

Fedtke, Stefan; Boysen, Nils; Schumacher, Patrick

**Article — Published Version**

## In-line kitting for part feeding of assembly lines: workload balancing and storage assignment to reduce the workers' walking effort

OR Spectrum

**Provided in Cooperation with:**

Springer Nature

*Suggested Citation:* Fedtke, Stefan; Boysen, Nils; Schumacher, Patrick (2023) : In-line kitting for part feeding of assembly lines: workload balancing and storage assignment to reduce the workers' walking effort, OR Spectrum, ISSN 1436-6304, Springer, Berlin, Heidelberg, Vol. 45, Iss. 3, pp. 717-758,  
<https://doi.org/10.1007/s00291-023-00723-1>

This Version is available at:

<https://hdl.handle.net/10419/305218>

**Standard-Nutzungsbedingungen:**

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

**Terms of use:**

*Documents in EconStor may be saved and copied for your personal and scholarly purposes.*

*You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.*

*If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.*



<https://creativecommons.org/licenses/by/4.0/>



# In-line kitting for part feeding of assembly lines: workload balancing and storage assignment to reduce the workers' walking effort

Stefan Fedtke<sup>1</sup> · Nils Boysen<sup>1</sup> · Patrick Schumacher<sup>2</sup>

Received: 18 February 2022 / Accepted: 20 April 2023 / Published online: 8 May 2023  
© The Author(s) 2023

## Abstract

An efficient part feeding is among the top challenges of many mass producers applying mixed-model assembly lines, for instance, in the automotive industry. This paper introduces a novel part feeding policy applied by a large German assembly plant for car engines: In-line kitting. Under this policy, the first stations of the line do not execute assembly operations, but are reserved for picking parts while passing containers of stock-keeping units (SKUs) arranged along the line. In this way, the parts are collected in traveling kits moving along with each workpiece on the conveyor, so that later assembly stations have the required parts directly available and do not lose precious labor time for unproductive parts handling. A major operational challenge when applying this part feeding policy is the walking effort for the human pickers while putting the SKUs of their respective stations into the traveling kits of the passing workpieces. Due to a high product variety, a large number of comparatively bulky SKU containers have to fit into each station, so that the walking distance to be covered by a worker during a work shift exceeds multiple kilometers. We show that this physical burden can be reduced significantly by balancing the workload among stations and optimizing the storage assignment of SKU containers within each in-line kitting station. We formulate the resulting optimization problem and provide suited solution procedures. Our computational study shows that the walking distance of pickers can be reduced significantly without producing any additional costs.

**Keywords** Mixed-model assembly lines · Part feeding · In-line kitting · Optimization

---

✉ Stefan Fedtke  
stefan.fedtke@uni-jena.de  
<https://www.om.uni-jena.de>

Extended author information available on the last page of the article

## 1 Introduction

The mid-sized assembly plant of German car producer BMW in Dingolfing has to timely supply their assembly stations with parts arriving in more than 13,000 containers delivered by about 600 suppliers on more than 400 trucks on each average workday (Battini et al. 2013). These figures elucidate that part logistics is among the greatest challenges of today's mass-producers, especially in the automotive industry (Boysen et al. 2015). Therefore, it is anything but surprising that selecting the right part feeding policy has received much attention both among practitioners and scientists in the recent years. Part feeding is the in-house logistics process which moves parts from receipt after their arrival at the assembly plant up to their respective workstations where they are finally assembled into the workpieces. Note that a similar definition is provided by Choi and Lee (2002).

A part feeding policy specifies this in-house logistics process for a subset of parts. To do so, a specific policy has to choose among the alternative elements of the four basic process elements defined in Fig. 1. Note that previous papers (i.e., Battini et al. 2013; Boysen et al. 2015; Kilic and Durmusoglu 2015; Schmid and Limère 2019) apply similar, yet slightly different elements of part feeding:

- Initially, the part feeding process is *triggered* either by previous consumption or by prospective demands, so that parts are either pulled (e.g., via a Kanban mechanism or by part inventory falling below a predefined reorder level) or pushed (i.e., according to the demands defined by the workpieces produced in the next production cycles) toward their respective assembly stations.
- Once the demand for parts is defined, the respective parts need to be retrieved from their *storage* positions. This can either be a central (receiving) warehouse, a decentralized logistics area directly on the shop floor (often called supermarket, see Emde and Boysen 2012), or local storage space directly next to the assembly line.
- Then, parts need to be moved by a suitable *transportation* device, which may be a traditional forklift, a tow train (i.e., a manned or automated tugger vehicle towing a handful of wagons (see Emde et al. 2012), or a conveyor belt.
- Finally, the parts need to be positioned next to the line, which is also called their *line-side presentation* (see also Limère et al. 2012; Schmid and Limère 2019).

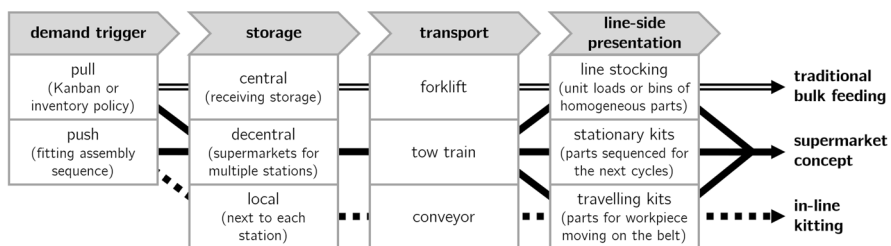


Fig. 1 Part feeding policies and their basic elements

Line stocking places containers (e.g., large unit loads or smaller bins) each filled with homogeneous pieces of the same stock-keeping unit (SKU) directly in the respective workstations (e.g., multiple bins each containing a specific exterior mirror variant), so that the assembly workers have to identify and fetch the respective parts required in each production cycle. To avoid unproductive part sorting of (highly paid) assembly workers stationary kits are placed next to the line and contain pre-sorted pieces of different SKUs ordered according prospective demand (e.g., the exterior mirror variants sequenced for the next production cycles). Traveling kits, instead, contain pre-sorted pieces of multiple SKUs for a single workpiece (e.g., the exterior and interior mirrors and other parts for a specific car) and accompany their workpiece on the conveyor.

The most widespread part feeding policies resulting from choosing a specific element for each process step are traditional bulk feeding and the supermarket concept (see Fig. 1). Triggered by previous part consumption, the former applies forklifts to deliver large homogeneous unit loads from central storage to the line and leaves the sorting of parts to the assembly workers (Boysen and Bock 2011). Decentralized storage in supermarkets allows for a more reactive part supply in smaller bins of pre-sorted products and applies tow trains delivering multiple workstations along fixed routes (Emde et al. 2012). The supermarket concept can be applied in a push and pull environment and can service all kinds of line-side presentation. Identifying a suitable part feeding policy for each part is an important practical decision task, and the pros and cons of each policy are vividly discussed in the scientific literature (e.g., Hanson and Brolin 2013; Sali et al. 2015). Moreover, quantitative decision models for selecting the right feeding policy (e.g., Limère et al. 2012; Caputo et al. 2015, 2018) and its interdependency with assembly line balancing (Sternatz 2015; Battini et al. 2017; Nourmohammadi et al. 2019) have been investigated. The manifold literature in this area is summarized by the survey papers of Boysen et al. (2015), Kilic and Durmusoglu (2015), and Schmid and Limère (2019).

This paper treats a part feeding policy, which (to the best of the authors' knowledge) has been overlooked by previous research. We elaborate this novel policy, which combines the elements marked by the dotted arrow in Fig. 1, in the following section.

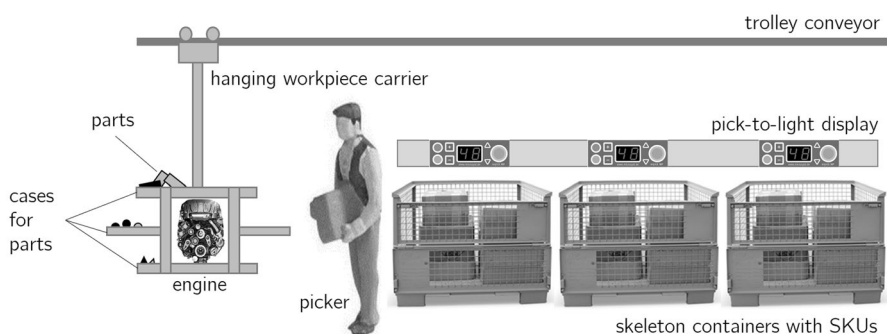
## 1.1 In-line kitting

We saw in-line kitting at work in the assembly plant of MDC Power in Köllda, which is one of Germany's largest assembly plants for car engines and belongs to the Daimler group. The engines are produced on multiple mixed-model assembly lines in lot-size one. Each engine moves from station to station on a hanging workpiece carrier transported by a trolley conveyor. The first stations of the line are reserved for in-line kitting, which means that the specific parts dedicated to each engine are directly added to the cases of their carriers. The carriers, thus, also serve as traveling kits. In this way, the parts required in later assembly stations are directly available

on each workpiece carrier, so that unproductive walking of assembly workers for fetching parts from racks and containers next to the line is eliminated.

The in-line kitting segment of the assembly line, whose schematic layout is depicted in Fig. 2, is separated into distinct areas called *stations* each operated by a single dedicated worker. To better distinguish these workers solely concerned with part logistics from assembly workers of later stations, we call them the *pickers*. Each station consists of a given set of successively arranged skeleton containers each containing homogeneous pieces of a specific SKU. Pick-to-light displays indicate which SKUs (and how many of each) have to be added to the cases of the engine carrier currently entering a station. The picker accompanies the engine, which is continuously moved forward by the conveyor, adds the indicated parts whenever they pass the respective SKU, and accredits the picks (by pushing a button on the pick-to-light display) until the end of the station is reached. Naturally, there are no precedence constraints that restrict the sequence in which the SKUs are packed on the carriers, so that the pickers are advised to follow the sequence in which the SKU containers are arranged along the line. Then, the picker rushes backward toward the successive engine of the next cycle. Since engines are produced in lot-size one, each engine has its different part demands. Given the specific part demands of the next cycle, the picker repeats the kitting process for the next workpiece. Note that the kitting line segment itself is replenished via a bulk feeding process (see Fig. 1), where forklifts replace (almost) empty with filled skeleton containers delivered from central storage once a specific reorder level is reached.

The biggest advantage of in-line kitting is that assembly workers, who are specially trained and earn higher wages, are relieved from part fetching. The parts are directly available on the cases of each engine carrier, so that unproductive time for walking to part containers next to the line as well as identifying and retrieving the required parts from there is eliminated. These work contents are executed by less costly pickers in the kitting area, who are typically assigned to another pay scale group. Another advantage is that there is no need to store part containers in assembly stations, which is especially valuable for small sized workpieces such as car engines where stations are much smaller compared to automobile assembly. These advantages, however, could also be realized by moving traveling kits assembled in



**Fig. 2** Schematic layout of the in-line kitting line segment

some remote logistics area onto the conveyor. In engine assembly, however, there is no belt conveyor, so that a traveling kit would have to be attached to the carrier, which requires a suited (and stable) technical solution. Furthermore, in-line kitting moves part logistics under the strict regime of a paced flow process, so that part feeding is under tight control and picking errors can easily be reduced, e.g., by adding an extra weighing mechanism for the added parts to the workpiece carrier.

On the negative side, in-line kitting requires an extended conveyor and blocks space directly on the shop floor where room is notoriously scarce. Car engines and the part variety they require, however, are smaller compared to typical assembly line products such as cars, so that the prolongation of the line due to the additional kitting segment is moderate. Furthermore, trolley conveyors are less costly than large ground conveyors for cars, so that in the specific case of engine assembly the additional investment for in-line kitting seems acceptable. These characteristics question the suitability of in-line kitting for car assembly, but make it a valid part feeding alternative for small product assembly, e.g., in the electronics industry.

## 1.2 How to reduce the pickers' walking effort

For our engine producer, one of the greatest operational challenges when applying in-line kitting is the high physical stress for the picker workforce. Since the parts in engine assembly are small and light-weighted, ergonomic stress caused by lifting, which is a major cause for long-term back injuries in many industries (e.g., Grosse et al. 2015; Otto et al. 2017), is less of an issue. Instead, accompanying the workpieces while putting the indicated SKUs into the engine carriers and walking back to the start of the station again and again during a work shift accumulates to a considerable total walking distance for each picker. The station each picker services is relatively large due to the following two reasons:

- The standardized skeleton containers for the SKUs applied by our engine producer are comparatively large (i.e., they have the same outline as a europallet with a width of 800 mm. Due to the small size of the parts, these containers have the advantage that plenty pieces of a SKU fit into each container, so that the logistics effort for replenishing the kitting stations is reduced.
- Each assembly line produces not only a single type of engine, but multiple variants in a mixed-model setting (see, e.g., Boysen et al. 2009). Therefore, each station contains many SKUs, but only some of them are required by any type of engine. Recall that the part variety is small compared to car assembly but still considerable.

This results in station sizes of dozens of meters. At a cycle time between 60 and 90 seconds, which are typical values in the automotive industry (see Boysen et al. 2015), the walking distance quickly accumulates to several kilometers per working shift. If a worker has to service 50 SKUs whose containers have an identical width of 800 mm, the resulting area width is 40 m. Under the assumption that the picker always has to traverse the complete area at a cycle time of 60 seconds, the

worst-case walking distance of a 7-hour working shift amounts to 17 km. We admit that this is rather a ballpark calculation to highlight the potential worst-case walking effort of workers. In the real-world, these extremes are seldom reached due to more generous break regulations, job rotation, or longer picking times that enforce smaller stations. However, the managers of our engine producer told us that more than 10-km walking per shift regularly occur, which they suspect as a major source for the comparatively high level of absenteeism and labor turnover in the kitting area. Furthermore, the kitting area employs many impaired workers not suitable for the even more demanding workplaces in the assembly stations, so that the managers (and the representatives of the trade union) aim to reduce the pickers' walking distances.

