Buckow, Jan-Niklas; Goerigk, Marc; Knust, Sigrid

Integrated pallet retrieval and processing in warehouses under uncertainty

OR Spectrum

Provided in Cooperation with:
Springer Nature

Suggested Citation: Buckow, Jan-Niklas; Goerigk, Marc; Knust, Sigrid (2025) : Integrated pallet retrieval and processing in warehouses under uncertainty, OR Spectrum, ISSN 1436-6304, Springer, Berlin, Heidelberg, Vol. 47, Iss. 3, pp. 817-855, https://doi.org/10.1007/s00291-024-00806-7

OR Spectrum (2025) 47:817–855
https://doi.org/10.1007/s00291-024-00806-7

ORIGINAL ARTICLE

Integrated pallet retrieval and processing in warehouses under uncertainty

Jan-Niklas Buckow[1] · **Marc Goerigk**[2] · **Sigrid Knust**[1]

Received: 20 August 2024 / Accepted: 12 December 2024 / Published online: 17 February 2025

Abstract

We study the problem of integrated pallet retrieval and successive processing in warehouses motivated by a practical company setting. A set of pallets has to be retrieved in a warehouse having a single stacker crane and multiple input/output-points, and the goal is to minimize the makespan. Previous research focuses on minimizing stacker crane travel times, while subsequent pallet processing times at the input/output-points are neglected. A distinction is made between a blocking and buffering variant, in which either no or sufficient buffer space is available at the input/output-points to temporarily store pallets there before they are processed. To hedge against uncertainties in the pallet processing times, we additionally apply robust optimization with budgeted uncertainty sets. We develop dynamic programming algorithms for the worst-case evaluation, identify polynomially solvable cases when there is only a single input/output-point, and present mathematical models for the general case with an arbitrary number of input/output-points. Our extensive computational study reveals that the integrated models yield considerable benefits compared to either ignoring the stacker crane travel times or the pallet processing times. We propose heuristic algorithms that provide good solutions for large instances in a short amount of computational time.

Keywords Warehouse · Stacker crane scheduling · Multiple input/output-points · Integrated pallet retrieval and processing · Robust optimization · Budgeted uncertainty

✉ Jan-Niklas Buckow
jabuckow@uni-osnabrueck.de

Marc Goerigk
marc.goerigk@uni-passau.de

Sigrid Knust
sknust@uni-osnabrueck.de

[1] Institute of Computer Science, Osnabrück University, Wachsbleiche 27, 49090 Osnabrück, Germany

[2] Business Decisions and Data Science, University of Passau, Dr.-Hans-Kapfinger-Straße 30, 94032 Passau, Germany

1 Introduction

In modern warehouses, *automated storage/retrieval systems* (AS/RS) are popular to increase the efficiency, where all items are packed into unit-load pallets that are stored and retrieved from the warehouse by an automated stacker crane via specific *input/output-points* (I/O-points). To obtain an overview of warehouses with AS/RS and different stacker crane scheduling problems therein, we refer to Roodbergen and Vis (2009) and Boysen and Stephan (2016).

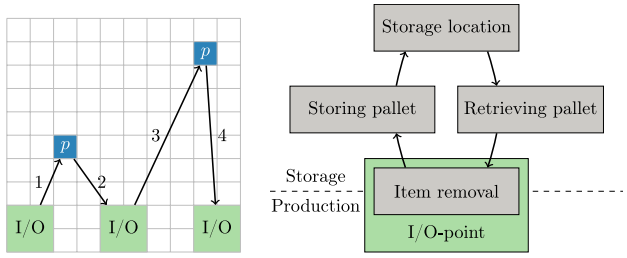
1.1 Motivation

Optimizing storage and retrieval operations in warehouses with AS/RS is important to minimize costs and ensure a smooth item flow in supply chains. While most publications in that field only deal with warehouses having a single I/O-point, Buckow et al (2024) studied a new stacker crane scheduling problem motivated by a practical setting in a company, where pallets are retrieved in a warehouse with multiple I/O-points, and employees remove specific items from the pallets for a further production process. In their basic problem variant, both the pallet retrieval sequence and the assignment of pallets to I/O-points have to be determined, and the goal is to minimize the total travel time of the stacker crane to perform all retrievals.

