer. https://doi.org/10.1007/978-3-658-28415-2
Sedding, H. A. (2021). A lower bound for sequentially placing boxes at the moving assembly line to minimize walking time. In *Proceedings of the 3rd International Workshop on Dynamic Scheduling Problems*, Adam Mickiewicz University, Poznań, Poland, pp 63–69.
Smith, W. E. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly, 3*(1–2), 59–66. https://doi.org/10.1002/nav.3800030106
Sternatz, J. (2014). Enhanced multi-Hoffmann heuristic for efficiently solving real-world assembly line balancing problems in automotive industry. *European Journal of Operational Research, 235*(3), 740–754. https://doi.org/10.1016/j.ejor.2013.11.005
Sternatz, J. (2015). The joint line balancing and material supply problem. *International Journal of Production Economics, 159*, 304–318. https://doi.org/10.1016/j.ijpe.2014.07.022
Thomopoulos, N. T. (1967). Line balancing-sequencing for mixed-model assembly. *Management Science, 14*(2), B59–B75. https://doi.org/10.1287/mnsc.14.2.B59