Obvious levers for reducing the walking distance in the kitting segment are a larger workforce or an extended cycle time. The former, however, increases wage costs and the latter reduces the output of the assembly line. Hence, the management rather prioritize less costly countermeasures, such as the following. Pickers need not traverse their complete stations in every cycle. Once pickers recognize that the remaining pick-to-light displays indicate no further demand for the current engine, they can let the workpiece pass the remainder of their station unattended and can prematurely return back toward the next workpiece. Such a reduction in the walking effort can be influenced by the following two decisions tasks:

- *Workload balancing:* Naturally, only SKUs required by the current workpiece and located within the respective station have to be serviced by a picker. Thus, distributing the SKUs among the kitting stations in a fair manner enables a workload balancing for pickers. In this way, exceptionally large walking distances of single pickers can be avoided. Thus, the partitioning of SKUs among kitting stations influences the walking effort of the pickers, and this decision task is tackled by the optimization procedures developed in this paper. Workload balancing alone (without the subsequent storage assignment within each station, see below) resembles the famous assembly line balancing problem (see Boysen et al. 2022, 2008), which decides on the division of labor among subsequent assembly stations. In our case, we face a special line balancing problem without precedence constraints (i.e., the SKUs can be processed in any sequence) and including the model mix (see Merengo et al. 1999), because each engine type requires a different subset of SKUs. The biggest difference to existing research, however, is the interdependence with the following decision task.
- *Storage assignment:* Another lever, which reduces picker walking without deteriorating picking performance, is the storage assignment of SKUs within each kitting station. It decides on the sequence, in which the set of SKUs of a station are arranged along the conveyor. In our setting, the picking orders, i.e., the subset of SKUs demanded by each engine, are known with certainty. If we find a storage assignment, such that the spread of each picking order, i.e., its distance from the first to the last container required by the respective order, is reduced, then pickers need not traverse their complete stations, but only a smaller subsegment. For this purpose, we integrate the storage assignment of SKUs to reduce the pickers' walking effort into our optimization procedures. Storage assignment of SKUs is

one of the classical decision problems of warehousing research (see De Koster et al. 2007). This also holds true for a special order picking system called trolley line picking (see Boysen et al. 2021). Here, hanging trolleys each assigned to a specific customer order pass by SKUs, and it is the duty of human pickers to add requested SKUs to the trolleys. Thus, trolley line picking directly transfers the basic setup of in-line kitting to the warehouse domain. Here, however, customer orders are typically not known with certainty, when having to decide on the storage assignment, we have no continuously moving line, but trolleys automatically stop in front of requested SKUs, and pickers need not strictly pick one order after the other but can switch among waiting trolleys (see Füllner et al. 2019). Furthermore, the main lever to improve these systems is the sequence, in which the orders are processed (see Füllner et al. 2019). This lever, however, is not available to us, as is argued in the following.

The production sequence, in which the engines are sent down the line, can also relieve pickers. If the storage position of the last SKU within a picker's station demanded by the previous engine is close to the storage position of the first SKU of the subsequent engine, the walk-back distance between successive engines can be reduced. In-line kitting, however, is typically dominated by the subsequent assembly stations, which have a complex demand for specific production sequences obeying different technical restrictions and leveling constraints (see Boysen et al. 2009). Thus, determining the model sequences according to the walking effort of kitting workers is, typically, not an option. Note that other industrial settings (e.g., where heavier or bulkier parts are to be handled) suggest a more holistic view on the physical strain of workers that also includes the ergonomic burden caused by lifting and unfavorable body postures. Blueprints on appropriate measures are provided by the order picking literature (see, e.g., Calzavara et al. 2017; Glock et al. 2019). An interesting future research issue in this direction would, for instance, be field tests that investigate for which types of products the total walking distance is a sufficient proxy for the total physical burden and for which more detailed models should be selected.

### 1.3 Contribution and paper structure

After the paper's first contribution, namely introducing and discussing the in-line kitting part feeding policy, the remainder of the paper is concerned with reducing the walking effort of pickers putting together the traveling kits under the in-line kitting policy. For this purpose, we formulate the joint workload balancing and storage assignment problem, which combines both levers for reducing the pickers' total walking distances in a holistic problem setting. Unfortunately, our computational results will show that a mixed-integer model for the holistic problem cannot be solved by an off-the-shelf solver for instances of real-world size. For solving these instances, we rather introduce different decomposition approaches addressing both decision parts in an iterative manner. Once a suitable heuristic solution procedure is available (and proven successful in a comprehensive performance test), we can



apply this algorithm to explore to what extent the total walking distance of pickers can be reduced in our case study. Our results indicate that the walking distances of pickers can be reduced significantly without deteriorating picking performance.

The remainder of the paper is structured as follows. Section 2 elaborates our optimization task and provides a mixed-integer model for the holistic problem. Two competing heuristic decomposition approaches, namely, balance first, sequence second and (vice versa) sequence first, balance second, are presented in Sect. 3. The computational performance of our solution methods is investigated in Sect. 4. This section also contains further tests on managerial issues, where we explore the possible reduction of the pickers' walking effort. Finally, Sect. 5 concludes the paper.

## 2 The joint workload balancing and storage assignment problem

This section treats the joint workload balancing and storage assignment problem, which we dub WBSAP. Section 2.1 defines the problem, and Sect. 2.2 discusses our basic assumptions and investigates its complexity status. Finally, Sect. 2.3 presents a mixed-integer programming (MIP) model for WBSAP.

### 2.1 Problem definition

Consider a mixed-model assembly line producing  $n$  different types of car engines. Each engine  $i \in I = \{1, 2, \dots, n\}$  is produced with a given frequency  $f_i$  within the planning horizon and requires a specific subset of parts for assembly. We define the total SKU set of parts by  $S = \{1, 2, \dots, |S|\}$  and the subset of SKUs required by engine  $i \in I$  constitutes its associated picking order  $O_i \subset S$  with  $O = \{O_1, O_2, \dots, O_n\}$ . Each SKU is available in a separate skeleton container. All containers are standardized and have identical size; they only differ in the dedicated SKU they contain. Note that we discuss this and all further assumptions in more detail below. The storage positions  $P = \{1, 2, \dots, |P| = |S|\}$  along the line are sorted in ascending order of their index. The part demands have to be fulfilled by a kitting line segment operated by a picking workforce of given size. Each picker operates a separate station, so that we have a given set  $K = \{1, \dots, |K|\}$  of kitting stations.

Our decision task is to partition the total SKU set among the stations and to decide on the storage assignment within each station. These two decisions can be encoded by  $(\Phi, \phi)$ , i.e.,

- a family of sets  $\Phi = \{\Phi_1, \Phi_2, \dots, \Phi_{|K|}\}$  with  $\Phi_k$  defining the subset of SKUs assigned to kitting station  $k$  and
- a SKU sequence  $\phi$ , i.e., a permutation of  $S$  defining the storage assignment of each SKU along the line.

We call a solution  $(\Phi, \phi)$  feasible, if the following three conditions are met:

- $\Phi_k \cap \Phi_{k'} = \emptyset$  for all  $k, k' \in K$  with  $k \neq k'$ , that is, a SKU cannot be assigned to more than one station at a time,
- $\bigcup_{k \in K} \Phi_k = S$ , that is, each SKU is assigned to at least one station, and
- $[\exists k \in K, p, p' \in P : p < p' \wedge \phi(p), \phi(p') \in \Phi_k] \rightarrow \phi(p+1) \in \Phi_k$ , that is, only consecutive SKUs can be assigned to the same station.

Our aim is to reduce the pickers' walking effort. Among all feasible solutions, we, thus, seek those which allow to assemble each picking order in a small subarea of each station. In this way, each picker does not have to traverse her complete kitting station in each cycle. We formalize this aim by the order spread  $\delta_{i,k}(\Phi, \phi)$ , which defines the number of SKU containers between the first and the last occurrence of a SKU required by order  $O_i$  within the storage assignment  $\phi$  of station  $k$ . Specifically, order spread  $\delta_{i,k}(\Phi, \phi)$  of picking order  $O_i \in O$  in station  $k \in K$  is defined as follows:

$$\delta_{i,k}(\Phi, \phi) = \begin{cases} \max_{s \in (O_i \cap \Phi_k)} \{\phi^{-1}(s)\} - \min_{s \in (O_i \cap \Phi_k)} \{\phi^{-1}(s)\} + 1, & \text{if } O_i \cap \Phi_k \neq \emptyset \\ 0, & \text{if } O_i \cap \Phi_k = \emptyset \end{cases} \quad \forall i \in I, k \in K, \quad (1)$$

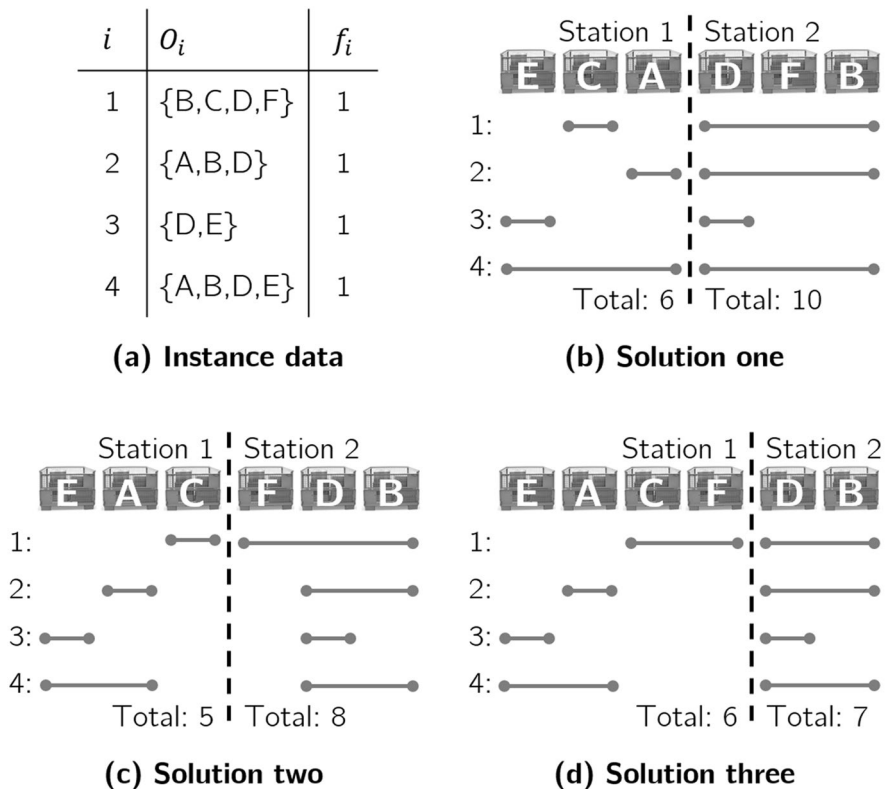
where function  $\phi^{-1}(s)$  returns the storage position  $p \in P = S$  of SKU  $s$  within storage assignment  $\phi$ .

Given the standardized container sizes we presuppose, the order spread directly defines the distance each picker has to accompany an order while picking. Unfortunately, at the point in time storage assignments are planned, the production sequence of engines is, typically, not available, so that we cannot exactly anticipate the total walking distance. To do so, the storage assignment, which defines the walking distances during order picking, and the exact order sequence, which defines the walk-back distance between two successive orders for walking from the last position of the predecessor to the first of the successor, both need to be available. As the latter information is not at hand, we have to do without and aim to minimize the total weighted order spread

$$Z(\Phi, \phi) = \max_{k \in K} \left\{ \sum_{i \in I} f_i \cdot \delta_{i,k}(\Phi, \phi) \right\}. \quad (2)$$

In our case, it is well-known how many of each engine type are to be produced within the next working shifts (but not their exact production sequence), so that it seems advisable to weigh the order spreads  $\delta_{i,k}(\Phi, \phi)$  with picking frequencies  $f_i$ . We dub the problem that seeks the minimum total weighted order spread  $Z(\Phi, \phi)$  among all feasible storage assignments  $\phi$ , the joint workload balancing and storage assignment problem (WBSAP).

*Example:* Consider the example data given in Fig. 3a. Four engines  $i = 1, \dots, 4$  each produced once during the planning horizon have different parts demands for six SKUs  $S = \{A, B, \dots, F\}$ . These picking orders have to be fulfilled by two workers in kitting stations  $k = 1, 2$ . Solution one depicted in Fig. 3b shows an objective of  $Z(\Phi, \phi) = 10$ , constituted by the total walking distance of Station 2 exceeding that of Station 1. Solution two (Fig. 3c) reduces the maximum total walking distance



**Fig. 3** Example for WBSAP

to  $Z(\Phi, \phi) = 8$  by altering the storage assignment within both stations, whereas the division of labor among both stations remains unaltered. Balancing the workload by reassigning SKUs among both stations leads to Solution three and a further reduction to  $Z(\Phi, \phi) = 7$  depicted in Fig. 3d.

## 2.2 Assumptions and complexity

After having defined our optimization problem, we summarize the simplifying assumptions (explicitly and implicitly) contained in our problem setting:

- We aim at a reduction in the physical effort of pickers induced by their total walking distance. We consider this general aim by minimizing the maximum walking distance among all stations. This is considered to be more fair compared to a min-sum objective where the total walking distance accumulated over all stations may be smaller, but some pickers may receive exceptionally large walking distances.