We extend the problem studied by Buckow et al (2024) by additionally taking pallet processing times at the I/O-points into account. Our setting is motivated by the same industrial company, where a given set of pallet retrieval requests has to be performed to prepare the next production shift. Their stacker crane starts at a specific depot I/O-point, and each retrieved pallet is allowed to be brought to an arbitrary I/O-point. At the I/O-points, there incur pallet processing times, mainly because removing items from the pallets by the employees takes some time. In addition, some items have to be sawn in half, with one half remaining in the pallet and the other half being needed in the next production shift. These pallet processing times thus vary between different retrieval requests, depending on the item type and quantity removed. To start the next production shift in time, the processing of the last pallet should be completed as early as possible.

When the stacker crane reaches an I/O-point, it drops the pallet to be retrieved there and then picks up another pallet to be stored. This is possible because at each I/O-point, the pallet previously processed there is available as a storage request (there usually remain items in the pallets after their processing and they are thus returned to the storage). After reaching an I/O-point, the stacker crane moves to the next pallet to be retrieved and swaps it with that currently held on the stacker crane. Note that the company's stacker crane can perform such pallet swaps as it has an additional buffer location. Due to these pallet swaps, the storage of the pallets that are returned to the warehouse is performed implicitly and does not need to be considered explicitly.

Figure 1 exemplifies the processes in the warehouse of the mentioned company. On the one hand, an example storage layout with three I/O-points and



(a) Storage layout (front view). (b) Pallet retrieval and storage process.

Fig. 1 Exemplary warehouse overview and the processes therein

possible stacker crane movements for retrieving two pallets are shown in Fig. 1a. The leftmost I/O-point corresponds to the depot at which the stacker crane is initially located, and the numbers indicate the order of the stacker crane movements. Each cell represents a storage location and accommodates exactly one pallet. On the other hand, the process of retrieving and storing pallets is illustrated in Fig. 1b. After a pallet is retrieved from its initial storage location, the stacker crane brings it to an I/O-point to remove items from it. Note that the I/O-points are placed at the border between the storage and production area. When the item removal is finished, the pallet is returned to the warehouse (but possibly to a different storage location than its initial one) as soon as the stacker crane next reaches the I/O-point after retrieving another pallet.

The aforementioned company runs a large high-bay warehouse equipped with a single stacker crane that has an additional buffer location and is thus designed to perform pallet swaps. According to the stacker crane's manufacturer, there are other industrial firms operating similar warehouses, with mainly metal processors as well as producers of windows and doors being among these firms. All these firms are processing long and heavy items, including rods and metal profiles. Hence, there also seem to be other companies facing a similar problem setting to ours in their distribution centers. Furthermore, note that due to the heaviness of their pallets, the stacker crane's heightened capacity cannot be used to transport two pallets at once, meaning that pallet swaps are the preferred operating mode.

The processing of a pallet at an I/O-point makes it temporarily unavailable and thus impacts the choices of the retrieval I/O-points. We address this setting more accurately by proposing an integrated problem considering both the stacker crane travel times as well as the pallet processing times. Although the approaches presented by Buckow et al (2024) provide feasible solutions for our new integrated problem, they completely neglect the pallet processing times, requiring more complex solution approaches to fully exploit the whole optimization potential.

From discussions with planners at the aforementioned company, we know that the processing of some pallets may actually consume more time than originally expected, as unpredictable disruptions occasionally occur when removing items from the pallets at the I/O-points. For instance, the working speeds of the employees differ, and they occasionally take a short break. Moreover, a vehicle required

to further transport the removed items may not be available in time. Therefore, we additionally take uncertainties in the pallet processing times into account, handling them by applying robust optimization.