- Furthermore, we want to relieve pickers without causing additional costs, so that we assume a given picker workforce. In line with the setting of our engine producer, each picker operates a dedicated kitting station defined by a fixed subset of SKUs. Dynamic workload sharing within changing station borders, for instance enabled by the bucket brigade protocol (Bartholdi and Eisenstein 1996), or the relieve promised by job rotation (Otto and Scholl 2013) are not considered.
- Assigning storage positions is a planning task executed over a mid-term planning horizon (De Koster et al. 2007). Thus, the composition  $O_i$  of picking orders and their picking frequencies  $f_i$  may be available, but not the specific order sequences. In the automotive industry, production sequences frequently need to be altered briefly before production starts, e.g., due to missing parts not timely delivered (Boysen et al. 2009). Thus, we cannot exactly quantify the picker's total walking distance, which also depends on the walk-back distances between successive jobs. Our order spreads, therefore, only measure one part of the total walking distance, namely the picker's company of an engine from its first to the last SKU container. In Sect. 4.3, we simulate the total walking distances by randomly drawing production sequences in order to check whether our proxy is indeed a good surrogate for the pickers' total walking distances.
- As is the case at our engine manufacturer, we presuppose standardized SKU containers each having identical size. This allows us to calculate the objective value in number of passed containers. Many manufacturers aim to apply standardized containers, because this reduces their logistics and handling costs (Boysen et al. 2015). If, nonetheless, differently sized containers are applied, an order spread no longer solely depends on the first and last storage position, but also on the size of the containers placed in between. We rather aim at the most basic problem setting, which is also relevant for our real-world case. But adapting our models and solution approaches by this aspect seems truly straightforward. However, smaller containers are certainly another potential lever to reduce the pickers' walking effort, which comes at the price of more frequent replenishments. Evaluating this trade-off could be an interesting issue for future research.
- We restrict our view on ground storage of containers along the conveyor, because this is the most basic setting. At our engine manufacturer this setting is applied to allow for quick container exchanges without double handling for repacking items or maneuvering containers stockpiled on top of each other. The alterations required to adapt our setting to the two-dimensional case are straightforward, so that we abstain from a detailed description.
- We assume that the total in-line kitting area has enough space to house containers for all SKUs. If this is not given, depending on the parts demanded by subsequent workpieces, SKU containers must be added to and removed from stations on short notice. Given the short cycle times of the automotive industry these dynamic SKU swaps seem rather unrealistic, but they could be an interesting field for future research in other industries.
- Missing parts in the assembly process produce excessive costs, in the worst case, the line needs to be stopped with hundreds of workers being idle (Boysen et al.

2009). Therefore, our engine manufacturer takes great care that there are always enough parts available in each SKU container. Consequently, we can abstract from the replenishment process and assume that there are always enough pieces in a container to satisfy demand.

Now, we consider the computational complexity of WBSAP, which has the following complexity status.

**Theorem 1** *WBSAP is NP-hard in the strong sense, even if we have just a single kitting station  $|K| = 1$ .*

The proof is by transformation from the linear arrangement problem (LAP), which is well-known to be strongly NP-hard (Garey and Johnson 1979) and stated as follows:

**LAP:** Given a graph  $G = (V, E)$  and a positive integer  $M$ , is there a one-to-one-function  $f : V \rightarrow \{1, 2, \dots, |V|\}$ , i.e., a numbering of nodes  $V$  with integer values from 1 to  $|V|$ , such that  $\sum_{(u,v) \in E} |f(u) - f(v)| \leq M$ ?

**Proof** The transformation scheme for generating an instance of WBSAP from an LAP instance is as follows. Since we have only a single station  $|K| = 1$ , the complete workload is to be processed by a single worker, and the problem reduces to the storage assignment part seeking a sequence of SKUs within the station. For each node of LAP, we introduce a SKU, i.e.,  $|S| = |V|$ , and for each edge, we generate an engine and its associated picking order with unit weight  $f_i = 1$  exclusively demanding the two SKUs represented by the adjacent nodes. Thus, we have a direct mapping between SKUs and nodes, picking orders and edges as well as storage positions and node numbers, so that a one-to-one mapping between both problems is readily available.  $\square$

**Table 1** Notation for WBSAP-MIP

$I$	Set of engine types (index $i$ )
$S$	Set of SKUs with $S = \{1, \dots,  S \}$ (indices $s, s'$ )
$O_i$	Picking order of engine $i$ with $O_i \subset S$
$P$	Set of storage positions with $P = \{1, \dots,  P  =  S \}$ (indices $p, p'$ )
$K$	Set of kitting stations with $K = \{1, \dots,  K \}$ (index $k$ )
$f_i$	Production frequency of engine $i$
$\delta^{max}$	Continuous variable: maximum order spread
$\delta_{i,k}$	Continuous variable: spread of order $O_i$ in station $k$
$x_{s,p}$	Binary variable: 1, if SKU $s$ is assigned to storage position $p$ (0, otherwise)
$y_{p,k}$	Binary variable: 1, if position $p$ is assigned to kitting station $k$ (0, otherwise)
$z_{s,k}$	Binary variable: 1, if SKU $s$ is assigned to a storage position in kitting station $k$
$Z$	Objective function: maximum order spread

### 2.3 Mixed-integer model for WBSAP

Given the notation summarized in Table 1, we are able to formulate WBSAP as a MIP with objective function (3) and constraints (4) to (14).

#### WBSAP-MIP:

$$\text{Minimize } Z(\delta^{max}, \delta, x, y, z) = \delta^{max} \quad (3)$$

subject to

$$\sum_{p \in P} x_{s,p} = 1 \quad \forall s \in S \quad (4)$$

$$\sum_{s \in S} x_{s,p} = 1 \quad \forall p \in P \quad (5)$$

$$\sum_{k \in K} y_{p,k} = 1 \quad \forall p \in P \quad (6)$$

$$\sum_{p \in P} y_{p,k} \geq 1 \quad \forall k \in K \quad (7)$$

$$z_{s,k} \geq x_{s,p} + y_{p,k} - 1 \quad \forall s \in S; p \in P; k \in K \quad (8)$$

$$\begin{aligned} y_{p+1,k} &\geq y_{p,k} + y_{p',k} - 1 && \forall k \in K; \\ &p = 1, \dots, |P| - 2; \\ &p' = p + 2, \dots, |P| \end{aligned} \quad (9)$$

$$\begin{aligned} \delta_{i,k} &\geq \sum_{p \in P} x_{s,p} \cdot p - \sum_{p \in P} x_{s',p} \cdot p + 1 \\ &\quad - |P| \cdot (2 - z_{s,k} - z_{s',k}) \quad \forall i \in I; s, s' \in O_i; k \in K \end{aligned} \quad (10)$$

$$\delta^{max} \geq \sum_{i \in I} f_i \cdot \delta_{i,k} \quad \forall k \in K \quad (11)$$

$$x_{s,p} \in \{0, 1\} \quad \forall s \in S; p \in P \quad (12)$$

$$y_{p,k} \in \{0, 1\} \quad \forall p \in P; k \in K \quad (13)$$

$$z_{s,k} \in \{0, 1\} \quad \forall s \in S; k \in K \quad (14)$$

Objective function (3) minimizes the maximum weighted order spread over all kitting stations. Constraints (4) to (7) enable proper assignments of SKUs and storage positions: (4) assigns each SKU to exactly one storage position, (5) assigns exactly one SKU to each storage position, (6) assigns each storage position to exactly one kitting station, and (7) assigns at least one storage position to each kitting station. In inequalities (8), our  $z_{s,k}$ -variables are aligned with  $x_{s,p}$  and  $y_{p,k}$ . Note that in case SKU  $s$  is not assigned to kitting station  $k$ , the  $z_{s,k}$ -variables can attain either value 0 or 1, if this does not negatively affect solutions. (9) ensures connected kitting stations along the line. Specifically, if two positions  $p$  and  $p'$  are assigned to the same station, all other positions between  $p$  and  $p'$  must be assigned to this station as well. Constraints (10) define the order spread for each order and for each kitting station, and (11) defines the maximum order spread over all kitting stations, which is minimized in the objective function. Finally, variable domains are set in (12) to (14).

To further strengthen this formulation of WBSAP, we introduce several valid inequalities that eliminate symmetric solutions (so-called symmetry breakers):

$$\sum_{k \in K} k \cdot y_{p,k} \leq \sum_{k \in K} k \cdot y_{p+1,k} \quad \forall p = 1, \dots, |P| - 1 \quad (15)$$

$$\begin{aligned} y_{p-1,k-1} + y_{p,k} + y_{p'-1,k} + y_{p',k+1} + x_{s,p} + x_{s',p'} &\leq 5 \quad \forall k = 2, \dots, |K| - 1; \\ p &= 2, \dots, |P| - 1; \\ p' &= p + 1, \dots, |P|; \\ s, s' &\in S : s > s' \end{aligned} \quad (16)$$

$$\begin{aligned} y_{1,1} + y_{p-1,1} + y_{p,2} + x_{s,1} + x_{s',p} &\leq 4 \quad \forall p = 2, \dots, |P| - 1; \\ s, s' &\in S : s > s' \end{aligned} \quad (17)$$

$$\begin{aligned} y_{p-1,k-1} + y_{p,k} + x_{s,p} + x_{s',p'} + y_{p',k} + y_{p'+1,k+1} &\leq 5 \quad \forall k = 2, \dots, |K| - 1; \\ p &= 2, \dots, |P| - 2; \\ p' &= p + 1, \dots, |P| - 1; \\ s, s' &\in S : s > s' \end{aligned} \quad (18)$$

$$\begin{aligned} y_{1,1} + x_{s,1} + x_{s',p} + y_{p,1} + y_{p+1,2} &\leq 4 \quad \forall p = 2, \dots, |P| - 1; \\ s, s' &\in S : s > s' \end{aligned} \quad (19)$$

$$\begin{aligned} y_{p-1,|K|-1} + y_{p,|K|} + x_{s,p} + x_{s',|P|} + y_{|P|,|K|} &\leq 4 \quad \forall p = 2, \dots, |P| - 1; \\ s, s' &\in S : s > s' \end{aligned} \quad (20)$$

Constraints (15) sort the kitting stations along the line in increasing index order. In this way, solutions that only differ in the numbering of stations are eliminated. Note that we have also tried out the alternative formulation of these constraints of Ritt and Costa (2018), which can be borrowed from their model for assembly line balancing, on a small sample of instances. Since we were not able to detect any remarkable

performance differences, we abstain from reporting these tests in detail. Inequalities (16) and (17) specify the assignment of SKU sets to stations: The indexes of the first SKUs in each station are sorted in increasing order. Hence, solutions that only differ in the assignments of SKU sets to stations are eliminated. Finally, (18) to (20) eliminate solutions that only differ in the direction of the SKU sequence within a station, since this direction is not relevant for the order spread. The first SKU in each station must have a smaller index than the second SKU in that station. Note that constraints (15) to (20) are only necessary, if we have more than one kitting station, i.e.,  $|K| \geq 2$ . In case of  $|K| = 1$ , (21) can be applied to define the direction of the SKU sequence:

$$x_{s,1} + x_{s',|P|} \leq 1 \quad \forall s, s' \in S : s > s' \quad (21)$$

Preliminary computational tests have shown that all of these additional constraints, individually and in combination, reduce the solution time of a standard solver. However, the combination of all symmetry breakers performs best. Therefore, all further tests are conducted with symmetry breaking constraints (15) to (20) resp. constraints (21). Note that we also tried out another MIP based on variables  $x_{s,p,k}$ , which directly encode the assignment SKU  $s$  to storage position  $p$  in kitting station  $k$ . This MIP, however, was outperformed by the one elaborated above, so that we abstain from reporting further details here and refer to Appendix A instead.

### 3 Decomposition approaches for the holistic problem

In order to solve a complex optimization problem such as WBSAP, decomposition is often a promising approach. We will follow this general idea in this section by decomposing WBSAP into its two core subproblems, storage assignment and workload balancing. If we have a solution for one of the subproblems, we will find it much easier to solve the other one. However, we still have to decide on the order in which we solve the problems, as both approaches go hand in hand with advantages and disadvantages:

- **Balance first, sequence second (BFSS):** In this approach, we first decide on the assignment of SKUs to kitting stations. Given this assignment, we can determine the final SKU sequence for each station individually. The advantage of this approach is that the remaining sequencing problems can be small enough to solve them in reasonable time. However, this strongly depends on the amount and size of kitting stations since the remaining sequencing problem for a single station is still NP-hard. In Sect. 3.1, we introduce a meta-heuristic procedure for the workload balancing on the upper level and a beam search approach for the remaining storage assignment problem.
- **Sequence first, balance second (SFBS):** If we have a sequence of SKUs along the storage area given, the remaining workload balancing problem reduces to the decision on sizing and positioning of kitting stations, a problem solvable in polynomial time. The downside of this approach, however, is the bigger size of



the SKU sequence to be determined upfront. In Sect. 3.2, we provide a metaheuristic procedure for the storage assignment problem on the upper level and a dynamic programming procedure for the remaining workload balancing problem.

Each of these approaches is dedicated a separate section in the following.

### 3.1 Balance first, sequence second

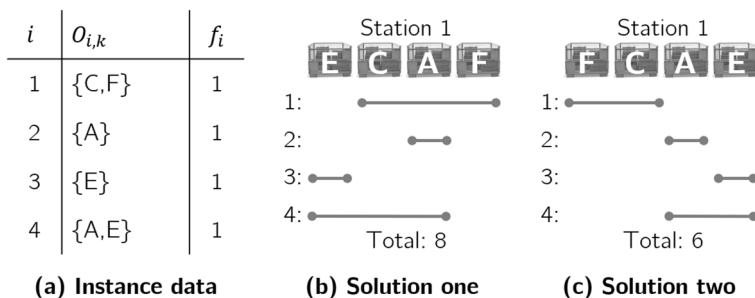
We explain this decomposition approach in reverted sequence. We start with the subsequent problem and elaborate how we determine a storage assignment problem for a given station workload. Then, we turn to the upper level and introduce a metaheuristic for determining the station workloads.

#### 3.1.1 Storage assignment for a given station workload

This section defines the storage assignment problem (SAP) for a single kitting station once the workload for this station is already given. Thus, we have a single kitting station  $k$  operated by a dedicated picker and the set  $S_k$  of SKUs to be placed within the station. The SKU containers have to be placed along the conveyor system, which has  $|S_k|$  storage positions, so that each SKU receives exactly one position. Such a storage assignment can, thus, be represented by a SKU sequence  $\phi_k$ , i.e., a permutation of SKU set  $S_k = \{1, 2, \dots, |S_k|\}$ . The order spread  $\delta_{i,k}^{SA}(\phi_k)$  of the picking order  $O_{i,k} \subset S_k$  associated with the SKUs of engine  $i \in I$  stored in station  $k$  is defined by:

$$\delta_{i,k}^{SA}(\phi_k) = \begin{cases} \max_{s \in O_{i,k}} \{\phi_k^{-1}(s)\} - \min_{s \in O_{i,k}} \{\phi_k^{-1}(s)\} + 1, & \text{if } O_{i,k} \neq \emptyset \\ 0, & \text{if } O_{i,k} = \emptyset \end{cases} \quad \forall i \in I, \quad (22)$$

where function  $\phi_k^{-1}(s)$  returns the storage position of SKU  $s$  within storage assignment  $\phi_k$ . By aggregating all order spreads, we receive the total weighted order spread within the station:



**Fig. 4** Example for SAP

$$Z_k^{SA}(\phi_k) = \sum_{i \in I} f_i \cdot \delta_{i,k}^{SA}(\phi_k). \quad (23)$$

*Example (cont.):* Consider the given workload of Station 1 within Solution three of Fig. 3d in Sect. 2.1. From the overall workload of six SKUs Station 1 only has to process SKU set  $S = \{A, C, E, F\}$  and the adapted order set depicted in Fig. 4a. Solution one and two depicted in Fig. 4b and c lead to a total order spread of 8 and 6, respectively.

Naturally, problem SAP can also be formulated as a MIP, which we present in Appendix B. In preliminary computational tests that are—for a matter of conciseness—not reported in this paper, however, the following approach proved much more successful. Recall that this problem has been shown to be strongly NP-hard within Theorem 1. Therefore, we introduce a dynamic programming (DP) scheme in the following, which is subsequently applied within a beam search heuristic. The following DP, which is an adaption of the basic sequencing DP of Held and Karp (1962), solves SAP to optimality.

Our DP consists of  $P_k + 1$  stages, each corresponding to a storage position in our line setup (plus a virtual starting stage). Each stage  $p$  includes states  $\bar{S}_k \subseteq S_k$ , each defining a subset of SKUs stored in the first  $p$  storage containers along the line. The initial state is represented by an empty set  $\bar{S}_{k,0} = \emptyset$ . The partial objective value of a state  $\bar{S}_k$  is denoted by  $z^{SA}(\bar{S}_k)$  and corresponds to the cumulative weighted order spread over all picking orders with respect to the first  $|\bar{S}_k|$  storage positions. Furthermore, we have a transition  $\bar{S}_k \rightarrow \bar{S}'_k$  from state  $\bar{S}_k$  to state  $\bar{S}'_k$  if  $\bar{S}_k \subset \bar{S}'_k$  and  $\bar{S}'_k \setminus \bar{S}_k = \{s\}$  with  $s \in S_k$ , that is, the successive state contains the same SKUs as its predecessor plus an additional SKU. The additional weighted order spread for such a transition amounts to

$$v(\bar{S}_k, \bar{S}'_k) = \sum_{i \in I} \begin{cases} f_i, & \text{if } O_{i,k} \setminus \bar{S}_k \neq \emptyset \wedge O_{i,k} \setminus \bar{S}'_k \neq O_{i,k} \\ 0, & \text{else} \end{cases}. \quad (24)$$

With the transition's contributions to the objective value on hand the Bellman recursion is defined by

$$z^{SA}(\bar{S}_k) = \min_{s \in \bar{S}_k} \{z(\bar{S}_k \setminus \{s\}) + v(\bar{S}_k \setminus \{s\}, \bar{S}_k)\} \quad (25)$$

with  $z(\emptyset) = 0$ . After a stage-wise forward recursion, the last stage  $P_k + 1$  contains only a single state  $\bar{S}_k = S_k$  including all SKUs with objective value  $Z_k^{SA} = z(S_k)$ . Finally, we can extract the optimal storage assignment by a simple backward recursion.

Regarding the computational effort of our DP, we have  $\mathcal{O}(2^{|S_k|})$  states and  $\mathcal{O}(|S_k| \cdot 2^{|S_k|})$  transitions. The implied exponential runtime of  $\mathcal{O}(|S_k| \cdot 2^{|S_k|})$  is in line with our complexity result for SAP.

Due to the exponential runtime (in the number of SKUs), DP struggles with larger instances of our storage assignment problem. To accelerate the procedure, we modify DP and introduce a heuristic beam search (BS) approach. BS

requires less runtime and memory as it only branches the  $\xi$  most promising (with respect to the partial objective value) states of each stage.  $\xi$  is called the beam width. With this modification, runtime and memory are polynomial bounded. However, optimality is not guaranteed hereby, since states leading to an optimal solution can be discarded during the forward recursion due to less promising partial objective values. Preliminary tests have shown that a beamwidth of  $\xi = 100$  performs well regarding solution quality and runtime for smaller and larger instances as well.

*Example (cont.):* Recall the example from above (see Fig. 4), where SKUs A, C, E, and F have to be sequenced within Station 1. The resulting beam search graph with a beamwidth of  $\xi = 2$  is depicted in Fig. 5. The procedure determines the four equally good—and in this case optimal—SKU sequences  $\langle C, F, A, E \rangle$ ,  $\langle C, F, E, A \rangle$ ,  $\langle F, C, E, A \rangle$  and  $\langle F, C, A, E \rangle$  with an objective value of 6, highlighted by bold transitions. In each stage, several states are evaluated regarding their partial objective value, e.g., in stage  $p = 1$ , the four states  $\{A\}$ ,  $\{C\}$ ,  $\{E\}$ , and  $\{F\}$  are evaluated. Due to the beamwidth, only the best two states are branched when constructing the next stage. Note that nodes that have been discarded due to the beamwidth are displayed with dashed outlines. In stage  $p = 2$ , several states have the same partial objective value, i.e.,  $\{A, C\}$ ,  $\{C, E\}$ ,  $\{A, F\}$  and  $\{E, F\}$ . Hence, we need to apply a tiebreaker to decide which states should be branched. In our case, we randomly choose a state ( $\{A, C\}$ ).

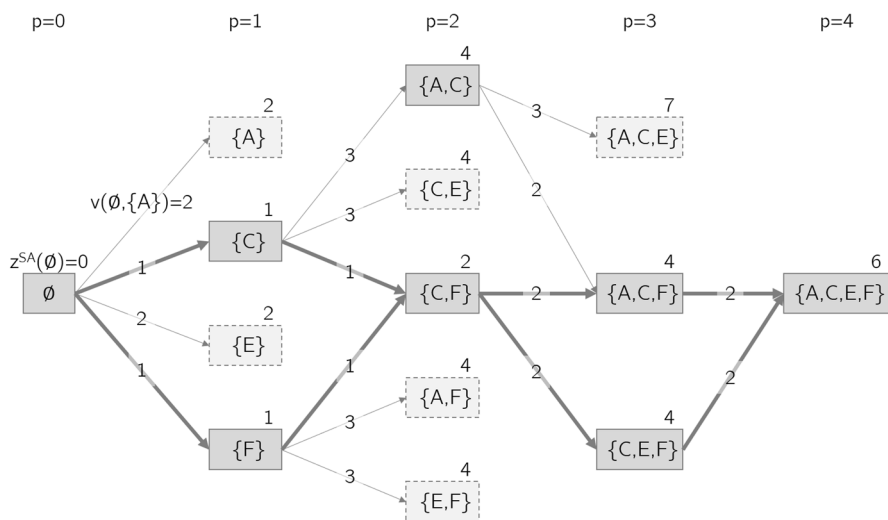


Fig. 5 BS graph for SAP

### 3.1.2 Workload balancing on the upper level

With the approach for solving the sequencing problem for a given kitting station  $k$  at hand, we can now tackle the balancing problem on the upper level. For this purpose, we implemented a straightforward simulated annealing (SA) approach to determine the assignment of SKUs to kitting stations. SA is a stochastic metaheuristic inspired by thermal processes for obtaining low-energy states in heat baths. Based on the probabilistic acceptance of neighboring solutions, SA is able overcome local optima (e.g., Kirkpatrick et al. 1983; Aarts et al. 1997).

Our SA operates on an array of sets  $(S_1, \dots, S_{|K|})$ , whereby set  $k$  contains all SKUs dedicated to station  $k$ . Hence, each SKU must be element of one of the sets, i.e.,  $\bigcup_{k \in K} S_k = S$ , each set must contain at least one SKU, i.e.,  $|S_k| \geq 1, \forall k \in K$ , and the sets must be disjoint, i.e.,  $S_k \cap S_{k'} = \emptyset, \forall k \neq k' \in K$ . The initial solution is determined by randomly assigning SKUs to sets, such that the above conditions are fulfilled. For obtaining a neighboring solution, we randomly perform either a swap move or a switch move:

- Swap: Randomly choose two SKUs from different stations and swap their assignment to sets.
- Switch: Move a randomly chosen SKU from one set to a different, also randomly chosen one.

Given the new (neighboring) assignment of SKUs to stations, we can obtain the weighted order spread  $Z_k$  for each station  $k$  by applying the BS approach introduced in the last section. The objective value of our (balancing) solution  $Z(S_1, \dots, S_k)$  can be determined by calculating the maximum weighted order spread over all stations  $Z(S_1, \dots, S_k) = \max_{k \in K} Z_k$ .

The decision whether or not a neighboring solution  $(S'_1, \dots, S'_k)$  should be accepted is made according to the following traditional probability scheme (see Aarts et al. 1997):

$$\begin{aligned} & \text{Prob}((S'_1, \dots, S'_k) \text{ replacing } (S_1, \dots, S_k)) \\ &= \begin{cases} 1, & \text{if } Z(S'_1, \dots, S'_k) \leq Z(S_1, \dots, S_k) \\ \exp\left(\frac{Z(S_1, \dots, S_k) - Z(S'_1, \dots, S'_k)}{T}\right), & \text{otherwise.} \end{cases} \end{aligned} \quad (26)$$

If accepted, our current solution  $(S_1, \dots, S_k)$  is replaced by neighbor  $(S'_1, \dots, S'_k)$  as the new starting point for further iterations.

We applied a simple static cooling schedule (see Kirkpatrick et al. 1983) for steering our SA. The initial value  $T^{init}$  for our control parameter  $T$ , the temperature, is given by  $T^{init} = Z((S_1, \dots, S_k)^{init})$ , where  $(S_1, \dots, S_k)^{init}$  denotes our randomly generated initial solution. For each value of  $T$ , we perform three iterations of constructing and comparing neighboring solutions. Afterward,  $T$  is decreased by multiplying it with the factor 0.999. We continue this procedure until the  $T \leq 0.001 \cdot T^{init}$ . If the given time limit for the procedure has not been reached, we restart SA with a new random solution and the initial temperature up to a maximum of five times.

Note that in our computational study, we invariably used control parameter values as described above as preliminary studies indicated that this parameter constellation outperforms other settings and delivers a reasonable compromise between solution quality and time.

## 3.2 Sequence first, balance second

The next decomposition approach solves the two subproblems in opposite order, which we again describe in reverted order. First, we elaborate the workload balancing problem for a given SKU sequence, and only afterward, we present a heuristic procedure to determine such a sequence.

### 3.2.1 Workload balancing for a given storage assignment

If the sequence of SKUs along the line  $\phi$  is given, the remaining workload balancing problem (WBP) simply decides on the range of storage positions covered by each kitting station. Hence, we look for a solution  $\psi^{WB} = (l_1, l_2, \dots, l_{|K|})$ , i.e., a sequence of storage positions with  $l_k$  defining the last storage position of station  $k \in K$ . Such a solution is feasible, if  $l_{k+1} \geq l_k + 1$  with  $l_0 = 0$ , that is each station contains at least one storage position. With the given SKU sequence  $\phi$ , we are now able to determine the order spread for each order in kitting station  $k$ :

$$\delta_{i,k}^{WB}(\psi) = \begin{cases} \max_{q = l_{k-1} + 1, \dots, l_k : \phi_q \in O_i} \{q\} - \min_{q = l_{k-1} + 1, \dots, l_k : \phi_q \in O_i} \{q\} + 1, & \text{if } \exists q \in \{l_{k-1} + 1, \dots, l_k\} : \phi_q \in O_i \\ 0, & \text{else,} \end{cases} \quad (27)$$

with  $l_0 = 0$ . The final objective value, i.e., the maximum of weighted order spreads across the kitting stations, can then be determined by

$$Z^{WB}(\psi) = \max_{k \in K} \left\{ \sum_{i \in I} f_i \cdot \delta_{i,k}^{WB}(\psi) \right\}. \quad (28)$$

*Example (cont.):* Recall the example from Fig. 3 in Sect. 2.1 and assume given SKU sequence  $\phi = \langle E, A, C, F, D, B \rangle$  and  $|K| = 3$  kitting stations. Figure 6 depicts three different solutions of WBP: (b) A poor arrangement of stations along the line can lead to very unequal workloads between the stations. (c) Even an equal partition of SKUs among kitting stations, where each station receives the same number of SKUs, does not guarantee an equal assignment of workload. (d) The optimal solution leads to a minimal workload of 5.