Our integrated problem can be viewed as a parallel machine scheduling problem with transportation considerations and uncertainties. The I/O-points correspond to the machines, the pallets are the jobs to be processed, and the stacker crane travel times are setup times needed to prepare the I/O-points before processing the pallets. Our problem is also related to hybrid/flexible flow-shop scheduling problems, where the stacker crane forms a first stage with a single machine, and the I/O-points correspond to a second stage with multiple machines.

1.2 Literature

Overall, our integrated problem is related to the three research fields of machine scheduling, warehousing problems, and robust optimization. In the following, we delimit our problem in relation to these three research fields, which intersect.

There are several studies researching scheduling problems with setup times or transportation considerations (see the surveys by Allahverdi et al (1999, 2008), Allahverdi (2015) and Hosseini et al (2024)). However, in contrast to the existing literature, our problem involves a single stacker crane with a transportation capacity of one. Therefore, the setup or transportation time to prepare the processing of a pallet depends not only on the pallet itself, its chosen retrieval I/O-point and its immediate predecessor there, but also on the scheduling decisions of all pallets previously retrieved, making our problem more difficult.

In recent decades, supply chain scheduling has emerged as a new research field that links machine scheduling problems with supply chain management decisions (see the book by Chen and Hall (2022)). Several such problems also deal with transportation considerations, and it is usually assumed that goods are produced on machines in a factory and then transported to customers or warehouses. Nevertheless, these steps are reversed in our problem, since the transport of a pallet takes place before it is processed at an I/O-point. Furthermore, in these supply chain scheduling problems with transportation considerations, all machines within a factory are typically modeled as one location, while we assume that both the pallets and the I/O-points can take different arbitrary locations within the warehouse.

In flow-shop scheduling problems, a distinction is made between blocking and buffering variants, depending on whether there is sufficient buffer space to temporarily store jobs when their next processing machine is busy (see the survey by Miyata and Nagano (2019)). In many warehouses with AS/RS, there is no buffer space at all at the I/O-points, meaning that the stacker crane must wait until the I/O-point is free again before it can drop off a pallet there (in this case, we call the stacker crane being blocked). Although there appears to be little research dealing with such buffer spaces in warehouses, de Koster et al (2012) and Jiang et al (2022) study manual pick-and-sort warehouses, where picked batches are buffered before they are sorted. While de Koster et al (2012) assume unlimited buffer space, Jiang et al (2022) suppose that the buffer capacity is limited and pickers are blocked if the buffer is full.

Teck et al (2024) studied a warehouse scheduling problem with processing time variability, where multiple robots are lifting pods from the storage to workstations and stochastic optimization is used in order to protect against the uncertain processing times. Gong and de Koster (2011) conducted a survey on stochastic optimization in warehouse operations in general, and described different sources of uncertainty, which can be both within and external to the warehouse system or the supply chain. For example, the uncertainties within the warehouse include human factors such as labor absence or fluctuations in the working speed. Uncertainties in the supply chain may arise from failures in the shipping equipment like trucks or pallet jacks.

As already noted by Gong and de Koster (2011), there remain academic blanks in the field of robust approaches for warehouse operations. Unlike stochastic optimization, the former does not require a probability distribution of the scenarios in the uncertainty set, making it applicable even without such knowledge. For in-depth discussions of the robust optimization paradigm, we refer to the books by Ben-Tal et al (2009), Bertsimas and den Hertog (2022), and Goerigk and Hartisch (2024). Ben-Tal et al (2004) introduced the concept of adjustable robust optimization and applied it to an inventory management problem with demand uncertainty, where some decisions must be made before the realization of the uncertain data, while other decisions can be made after the realization (see also the survey by Yanıkoğlu et al (2019)).

Since the survey of Gong and de Koster (2011), there are some further studies on robust optimization in warehouses. Ang et al (2012) examined a robust storage assignment problem with variable supply and uncertain demand. Most papers dealing with robust warehouse operations considered inventory management problems with demand or lead time uncertainty (for instance, see Thorsen and Yao (2015), Jiu (2022), Qiu et al (2022) and Sun et al (2024)). Nevertheless, instead of scheduling pallet retrievals, other decisions are optimized in these robust problems, mainly determining periodic inventory review policies. There seem to be no studies on robust optimization in warehouse operations that consider pallet retrieval optimization or uncertain processing times at the I/O-points.