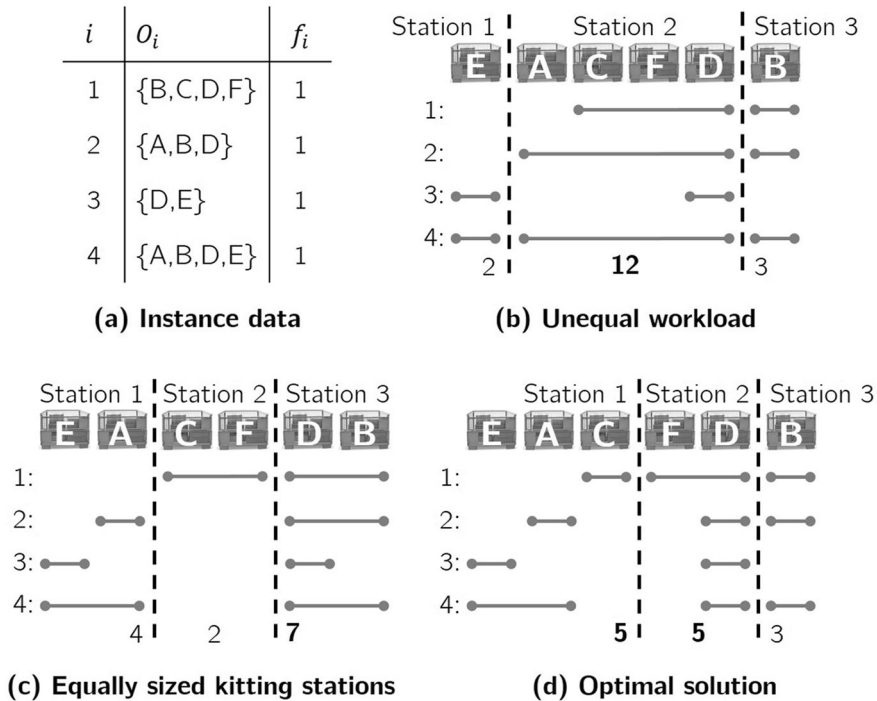


Fig. 6 Example for WBP

Naturally, WBP can also be modeled as a MIP. However, since this problem can be solved to optimality in polynomial time, we present such a MIP only in Appendix C. Instead, we elaborate a more efficient DP approach in the following.

This DP consists of  $|K| + 1$  stages, each corresponding to a kitting station (plus a virtual starting stage). Each stage  $k$  includes states  $(k, p)$  that each define the ending position  $p$  of kitting station  $k$ . The initial state is represented by  $(0, 0)$ . The partial objective value of a state  $(k, p)$  is denoted by  $z^{WB}(k, p)$  and corresponds to the maximum weighted order spread over the first  $k$  kitting stations along the line. Furthermore, we have a transition  $(k, p) \rightarrow (k + 1, p')$  from state  $(k, p)$  to state  $(k + 1, p')$ , if  $p < p'$ , that is, the ending position of a station is smaller than the ending position of the subsequent kitting station, and  $p' \leq |P| - |K| + k + 1$ , that is, the remaining number of positions is not smaller than the remaining number of kitting stations. The order spread for the new station added by such a transition is given by  $w_{p+1,p'}$ . With the transition's contributions to the objective value on hand the Bellman recursion is defined by

$$z^{WB}(k, p') = \min_{k-1 \leq p < p'} \{ \max \{ z(k-1, p), w(p+1, p') \} \} \quad (29)$$

with  $z^{WB}(0,0) = 0$ . After a stage-wise forward recursion, the last stage  $|K|$  contains only a single state  $(|K|, |P|)$  with objective value  $Z = z^{WB}(|K|, |P|)$ . Finally, we can extract the optimal workload balancing by a simple backward recursion.

Regarding the computational effort of our DP, we have  $\mathcal{O}(|K|)$  stages each containing  $\mathcal{O}(|P|)$  states. Since the number of transitions leaving a state is in  $\mathcal{O}(|P|)$  and  $|P| = |S|$ , the resulting runtime is in  $\mathcal{O}(|K| \cdot |S|^2)$ , thus polynomial. However, in order to apply the DP procedure, the values  $w_{p,p'}$  have to be determined beforehand, which requires a runtime in  $\mathcal{O}(|I| \cdot |S|^3)$ . Since  $|K| \leq |S|$ , the total runtime is in  $\mathcal{O}(|I| \cdot |S|^3)$ , thus polynomial. This allows us to solve even larger sized instances in a short runtime.

*Example:* Recall the example above depicted in Fig. 6 with the given SKU sequence  $\phi = \langle E, A, C, F, D, B \rangle$  and  $|K| = 3$  kitting stations. The aggregated weighted order spreads are given by

$$w = \begin{pmatrix} 2 & 4 & 5 & 6 & 17 & 20 \\ - & 2 & 3 & 4 & 12 & 15 \\ - & - & 1 & 2 & 6 & 9 \\ - & - & - & 1 & 5 & 8 \\ - & - & - & - & 4 & 7 \\ - & - & - & - & - & 3 \end{pmatrix}$$

Applying the introduced DP procedure, we obtain the graph depicted in Fig. 7. The optimal solution with an objective value of 5 is highlighted by the bold transitions. Here, the first kitting station includes the storage positions 1, 2, and 3 (state  $(1, 3)$ ) with a workload of  $w_{1,3} = 5$ , the second station includes positions 4 and 5 (state  $(2, 5)$ ) with a workload of  $w_{4,5} = 5$  and the last (third) station includes only position

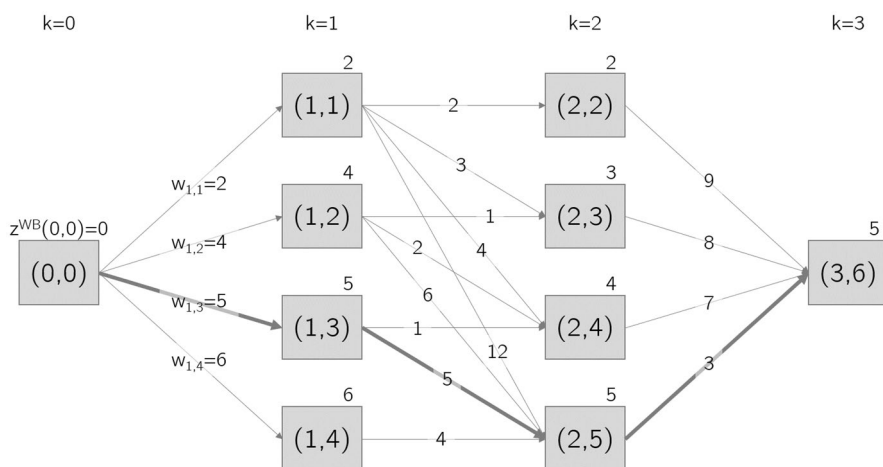


Fig. 7 DP graph for WBP

6 (state (3, 6)) with a workload of  $w_{6,6} = 3$ . The maximum workload over all three stations is  $\max\{5, 5, 3\} = 5$ .

### 3.2.2 Storage assignment on the upper level

In the first stage of our sequence first, balance second approach, we address the sequencing problem, i.e., the ordering the SKUs along the kitting area. Again, we implemented a straightforward simulated annealing heuristic, which operates on a permutation  $(s_1, s_2, \dots, s_{|P|})$  of SKU set  $|S|$ , with  $s_p$  defining the SKU stored at storage position  $p$ . To obtain an initial solution, we simply assign SKUs randomly. A neighboring solution  $(s'_1, \dots, s'_{|P|})$  is then determined by randomly performing one of the following moves:

- Swap: Two (different) SKUs are randomly chosen and swap their sequence positions.
- Switch: Relocate a random SKU to a (randomly selected, different) position within the storage assignment.

With the new (neighboring) sequence on hand, we can determine the weighted order spread  $Z(s'_1, \dots, s'_{|P|})$  of our (sequencing) solution by applying the DP approach introduced in the previous section. The probability for accepting the neighboring solution—and therefore replacing the former solution—is given by

$$\begin{aligned} & \text{Prob}((s'_1, \dots, s'_{|P|}) \text{ replacing } (s_1, \dots, s_{|P|})) \\ &= \begin{cases} 1, & \text{if } Z(s'_1, \dots, s'_{|P|}) \leq Z(s_1, \dots, s_{|P|}) \\ \exp\left(\frac{Z(s_1, \dots, s_{|P|}) - Z(s'_1, \dots, s'_{|P|})}{T}\right), & \text{otherwise.} \end{cases} \end{aligned} \quad (30)$$

**Table 2** Parameters for instance generation

Symbol	Description	Values			
		Small	Large	Surrogate	Managerial (Default)
$ S $	Number of SKUs	5,7,...,15	25,50,75	10,20,...,50	10,20,...,100 (50)
$ I $	Number of engine types	5,10,...,25	5,15,25	5,10,...,25	2,4,...,20 (10)
$ K $	Number of kitting stations	1,2,...,5	1,2,...,5	1,2,...,5	1,2,...,10 (5)
$r$	SKU ratio of reference order	0.5	0.5	0.5	0.1,0.2,...,0.9 (0.5)
$\alpha$	Heterogeneity of frequencies	0.5	0.5	0.5	0.1,0.2,...,0.9 (0.5)
$\beta$	Heterogeneity of order set	1	1	1	0.1,0.2,...,0.9 (0.5)



If accepted, the neighboring solution  $(s'_1, \dots, s'_{|P|})$  replaces  $(s_1, s_2, \dots, s_{|P|})$  and serves as the new starting point for further iterations. We apply exactly the same static cooling, restart, and stop policy as in our previous SA of Sect. 3.1.2.

## 4 Computational study

First, we evaluate the computational performance of our different solution methods in Sect. 4.2. Then, we answer the question whether our surrogate objective (i.e., the order spread) is indeed a suitable proxy for the workers' actual total walking distances (see Sect. 4.3). Furthermore, we address several managerial aspects and analyze the benefit of our optimization approach in Sect. 4.4. Beforehand, however, we have to elaborate on our data instances in Sect. 4.1.

### 4.1 Instance generation

Since there is no established testbed for our WBSAP and confidentiality issues prohibit the application of real-world data, we had to generate our own instance generator. To do so, Table 2 lists the main parameters that are handed over to our generator as its own input data.

Table 2 lists the parameter values for four different sets of instances employed during the subsequent sections. For a given combination of parameter values, each single instance is generated as follows: For given values of  $|S|$ ,  $|I|$ , and  $|K|$ , we construct a set of different orders that refer to different engine types to be produced. For being able to generate order sets that are either more homogeneous or rather heterogeneous, we first derive a reference order with  $\max(1, rd(|S| \cdot r))$  randomly chosen SKUs, whereby  $rd(x)$  yields the closest integer to  $x$ . Subsequently, we construct  $|I|$  orders  $O_i$ , whereby each SKU (not) contained in the reference order is also demanded by an order  $O_i$  with a probability of  $1 - \frac{\beta}{2} (\frac{\beta}{2})$ . Smaller (larger) values of  $\beta$ , with  $\beta \in (0, 1]$ , thus lead to rather homogeneous (heterogeneous) orders. As each engine type demands an individual set of parts, we make sure that all orders are different. To ensure this,  $|I|$  must be smaller or equal to  $2^{|S|} - |S| - 2$ . If the constructed order is already contained in our order set, we dispose it and restart order generation. We also make sure that each order contains at least one SKU and that each SKU is used at least once. We, then, determine the  $|I|$  order frequencies randomly. Hereby, we presuppose a 7-hour working shift and a cycle time of 90 s, which equals the situation at our engine producer. Therefore, during a shift, about 280 engines are produced. We determine order frequencies  $\hat{f}_i \in (1 - \alpha, 1 + \alpha)$  and normalize them, so that  $\sum_{i \in I} \hat{f}_i \approx 280$ . Hence, we set

$$f_i = rd\left(280 \cdot \frac{\hat{f}_i}{\sum_{i \in I} \hat{f}_i}\right). \quad (31)$$

Analogously to our order set, smaller (larger) values of  $\alpha$ , with  $\alpha \in [0, 1)$ , lead to rather homogeneous (heterogeneous) frequencies. Note that due to rounding the total number of engines may differ slightly from 280.

For the first three instance sets (Small, Large, and Surrogate), we repeat the construction process five times for each parameter combination, resulting in sets of 750, 225, and 625 instances, respectively. For the fourth set (Managerial), we only vary one parameter at a time and fix the others to their default values listed in parentheses within Table 2. We repeat the generation process 25 times for each parameter combination and obtain 1425 instances in this data set.

## 4.2 Computational performance

To evaluate our solution methods for WBSAP, we benchmark their computational performance for differently sized sets of instances: Small instances are still manageable by a commercial solver (to be solved to proven optimality in acceptable time) and large instances represent real-world problem sizes (see Table 2, Columns 'Small' and 'Large').

We solved the 750 small and 225 large instances each with both the balance first, sequence second (BFSS) algorithm and the sequence first, balance second (SFBS) procedure. Our industry partner currently determines a random SKU assignment to storage positions and assigns all stations the same amount of SKUs. We dub this approach *RND*. All procedures were implemented in Visual Basic, and tests were performed on a 64-bit system on an Intel(R) Core(TM)2 Quad CPU with 2.83 gigahertz and 8 gigabytes memory. Moreover, we apply optimization software 'Gurobi' (GRB, version 9, see Gurobi Optimization, LLC 2021) with a time limit of 300 s (900 s) for small (large) instances. The criteria with which we evaluate computational performance is defined at the bottom of Tables 3 and 4. The results summarized in these tables suggest the following findings.

- *Gurobi*: Among the 750 small instances, Gurobi solves 443 (59%) to proven optimality within the given time frame of 300 s. However, even for the small instances, Gurobi regularly hits the timeout when more stations  $|K|$  and SKUs  $|S|$  are involved. On average over all small instances, Gurobi is clearly outperformed by our decomposition approaches both with regard to solution quality and runtime. This finding becomes even more pronounced for the large instances. Here, Gurobi does not terminate prior to the timeout of 900 s in any instance, and for 109 out of 225 instances (48.4%), Gurobi is not even able to find a feasible solution during this time. Hence, we conclude that Gurobi is not suitable for instances of real-world size, and heuristic solutions seem preferable instead.
- *BFSS* vs. *SFBS*: We can observe no clear dominance of either of our decomposition approaches. In some cases, BFSS delivers better results than SFBS; in others, it is the other way round. This holds true for both runtime and solution quality. On average, however, SFBS leads to better results regarding both test sets, small and large, and both performance criteria, runtime and solution quality. Thus, SFBS seems the better option.

**Table 3** Performance results of Gurobi (GRB), random approach Rseq, balance first, sequence second (BFSS), and sequence first, balance second (SFBS) on small instances

IK1	S	#opt		gapOpt [%]		#best		gapBest [%]				cpu time [s]			
		GRB	GRB	GRB	GRB	GRB	GRB	RND	BFSS	SFBS		GRB	RND	BFSS	SFBS
1	5	25	<0.01	25	1	25	<0.01	12.87	<0.01	<0.01		0.03	<0.01	<0.01	0.31
1	7	25	<0.01	25	0	25	<0.01	20.42	<0.01	<0.01		0.13	<0.01	<0.01	0.44
1	9	25	<0.01	25	0	25	<0.01	25.01	<0.01	<0.01		1.27	<0.01	0.02	0.56
1	11	25	<0.01	25	0	22	<0.01	28.59	0.26	0.05		9.63	<0.01	0.05	0.68
1	13	19	0.03	23	0	17	0.03	31.42	0.36	0.03		136.23	<0.01	0.07	0.81
1	15	8	0.08	16	0	10	0.14	25.70	0.89	0.01		251.78	<0.01	0.11	0.97
2	5	25	<0.01	25	1	25	<0.01	20.33	<0.01	<0.01		0.11	<0.01	15.11	6.29
2	7	25	<0.01	25	0	25	<0.01	28.40	<0.01	<0.01		0.97	<0.01	41.67	10.26
2	9	25	<0.01	25	0	25	<0.01	28.24	<0.01	<0.01		20.11	<0.01	120.73	15.47
2	11	9	0.23	14	0	23	0.48	34.54	0.10	0.16		248.00	<0.01	294.25	22.28
2	13	0	0.63	7	0	19	1.90	36.95	0.48	0.48		300.02	<0.01	299.53	30.94
2	15	0	0.75	1	0	17	3.09	40.28	0.88	0.59		300.03	<0.01	299.54	40.81
3	5	25	<0.01	25	4	25	<0.01	12.51	<0.01	<0.01		0.14	<0.01	9.06	8.03
3	7	25	<0.01	25	0	25	<0.01	35.97	<0.01	<0.01		2.52	<0.01	19.15	15.63
3	9	25	<0.01	25	0	25	<0.01	19.70	<0.01	<0.01		51.78	<0.01	38.99	26.10
3	11	5	0.46	10	0	25	0.92	37.36	<0.01	0.14		280.39	<0.01	90.30	39.70
3	13	0	0.70	6	0	23	3.06	43.33	0.02	0.58		300.04	<0.01	201.92	55.89
3	15	0	0.78	0	0	4	2.83	35.68	3.26	0.07		300.06	<0.01	299.31	77.08
4	5	25	<0.01	25	3	25	<0.01	21.26	<0.01	<0.01		0.08	<0.01	7.42	7.34
4	7	25	<0.01	25	0	25	<0.01	23.60	<0.01	<0.01		2.73	<0.01	13.22	17.13
4	9	25	<0.01	25	0	25	<0.01	40.21	<0.01	<0.01		57.51	<0.01	23.80	31.30
4	11	2	0.47	12	0	24	0.81	28.61	0.03	<0.01		293.64	<0.01	43.17	50.30

**Table 3** (continued)

K	S	#opt		gapOpt [%]		#best		gapBest [%]				cpu time [s]			
		GRB	GRB	GRB	GRB	GRB	GRB	GRB	RND	BFSS	SFBS	GRB	RND	BFSS	SFBS
4	13	0	0.70	6	0	23	18	2.45	40.46	0.09	0.24	300.06	<0.01	82.93	74.27
4	15	0	0.84	0	0	22	10	7.12	39.20	0.07	0.56	300.06	<0.01	161.97	104.05
5	5	25	<0.01	25	25	25	25	<0.01	<0.01	<0.01	<0.01	0.01	<0.01	<0.01	<0.01
5	7	25	<0.01	25	1	25	25	<0.01	27.77	<0.01	<0.01	2.11	<0.01	11.32	15.73
5	9	25	<0.01	25	0	25	25	<0.01	24.58	<0.01	<0.01	46.64	<0.01	17.66	32.09
5	11	0	0.41	15	0	25	25	1.48	42.24	<0.01	<0.01	300.04	<0.01	29.11	54.79
5	13	0	0.67	2	0	24	24	3.20	42.38	0.05	0.01	300.06	<0.01	48.85	84.50
5	15	0	0.81	0	0	22	17	11.67	26.70	0.05	0.23	300.09	<0.01	83.60	121.67
avg/total		443	0.25	512	35	675	663	1.31	29.14	0.22	0.11	136.87	<0.01	75.10	31.51

*#opt*: Number of instances (out of 25) solved to optimality, *gapOpt*: Optimality gap, *#best*: Number of instances (out of 25)

where the method performed best compared to others, *gapBest*: Objective gap to best performing procedure

**Table 4** Performance results of Gurobi (GRB), random approach Rseq, balance first, sequence second (BFSS), and sequence first, balance second (SFBS) on large instances

IKI	S	#noSol		#best		gapBest [%]						cpu time [s]			
		GRB	GRB	GRB	GRB	RND	BFSS	SFBS	GRB	RND	BFSS	RND	GRB	RND	SFBS
1	25	0	2	0	2	0	2	12	2.40	27.49	2.85	0.19	900.03	0	0.58
1	50	0	1	0	6	0	6	8	5.45	26.46	4.05	0.57	900.06	0	6.16
1	75	0	0	0	11	0	11	4	7.88	21.88	2.53	2.27	900.32	0	25.80
2	25	0	0	0	4	0	4	11	3.91	43.10	0.90	0.28	900.07	0	899.86
2	50	0	0	0	8	0	8	7	8.89	34.13	2.91	4.12	900.36	0	903.32
2	75	0	1	0	9	0	9	5	14.45	28.27	2.59	6.59	900.56	0	907.10
3	25	0	0	0	2	0	2	14	23.54	44.92	3.18	0.06	900.25	0	899.57
3	50	14	0	0	7	0	7	8	56.93	31.19	3.44	4.47	901.06	0	901.21
3	75	15	0	0	12	0	12	3	—	22.23	0.19	12.79	900.82	0	903.54
4	25	5	0	0	3	0	3	12	140.69	47.81	8.44	0.20	900.29	0	899.58
4	50	15	0	0	9	0	9	6	—	33.62	3.05	3.14	902.34	0	900.53
4	75	15	0	0	7	0	7	8	—	21.76	4.00	5.73	901.93	0	903.07
5	25	15	0	0	5	0	5	11	—	41.09	10.52	0.12	900.15	0	847.87
5	50	15	0	0	5	0	5	10	—	22.59	3.70	0.84	902.53	0	900.30
5	75	15	0	0	7	0	7	8	—	15.89	5.76	4.29	902.90	0	902.02
avg/total		109	4	0	97	0	97	127	21.22	30.83	3.87	3.05	900.91	0	720.03

*#noSol*: Number of instances (out of 15) without feasible solution within time limit, *#best*: Number of instances (out of 15)

where method performed best compared to others, *gapBest*: Objective gap to best performing procedure

- *Real-world method:* Finally, we can observe that the status quo method applied by our engine producer seems not advisable. Although its runtime requirement is barely measurable, it produces gaps to the best solutions of up to 47.81%. How this translates into walking effort for the pickers is investigated in more detail in Sect. 4.4. Note that a single random SKU sequence is a rather weak competitor from an optimization perspective. In Appendix D, we explore the performance impact if more than a single random solution are drawn. Our results show that especially for large instances such a simple approach cannot compete with our more sophisticated optimization approaches.

To conclude, our results suggest that both our decomposition approaches BFSS and SFBS deliver good solutions in a short amount of time, so that they seem well-suited even if large instances of real-world size are to be solved. SFBS is a bit faster and delivers better average objective values, so that for all further tests SFBS is our method of choice.

### 4.3 On the appropriateness of the surrogate objective

In this section, we aim to answer the question whether our surrogate objective, namely, minimizing the maximum weighted order spread, is indeed a good proxy for the actual objective, which is to minimize the workers' actual maximum walking distance. Recall that when planning the storage assignment, which is done on a mid-term basis (e.g., twice a year), we do not know the actual daily production sequences of engines. Instead, we only have reliable forecasts on the frequencies, in which each engine type is produced. Note that our engine producer has long-term contracts with car manufactures and rather long lead times, so that aggregate demands for the next months are indeed well predictable in this case. This allows us to quantify the distances each worker has to accompany the engines over the planning horizon with the help of our surrogate objective. However, we cannot add the walk-back distances between subsequent engines, because they depend on the production sequences. In this section, we investigate if our proxy, in spite of this inaccuracy, still supports our 'true' objective.

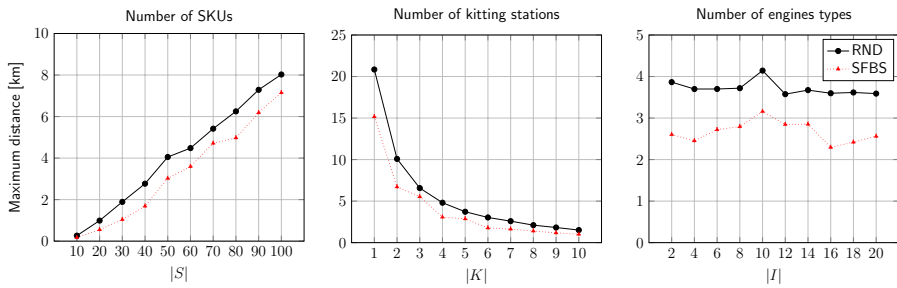
To do so, we evaluate 625 instances, including 25 different settings of station numbers  $|K| \in \{1, 2, \dots, 5\}$  and SKU numbers  $|S| \in \{10, 20, \dots, 50\}$ , each generated with the data generator defined in Sect. 4.1, applying the parameter settings stated in Column 'Surrogate' of Table 2. For each of these instances, 100 random WBSAP solutions, each consisting of a random storage assignment of SKUs along the line and random station borders, are obtained. Each solution is evaluated with our surrogate objective to determine the resulting maximum order spread over all kitting stations. These values are compared to the actual maximum walking distances that include the walk-back distances. To obtain them, we draw 250 production sequences per instance, where each engine type occurs in this sequence according to the given production frequencies. For a given production sequence, the actual total walking distance per station and thus the maximum over all stations can easily be obtained. Finally, we average the maximum actual walking distances over all 250 production

**Table 5** Correlation coefficients (Pearson) between surrogate and actual objective, average surrogate objective value, and average total walking distance (simulated) for different instance sizes

K	S	Pearson			Objective	Total
		Minimum	Average	Maximum	Value	Distance [m]
1	10	0.995203	0.999220	1.000000	2263	3170
1	20	0.999999	1.000000	1.000000	5032	7589
1	30	1.000000	1.000000	1.000000	7863	12111
1	40	1.000000	1.000000	1.000000	10635	16539
1	50	1.000000	1.000000	1.000000	13426	20996
2	10	0.998740	0.999720	0.999996	1515	1987
2	20	0.999956	0.999997	1.000000	3558	5236
2	30	1.000000	1.000000	1.000000	5677	8619
2	40	1.000000	1.000000	1.000000	7761	11949
2	50	1.000000	1.000000	1.000000	9868	15314
3	10	0.995105	0.998915	0.999723	1091	1326
3	20	0.999814	0.999964	0.999999	2677	3831
3	30	0.999999	1.000000	1.000000	4440	6645
3	40	1.000000	1.000000	1.000000	6183	9429
3	50	1.000000	1.000000	1.000000	7824	12049
4	10	0.995290	0.998249	0.999226	844	947
4	20	0.999776	0.999952	0.999999	2235	3127
4	30	0.999977	0.999998	1.000000	3672	5418
4	40	1.000000	1.000000	1.000000	5133	7751
4	50	1.000000	1.000000	1.000000	6578	10059
5	10	0.979236	0.994913	0.999257	640	644
5	20	0.999697	0.999884	0.999975	1829	2483
5	30	0.999985	0.999995	1.000000	3106	4515
5	40	0.999996	1.000000	1.000000	4342	6489
5	50	0.999999	1.000000	1.000000	5637	8556
over all instances		0.979236	0.999632	1.000000	4953	7471

sequences. For the resulting  $2 \cdot 100$  values, namely of our surrogates and the actual objectives, we calculate Pearson's correlation coefficient. The minimum, average and maximum correlation coefficients, rounded to six digits, are listed in Table 5 for all 25  $|S|$ - $|K|$ -parameter settings. To get an idea on the absolute values of our surrogate objective and the (simulated) total walking distance (in m), we also present the respective average values in the table as well. Note that the actual forward distance (in m) can be determined by  $0.8 \cdot (\text{objective value})$  as SKUs have an assumed width of 80 cm.