The type of robust problem we consider has some similarity to Bruni et al (2017) and Bold and Goerigk (2021), who study a resource-constrained project scheduling problem, where the activity durations are subject to budgeted uncertainty. Handling the problem as a robust two-stage approach, the resource allocation decisions have to be made before the uncertain activity durations become known, but the activity start times can be fixed after the realization of the activity durations.

1.3 Contributions

In this paper, we study the *pallet retrieval and processing problem* (PRPP), where a given set of pallets has to be retrieved and processed in a warehouse equipped with a single stacker crane and multiple I/O-points. The PRPP can be considered as an integrated problem, generalizing both existing pallet retrieval optimization as well as machine scheduling problems from the literature. Our goal is to minimize the makespan, which is defined as the completion time of the last pallet processed.

We also investigate the gain of having buffer spaces at the I/O-points to temporarily store pallets there before they are processed, and distinguish between two problem variants. In the first variant, the stacker crane temporarily blocks when approaching an occupied I/O-point (blocking variant), while in the second variant, sufficient buffer space always allows pallets to be dropped off even at occupied I/O-points (buffering variant).

To protect against uncertainties in the pallet processing times, we additionally incorporate robust optimization into our problem. We assume that nominal values and upper bounds on the possible processing time delays are known by the decision maker. As it seems unlikely that the processing times of all pallets take their worst-case values at the same time, we assume budgeted uncertainty sets, which limit the total number of deviating processing times.

We identify special cases of the PRPP with only a single I/O-point, where the integrated problem becomes polynomially solvable (in the blocking case even the robust variant). For the general case with an arbitrary number of I/O-points, we develop mathematical models for the blocking and buffering variant with and without robustness, and present dynamic programming algorithms for the worst-case evaluation, i.e., finding the worst possible objective value under all considered scenarios. Our results show that the integrated consideration of pallet retrieval and processing is crucial to obtain good solutions. Moreover, schedules obtained by our robust models are considerably less vulnerable to delays in the pallet processing times. Having buffers at the I/O-points leads to clear reductions in the makespan.

This paper is organized as follows. First, the PRPP and its variants are introduced and formally defined in Sect. 2. Afterwards, in Sect. 3, we theoretically study the special case with a single I/O-point for both the blocking and the buffering variant, and reveal variants that are solvable in polynomial time. In addition, the general multiple I/O-point case is tackled in Sect. 4, for which we develop mathematical models and dynamic programming algorithms for the worst-case evaluation. In Sect. 5, it follows a detailed computational study, in which we first investigate the benefits of the newly developed integrated mathematical models in both the nominal and robust cases. Besides, the benefits of having a buffer as well as some heuristics are evaluated. Eventually, Sect. 6 concludes the paper.

2 Problem definition

The PRPP can be regarded as an integrated two-stage problem, where the first stage involves retrieving each pallet from its storage location and then moving it to one of the different I/O-points. Once the stacker crane has transported a pallet to an I/O-point, the pallet is processed there in a second stage with a pallet dependent processing time. Note that the resulting makespan is impacted by both the stacker crane travel times to retrieve the pallets during the first stage, and the pallet processing times during the second stage.

Due to its stages, the PRPP is related to hybrid or flexible flow-shop scheduling problems, where a set of jobs must be processed successively in a series of stages, each stage consisting of one or more machines. In the PRPP, the stacker crane that retrieves

the pallets forms the first stage with a single machine, while the I/O-points where the pallets are processed correspond to the second stage with multiple machines. However, the key distinction of the PRPP to traditional hybrid flow-shop scheduling problems is that we also tackle transportation considerations: the stacker crane travel times (corresponding to the processing times in the first stage) are not constant per pallet (corresponding to a job), but depend on both the pallet retrieval sequence and the assignment of pallets to I/O-points.