For every parameter combination, we observe correlation coefficients close to 1. This result verifies that we have a strong positive correlation between our surrogate



**Fig. 8** Maximum walking distances of status-quo method RND and our decomposition approach SFBS depending on the number  $|S|$  of SKUs (left), the number  $|K|$  of stations (middle), and the number  $|I|$  of engine types (right)

'maximum weighted order spread' and actual objective 'maximum actual walking distance.' We can thus conclude that our proxy seems well-suited and, if applied to guide an optimization task, can successfully support the reduction in the workers' actual walking distances.

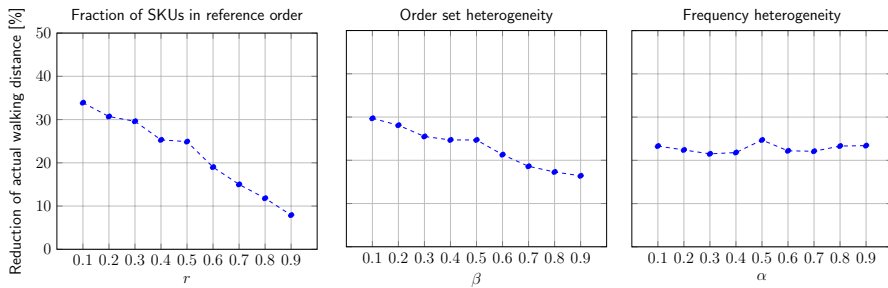
#### 4.4 Managerial issues: reduction in walking effort

In this section, we evaluate whether our optimization task, i.e., optimizing the storage assignment and division of labor among kitting stations, actually has the potential to considerably reduce the workers' walking effort. To explore this, our computational experiment is designed as follows. We start with the 1425 instances generated with our instance generator (see Sect. 4.1) and the parameter settings from Column 'Managerial' in Table 2, and derive WBSAP solutions with the status quo method RND of our engine producer (i.e., random storage assignment with equally sized stations) and our best-performing decomposition approach SFBS for each instance. To obtain the actual maximum walking distances for each of these solutions that also include the walk-back distances, we again derive 250 random production sequences of 7-hour shifts with engine occurrences according to the given frequencies. The resulting actual maximum walking distances are finally averaged over all 250 production sequences.

First, we take a look on the impact of parameters  $|S|$  (number of SKUs),  $|K|$  (number of kitting stations), and  $|I|$  (number of engine types to be produced) when benchmarking our two competitors. In Fig. 8, we report the actual maximum walking distances of our two solution methods depending on the above three parameters. These results suggest the following findings:

- *Impact of SKUs:* First, we can confirm an expectable result in Fig. 8 (left). If the engines require more parts and  $|S|$  increases, then more SKU containers have to be arranged along the line. Obviously, this leads to a linear increase in the walking distances.





**Fig. 9** Reduction in the actual maximum walking distance of our optimization approach SFBS compared to the status quo method of our engine producer depending on fraction  $r$  of SKUs that belong to the reference order (left), order set heterogeneity  $\beta$  (middle), and frequency heterogeneity  $\alpha$  (right)

- *Impact of stations:* Figure 8 (middle) displays the impact of another lever to reduce the walking effort of the workers: additional kitting stations. This, however, also increases the kitting workforce and thus wage costs. We can observe diminishing returns of additional workers. When having a small workforce every additional worker leads to a considerable relieve. If however the workforce is already substantial, the extra relieve of yet another worker is small.
- *Impact of engine types:* Finally, Fig. 8 (right) shows that the number  $|I|$  of different engine types that are produced on the line has only moderate impact. It is not the sheer number of engines but rather the level of heterogeneity of their part demands that impacts the walking effort, which can be seen in our further analysis.

Our previous results of Fig. 8 have shown a consistent improvement of our optimization approach over status quo method RND. We analyze these gains in more detail with the help of Fig. 9. Here, we display the percental reduction of the actual maximum walking distances of our optimization approach SFBS compared to the baseline of status-quo method RND of our engine producer. The results of Fig. 9 suggest the following findings:

- *Impact of reference order:* During the discussions with our practice partner, we had to learn that in engine production there are quite a few basic parts that are required by most engine types. The reference order represents these parts. A larger  $r$ -value (with default  $\beta = 0.5$ ) mainly results in larger order sets per engine (see Fig. 9 (left)). In this case, large parts of each kitting station must be passed in each cycle anyway, so that optimization cannot gain much. If only a few SKUs are needed (i.e., at a smaller  $r$ -value), optimization has much more flexibility to cut the accompanying walks for specific engines short.
- *Impact order heterogeneity:* Parameter  $\beta$  controls the probability with which a SKU of the reference set is contained in a specific order (see Fig. 9 (middle)). Thus, a small (large)  $\beta$  leads to more (fewer) SKUs of the reference set being contained in each order. Thus, smaller  $\beta$ -values lead to rather homogeneous SKU demands of all engine types. Hence, we benefit substantially by optimiz-

ing the arrangement of the most common SKUs. Optimization can build clusters of SKUs jointly demanded in similar orders and the workers must only traverse subsections of their stations for each engine. In this case, the walking effort of pickers can considerably be reduced by up to 30%. For more heterogeneous orders, it is harder to find SKU arrangements that lead to small order spread for each order, and therefore, the benefit of optimization decreases with increasing values of  $\beta$ .

- *Impact of frequency:* In Fig. 9 (right), we can observe that more or less variation in the frequencies, in which the engine types are produced (i.e., regulated by parameter  $\alpha$ ), has only negligible impact. The improvements of optimization over the status quo are consistently over 20 %. This result indicates that when applying the default values for all other parameters, this obviously leads to rather homogeneous demands for parts among different engine types. Hence, varying their frequencies still leads to similar part demands.

To conclude, our study reveals the following main take-home message. On average over our instances, sophisticated optimization can reduce the maximum walking distances of pickers by more than 20 %, which is good news for both workers and their employers. For workers, it makes a considerable difference, especially on their long-term well-being, if their daily walking distances reduce by several kilometers. Healthy and more satisfied workers are certainly also important for their employers, especially in the aging societies of many industrialized countries. Furthermore, these gains can be realized without more wage costs for additional workers or a reduction in the output; merely a low-cost rearrangement of SKU containers along the line and altered station borders are required.

## 5 Conclusions and outlook

In this paper, we introduce a novel part feeding policy to the scientific literature: in-line kitting. The first part of an assembly line is reserved to part logistics. Here, pickers have to add the requested parts that are arranged in containers along the line to traveling part kits that move with their workpieces on the conveyor. To reduce the ergonomic effort for these pickers when having to walk along their kitting stations, we formulate a novel optimization problem that arranges the SKU containers along the line and determines the borders of kitting stations. We introduce different solution methods, and our computational performance tests identify decomposition approach sequence first, balance second (SFBS) as an appropriate approach even if large instances of real-world size must be solved. In further tests on managerial issues, we show that applying this method can reduce the walking distances of pickers by about 20 % without producing any additional costs or reducing the output.

Future research could challenge our solution methods and provide even more powerful (especially exact) optimization approaches. Furthermore, it could be beneficial to provide the same SKU not only at a single kitting station, but also at multiple ones. This adds an operational selection problem, from which station a specific SKU demand should be satisfied, and increases the space demand for kitting. On

**Table 6** Notation for WBSAP-MIP2

$I$	Set of engine types (index $i$ )
$S$	Set of SKUs with $S = \{1, \dots,  S \}$ (indices $s, s'$ )
$O_i$	Picking order of engine $i$ with $O_i \subset S$
$P$	Set of storage positions with $P = \{1, \dots,  P  =  S \}$ (indices $p, p'$ )
$K$	Set of kitting stations with $K = \{1, \dots,  K \}$ (index $k$ )
$f_i$	Production frequency of engine $i$
$\delta^{max}$	Continuous variable: maximum order spread over all kitting stations
$\delta_{i,k}$	Continuous variable: spread of order $O_i$ in station $k$
$x_{s,p,k}$	Binary variable: 1, if SKU $s$ is assigned to storage position $p$ in kitting station $k$ (0, otherwise)
$Z$	Objective function: maximum order spread

the other hand, duplicate SKUs could—if properly placed—further reduce the walking effort of the workers. Furthermore, reducing the walking effort could not only lower the physical effort of the existing workforce. These savings could also be used to improve productivity and reduce the kitting workforce. The resulting balancing problem that minimizes the workforce for a given cycle time is an interesting challenge for future research. Finally, future research should evaluate whether in-line kitting could also be a suitable alternative for other production lines (beyond engine production). This suggests an extensive benchmark study of all alternative part feeding policies in different production environments.

## Appendix A: An alternative MIP for WBSAP

This appendix presents an alternative MIP model, which due to its three-dimensional assignment variables seems better accessible but turned out non-competitive. Applying the notation summarized in Table 6, this MIP (dubbed WBSP-MIP2) consists of objective function (32) subject to constraints (33) to (46):

### WBSAP-MIP2:

$$\text{Minimize } Z(\delta^{max}, \delta, x) = \delta^{max} \quad (32)$$

subject to

$$\sum_{p \in P} \sum_{k \in K} x_{s,p,k} = 1 \quad \forall s \in S \quad (33)$$

$$\sum_{s \in S} \sum_{k \in K} x_{s,p,k} = 1 \quad \forall p \in P \quad (34)$$

$$\sum_{s \in S} \sum_{p \in P} x_{s,p,k} \geq 1 \quad \forall k \in K \quad (35)$$

$$\sum_{s \in S} x_{s,p+1,k} \geq \sum_{s \in S} x_{s,p,k} + \sum_{s \in S} x_{s,p',k} - 1 \quad \forall k \in K; p = 1, \dots, |P| - 2; \quad (36)$$

$$p' = p + 2, \dots, |P|$$

$$\delta_{i,k} \geq \sum_{p \in P} x_{s,p,k} \cdot p - \sum_{p \in P} x_{s',p,k} \cdot p + 1$$

$$- |P| \cdot \left( 2 - \sum_{p \in P} x_{s,p,k} - \sum_{p \in P} x_{s',p,k} \right) \quad \forall i \in I; s, s' \in O_i; k \in K \quad (37)$$

$$\delta^{max} \geq \sum_{i \in I} f_i \cdot \delta_{i,k} \quad \forall k \in K \quad (38)$$

$$x_{s,p,k} \in \{0, 1\} \quad \forall s \in S; p \in P; k \in K \quad (39)$$

$$\sum_{s \in S} \sum_{k \in K} k \cdot x_{s,p,k} \leq \sum_{s \in S} \sum_{k \in K} k \cdot x_{s,p+1,k} \quad \forall p = 1, \dots, |P| - 1 \quad (40)$$

$$\sum_{t \in S} x_{t,p-1,k-1} + x_{s,p,k} + \sum_{t \in S} x_{t,p'-1,k} + x_{s',p',k+1} \leq 3 \quad \forall k = 2, \dots, |K| - 1; \quad (41)$$

$$p = 2, \dots, |P| - 1;$$

$$p' = p + 1, \dots, |P|;$$

$$s, s' \in S : s > s'$$

$$x_{s,1,1} + \sum_{t \in S} x_{t,p-1,1} + x_{s',p,2} \leq 2 \quad \forall p = 2, \dots, |P|; \quad (42)$$

$$s, s' \in S : s > s'$$

$$\sum_{t \in S} x_{t,p-1,k-1} + x_{s,p,k} + x_{s',p',k} + \sum_{t \in S} x_{t,p'+1,k+1} \leq 3 \quad \forall k = 2, \dots, |K| - 1; \quad (43)$$

$$p = 2, \dots, |P| - 2;$$

$$p' = p + 1, \dots, |P| - 1;$$

$$s, s' \in S : s > s'$$

$$x_{s,1,1} + x_{s',p,1} + \sum_{t \in S} x_{t,p+1,2} \leq 2 \quad \forall p = 2, \dots, |P| - 1; \quad (44)$$

$$s, s' \in S : s > s'$$

$$\sum_{t \in S} x_{t,p-1,|K|-1} + x_{s,p,|K|} + x_{s',|P|,|K|} \leq 2 \quad \forall p = 2, \dots, |P| - 1; \quad (45)$$

$$s, s' \in S : s > s'$$

$$x_{s,1,1} + x_{s',|P|,1} \leq 1 \quad \forall s, s' \in S : s > s' \quad (46)$$

Objective function (32) minimizes the maximum order spread. Constraints (33) assign each SKU to exactly one storage position and one kitting station. Analogously, constraints (34) assign exactly one SKU to each storage position and each storage position to exactly one kitting station. Furthermore, (35) assigns at least one SKU and storage position to each kitting station. The connection of kitting stations along the line is ensured by (36). Constraints (37) define the order spreads for each order and each kitting station. Derived from that, constraints (38) define the maximum order spread over all kitting stations. Finally, the model is completed by setting the variable domains in (39).

To enable a fair comparison with our previous MIP of Sect. 2.3, we enrich this model too with symmetry breakers. Specifically, the following ones are applied:

- Based on constraints (40), stations are sorted along the line in increasing index order.
- Constraints (41) to (42) specify the assignment of SKU sets to stations, so that the indexes of the first SKUs in each station are in increasing order.
- Constraints (43) to (45) specify the direction of the SKU sequences within each station by setting the first SKU's index smaller than that of the last one. The special case of  $|K| = 1$  is considered by (46).

In spite of these symmetry breakers, this model is outperformed by competitor WBSAP-MIP of Sect. 2.3. We evaluated the two MIPs on 3475 instances ranging from 5 to 25 SKUs. WBSAP-MIP (MIP2) was able to find the optimal solution in 893 (858) instances, found a feasible solution within 300 s in 1721 (1505) instances and missed feasible solutions in 861 (1112) instances. While runtimes are rather similar, WBSAP-MIP outperforms WBSAP-MIP2 with respect to solution quality. Therefore, we decided to move WBSAP-MIP2 into this appendix. All computational studies reported in the main paper only apply WBSAP-MIP.

## Appendix B: A MIP for SAP: Storage assignment for a given station workload

This appendix provides a MIP for problem SAP of Sect. 3.1.1. Applying the additional notation summarized in Table 7, SAP-MIP consists of objective function (47) and constraints (48) to (51).

**Table 7** Notation for SAP

$S_k$	Set of SKUs with $S_k = \{1, \dots,  S_k \}$ within station $k$ (indices $s, s'$ )
$O_{i,k}$	Picking order of engine $i$ in station $k$
$P_k$	Set of storage positions in station $k$ with $P_k = \{1, \dots,  P_k  =  S_k \}$ (index $p$ )
$Z_k$	Objective value: order spread in station $k$

**SAP-MIP:**

$$\text{Minimize } Z_k(x, \delta) = \sum_{i \in I} f_i \cdot \delta_{i,k} \quad (47)$$

subject to

$$\sum_{s \in S_k} x_{s,p} = 1 \quad \forall p \in P_k \quad (48)$$

$$\sum_{p \in P_k} x_{s,p} = 1 \quad \forall s \in S_k \quad (49)$$

$$\delta_{i,k} \geq \sum_{p \in P_k} x_{s,p} \cdot p - \sum_{p \in P_k} x_{s',p} \cdot p + 1 \quad \forall i \in I; s, s' \in O_{i,k} \quad (50)$$

$$x_{s,p} \in \{0, 1\} \quad \forall s \in S_k; p \in P_k \quad (51)$$

Objective function (47) minimizes the weighted order spread in kitting station  $k$  summarized over all picking orders. Constraints (48) and (49) ensure that each storage position receives exactly one SKU and vice versa. The order spreads for all picking orders are dimensioned by (50). Here, each pair of SKUs per order is evaluated, and the maximum distance among any such pair defines the order spread. Finally, constraints (51) set the domain of the binary variables.

We have carefully tested the application of this MIP within our decomposition approach. However, especially for larger instances solving this model took default solver Gurobi too much time to evaluate a large number of storage assignments. Our beam search approach delivers only heuristic results but is much faster, so that it turned out as the better option for this stage. For a matter of conciseness, we abstain from a detailed elaboration of these computational results.

**Table 8** Notation for WBP

$w_{p,p'}$	Weighted order spread for a station starting at $p$ and ending at $p'$
$M$	Big value, e.g., $M =  P  \cdot \max_{i \in I} f_i$
$l_{k,p}$	Binary variable: 1, if kitting station $k$ is ending at position $p$ (0, otherwise)

## Appendix C: A MIP for WBP: Workload balancing for a given storage assignment

This appendix provides a MIP for problem WBP, which is defined in Sect. 3.2.1 and part of our sequence first, balance second decomposition approach. First, we preprocess, the aggregated weighted order spread between positions  $p$  and  $p'$ :

$$w_{p,p'} = \sum_{i \in I} f_i \cdot \begin{cases} \max_{\substack{q = p, \dots, p' : \\ \phi_q \in O_i}} \{q\} - \min_{\substack{q = p, \dots, p' : \\ \phi_q \in O_i}} \{q\} + 1, & \text{if } \exists q \in \{p, \dots, p'\} : \phi_q \in O_i \\ 0, & \text{else} \end{cases} \quad (52)$$

This term is strongly related to (27). However, it also includes the weight of each order, i.e., the frequencies  $f_i$  and aggregates the weighted order spread over all orders  $i \in I$  for the storage area starting at position  $p$  and ending at position  $p'$ .

Applying these parameters and additional notation summarized in Table 8, WBP can also be formulated as a MIP (dubbed WBP-MIP) consisting of objective function (53) and constraints (54) to (59).

### WBP-MIP:

$$\text{Minimize } Z(l, \delta^{\max}) = \delta^{\max} \quad (53)$$

subject to

$$\sum_{p=1}^{|P|-1} l_{k,p} = 1 \quad \forall k = 1, \dots, |K| - 1 \quad (54)$$

$$l_{|K|,|P|} = 1 \quad (55)$$

$$\sum_{p=1}^{|P|-1} p \cdot l_{k+1,p} \geq \sum_{p=1}^{|P|-1} p \cdot l_{k,p} + 1 \quad \forall k = 1, \dots, |K| - 1 \quad (56)$$

**Table 9** Computational performance for different random solution approaches

	Small instance		Large instances	
	gapBest [%]	cpu time [s]	gapBest [%]	cpu time [s]
GRB	1.31	136.87	21.22	900.91
BFSS	0.22	75.10	4.03	720.03
SFBS	0.11	31.51	3.20	532.31
RND <sub>1</sub>	28.81	0.00	30.69	0.00
RND <sub>10</sub>	13.79	0.00	24.60	0.01
RND <sub>100</sub>	7.34	0.00	20.68	0.06
RND <sub>1000</sub>	4.46	0.03	18.10	0.57
RND <sub>10000</sub>	2.79	0.26	15.82	5.72
RND <sub>100000</sub>	1.71	2.58	13.82	57.30
RND <sub>1000000</sub>	1.00	25.94	12.38	573.58

$$\delta^{max} \geq w_{p,p'} - M \cdot (2 - l_{k,p-1} - l_{k+1,p'}) \quad \forall k = 2, \dots, |K|; \\ p, p' \in P : p < p' \quad (57)$$

$$\delta^{max} \geq w_{1,p} - M \cdot (1 - l_{1,p}) \quad \forall p \in P \quad (58)$$

$$l_{k,p} \in \{0, 1\} \quad \forall k \in K; p \in P \quad (59)$$

Objective function (53) minimizes the maximum order spread. Constraints (54) and (55) assign an end position to each kitting station, whereas constraints (56) sort the stations in increasing index order. Constraints (57) and (58) set the maximum order spread over all stations, where the latter handles the special case of the first station. Finally, the variable domains are set by (59).

## Appendix D: Computational performance of the random solution approach (RND)

In Sect. 4, we compare the performance of our sophisticated optimization procedures with the current status quo approach implemented by our partner in practice. There, the assignment of SKUs to storage positions is randomly chosen and the kitting stations are equally distributed along the storage area. To obtain insights regarding the impact of more random solutions on the performance, we report on further experiments with modified random approaches. Instead of drawing only a single random solution, we now generate several random SKU sequence each evaluated by the equal-number-of-SKUs-per-station policy of our engine producer. The best among our random solutions is finally returned. We denote the resulting procedures by RND<sub>#sol</sub>, where #sol gives the number of randomly generated storage assignments.



We applied these procedures to the small and large set of instances from Subsect. 4.2 for  $\#sol \in \{1, 10, 100, 1000, 10^4, 10^5, 10^6\}$ . The results are presented in Table 9. Note that to improve readability, we only list the average values for the entire data set rather than the results for each parameter combination. *gapBest* denotes the average objective gap to the best performing procedure.

The results of Table 9 clearly indicate that even drawing a significant amount of random sequences, so that the solution times are about as high as those of our more sophisticated optimization approach SFBS, is not competitive. This supports our choice of applying method SFBS (see Sect. 3.2) for solving WBSAP.

**Funding** Open Access funding enabled and organized by Projekt DEAL.

**Data availability** The datasets generated and analyzed during the current study are available from the corresponding author upon reasonable request.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Aarts EHL, Korst JHM, van Laarhoven JM (1997) Simulated annealing. In: Aarts EHL, Lenstra JK (eds) Local search in combinatorial optimization. Wiley, Hoboken, pp 91–120
- Bartholdi JJ III, Eisenstein DD (1996) A production line that balances itself. *Oper Res* 44:21–34
- Battini D, Boysen N, Emde S (2013) Just-in-time supermarkets for part supply in the automobile industry. *J Manag Control* 24:209–217
- Battini D, Calzavara M, Otto A, Sgarbossa F (2017) Preventing ergonomic risks with integrated planning on assembly line balancing and parts feeding. *Int J Prod Res* 55:7452–7472
- Boysen N, Flidner M, Scholl A (2008) Assembly line balancing: which model to use when? *Int J Prod Econ* 111:509–528
- Boysen N, Flidner M, Scholl A (2009) Sequencing mixed-model assembly lines: survey, classification and model critique. *Eur J Oper Res* 192:349–373
- Boysen N, Bock S (2011) Scheduling just-in-time part supply for mixed-model assembly lines. *Eur J Oper Res* 211:15–25
- Boysen N, Emde S, Hoeck M, Kauderer M (2015) A survey on part logistics in the automotive industry. *Eur J Oper Res* 242:107–120
- Boysen N, De Koster R, Füller D (2021) The forgotten sons: warehousing systems for brick-and-mortar retail chains. *Eur J Oper Res* 288:361–381
- Boysen N, Schulze P, Scholl A (2022) Assembly line balancing: what happened in the last fifteen years? *Eur J Oper Res* 301:797–814
- Calzavara M, Glock CH, Grosse EH, Persona A, Sgarbossa F (2017) Analysis of economic and ergonomic performance measures of different rack layouts in an order picking warehouse. *Comput Ind Eng* 111:527–536

- Caputo AC, Pelagagge PM, Salini P (2015) A decision model for selecting parts feeding policies in assembly lines. *Ind Manag Data Syst* 115:974–1003
- Caputo AC, Pelagagge PM, Salini P (2018) Selection of assembly lines feeding policies based on parts features and scenario conditions. *Int J Prod Res* 56:1208–1232
- Choi W, Lee Y (2002) A dynamic part-feeding system for an automotive assembly line. *Comput Ind Eng* 43:123–134
- De Koster R, Le-Duc T, Roodbergen KJ (2007) Design and control of warehouse order picking: a literature review. *Eur J Oper Res* 182:481–501
- Emde S, Boysen N (2012) Optimally locating in-house logistics areas to facilitate JIT-supply of mixed-model assembly lines. *Int J Prod Econ* 135:393–402
- Emde S, Fliedner M, Boysen N (2012) Optimally loading tow trains for JIT-supply of mixed-model assembly lines. *IIE Trans* 44:121–135
- Füßler D, Fedtke S, Boysen N (2019) The cafeteria problem: order sequencing and picker routing in on-the-line picking systems. *OR Spectr* 41:727–756
- Füßler D, Boysen N, Stephan K (2019) Trolley line picking: storage assignment and order sequencing to increase picking performance. *OR Spectr* 41:1087–1121
- Garey MR, Johnson DS (1979) *Computers and intractability: a guide to the theory of NP-completeness*. Freeman, New York
- Glock CH, Grosse EH, Abedinnia H, Emde S (2019) An integrated model to improve ergonomic and economic performance in order picking by rotating pallets. *Eur J Oper Res* 273:516–534
- Grosse EH, Glock CH, Jaber MY, Neumann WP (2015) Incorporating human factors in order picking planning models: framework and research opportunities. *Int J Prod Res* 53:695–717
- Gurobi Optimization, LLC (2021) Gurobi optimizer reference manual. <http://www.gurobi.com>
- Hanson R, Brolin A (2013) A comparison of kitting and continuous supply in in-plant materials supply. *Int J Prod Res* 51:979–992
- Held M, Karp RM (1962) A dynamic programming approach to sequencing problems. *J Soc Ind Appl Math* 10:196–210
- Kilic HS, Durmusoglu MB (2015) Advances in assembly line parts feeding policies: a literature review. *Assem Autom* 35:57–68
- Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. *Science* 220:671–680
- Limère V, Landeghem HV, Goetschalckx M, Aghezzaf EH, McGinnis LF (2012) Optimising part feeding in the automotive assembly industry: deciding between kitting and line stocking. *Int J Prod Res* 50:4046–4060
- Merengo C, Nava F, Pozzetti A (1999) Balancing and sequencing manual mixed-model assembly lines. *Int J Prod Res* 37:2835–2860
- Nourmohammadi A, Eskandari H, Fathi M (2019) Design of stochastic assembly lines considering line balancing and part feeding with supermarkets. *Eng Optim* 51:63–83
- Otto A, Scholl A (2013) Reducing ergonomic risks by job rotation scheduling. *OR Spectr* 35:711–733
- Otto A, Boysen N, Scholl A, Walter R (2017) Ergonomic workplace design in the fast pick area. *OR Spectr* 39:945–975
- Ritt M, Costa AM (2018) Improved integer programming models for simple assembly line balancing and related problems. *Int Trans Oper Res* 25:1345–1359
- Sali M, Sahin E, Patchong A (2015) An empirical assessment of the performances of three line feeding modes used in the automotive sector: line stocking vs. kitting vs. sequencing. *Int J Prod Res* 53:1439–1459
- Schmid NA, Limère V (2019) A classification of tactical assembly line feeding problems. *Int J Prod Res* 57:7586–7609
- Sternatz J (2015) The joint line balancing and material supply problem. *Int J Prod Econ* 159:304–318

## Authors and Affiliations

**Stefan Fedtke<sup>1</sup>**  · **Nils Boysen<sup>1</sup>** · **Patrick Schumacher<sup>2</sup>**

Nils Boysen  
nils.boysen@uni-jena.de

Patrick Schumacher  
p.schumacher@tu-braunschweig.de  
<https://www.tu-braunschweig.de/aip/pl>

<sup>1</sup> Friedrich-Schiller-Universität Jena Lehrstuhl für Operations Management, Carl-Zeiss-Str. 3, 07743 Jena, Germany

<sup>2</sup> Technische Universität Braunschweig Lehrstuhl für Produktion und Logistik, Mühlenfordtstraße 23, 38106 Braunschweig, Germany