As in the case of flow-shop scheduling problems, we consider two different variants for the PRPP. In the blocking variant (PRPP_{Bl}), the I/O-points have no buffer space, blocking the stacker crane when bringing a pallet to an I/O-point until the processing of the pallets previously brought to the same I/O-point is completed. The stacker crane can then drop off the new pallet and continue to retrieve the next pallet in the warehouse. On the other hand, in the buffering variant (PRPP_{Bu}), there is sufficient buffer space available at the I/O-points where pallets can be stored temporarily, avoiding idle times of the stacker crane, i.e., it is always possible to drop off a pallet immediately when the stacker crane arrives. Furthermore, the special cases with only a single I/O-point of these problem variants we examine in Sect. 3 are denoted by 1PRPP_{Bl} and 1PRPP_{Bu}, respectively.

More formally, the PRPP can be stated as follows. We are given a warehouse with n pallets $P = \{p_1, \ldots, p_n\}$, m different I/O-points $\Theta = \{\theta_1, \ldots, \theta_m\}$, and a stacker crane which is initially located at the depot I/O-point $\theta_{\text{Depot}} \in \Theta$ (note that our solution approaches can easily be adapted to handle the case of an arbitrary initial location). Moreover, each pallet $p \in P$ is associated with a processing time $t(p)$, where we later assume that these processing times are subject to uncertainty. In total, we have $\mu = n + m$ locations $L = \{\ell_1, \ldots, \ell_\mu\}$, i.e., a location $\ell(p)$ for each pallet $p \in P$ as well as a location $\ell(\theta)$ for each I/O-point $\theta \in \Theta$. Note that each storage location corresponding to a pallet can accommodate only a single pallet. For all locations $\ell_i, \ell_j \in L$, there are stacker crane travel times $\tau[\ell_i, \ell_j]$. In real warehouses, these stacker crane travel times typically result from a metric, i.e., they are symmetric and satisfy the triangle inequality. However, we do not impose any restrictions on these travel times, resulting in a more general problem setting.

We denote by Π the set of all permutations of pallets P, and $\pi = (\pi_1, \ldots, \pi_n) \in \Pi$ corresponds to a specific sequence in which the pallets are retrieved. In addition, let $\alpha(p) \in \Theta$ describe the selected I/O-point to which pallet $p \in P$ is brought and then processed, and $\alpha = (\alpha(p_1), \ldots, \alpha(p_n)) \in \Theta^n$ represents an assignment of pallets to I/O-points. The goal of the PRPP is to find both a pallet retrieval sequence $\pi \in \Pi$ and an assignment $\alpha \in \Theta^n$ of pallets to I/O-points such that the makespan $\text{C}_{\max}\colon \Pi \times \Theta^n \to \mathbb{R}_+$ is minimized, i.e.,

$$\min_{(\pi,\alpha)\in\Pi\times\Theta^n} \text{C}_{\max}(\pi, \alpha, t), \tag{1}$$

with t being the possibly uncertain processing times parameters and the objective function $\text{C}_{\max}$ being defined as

$$\text{C}_{\max}(\pi, \alpha, t) = \max_{k=1}^{n} \{s_k(\pi, \alpha, t) + t(\pi_k)\}, \tag{2}$$

where for $k = 1, \ldots, n$ the value $s_k(\pi, \alpha, t)$ denotes the start time of processing the kth pallet retrieved by the stacker crane. The start time of processing the first pallet retrieved is given by

$$s_1(\pi, \alpha, t) = \tau[\ell(\theta_{\text{Depot}}), \ell(\pi_1)] + \tau[\ell(\pi_1), \ell(\alpha(\pi_1))], \tag{3}$$

meaning the stacker crane has to move from its initial location $\ell(\theta_{\text{Depot}})$ to $\ell(\pi_1)$, where the first pallet is initially positioned, and bring it to the chosen target I/O-point $\alpha(\pi_1)$. In the following, for $k = 1, \ldots, n$, we use $pred(k) \in \{0, \ldots, k-1\}$ to denote the position in π of the pallet previously processed at the same I/O-point as pallet π_k, with $pred(k) = 0$ if the pallet at position k is the first one processed at its I/O-point, and we define $s_0(\pi, \alpha, t) = 0$ and $t(\pi_0) = 0$. Then, for $k = 2, \ldots, n$, the start time values can be recursively calculated by

$$s_k(\pi, \alpha, t) = \max\{s_{pred(k)}(\pi, \alpha, t) + t(\pi_{pred(k)}), r_k(\pi, \alpha, t)\}, \tag{4}$$

where the left term considers the completion time of the pallet previously processed at the same I/O-point, and $r_k(\pi, \alpha, t)$ is the time when the stacker crane has retrieved the kth pallet and brought it to the chosen target I/O-point. Due to the missing buffer space in the case of PRPP_{Bl}, the stacker crane must wait at the chosen I/O-point when retrieving a pallet until the processing of the preceding pallet at the same I/O-point is completed. In this scenario, we call the stacker crane blocking, and for $k = 2, \ldots, n$, we have

$$r_k(\pi, \alpha, t) = s_{k-1}(\pi, \alpha, t) + \tau[\ell(\alpha(\pi_{k-1})), \ell(\pi_k)] + \tau[\ell(\pi_k), \ell(\alpha(\pi_k))]. \tag{5}$$

However, in the case of PRPP_{Bu}, a sufficient number of pallets can be buffered at each I/O-point. Hence, the stacker crane can move without any interruptions until all pallets are retrieved, and we have to replace (5) by

$$r_k(\pi, \alpha, t) = s_1(\pi, \alpha, t) + \sum_{k'=2}^{k} (\tau[\ell(\alpha(\pi_{k'-1})), \ell(\pi_{k'})] + \tau[\ell(\pi_{k'}), \ell(\alpha(\pi_{k'}))]). \tag{6}$$

Example 1 Consider the example instance of the PRPP with $n = 6$ pallets, $m = 2$ different I/O-points and the depot I/O-point $\theta_{\text{Depot}} = \theta_1$ as shown in Table 1. The pallet processing times are displayed in Table 1a, and the stacker crane travel times in Table 1b. Assuming the pallet retrieval sequence $\pi = (p_1, p_2, p_3, p_4, p_5, p_6)$ and the assignment of pallets to I/O-points $\alpha = (\theta_1, \theta_2, \theta_2, \theta_1, \theta_2, \theta_1)$, the corresponding schedules for PRPP_{Bl} and PRPP_{Bu} are shown in Fig. 2 and 3. There, the activities of the stacker crane and the I/O-points are visualized, where idle times are represented as hatched area, and blocking times of the stacker crane are drawn as dotted area. In the case of PRPP_{Bl}, the stacker crane blocks and has idle times after the processing of the pallets p_3 and p_5, since their target I/O-point keeps busy for a while, resulting in a makespan of $C_{\max} = 30$. In contrast, there are no stacker crane idle times in the case of PRPP_{Bu}, leading to a smaller makespan of $C_{\max} = 27$.

Table 1 Example instance of the PRPP

(a) Pallet processing times

j	1	2	3	4	5	6
$t(p_j)$	11	3	12	5	4	3

(b) Stacker crane travel times $t[\ell_i, \ell_j]$

	$\ell(\theta_1)$	$\ell(\theta_2)$	$\ell(p_1)$	$\ell(p_2)$	$\ell(p_3)$	$\ell(p_4)$	$\ell(p_5)$	$\ell(p_6)$
$\ell(\theta_1)$	0	2	1	3	3	4	1	2
$\ell(\theta_2)$	2	0	2	3	1	4	1	2
$\ell(p_1)$	1	2	0	2	3	3	1	1
$\ell(p_2)$	3	3	2	0	3	1	2	1
$\ell(p_3)$	3	1	3	3	0	4	2	3
$\ell(p_4)$	4	4	3	1	4	0	3	2
$\ell(p_5)$	1	1	1	2	2	3	0	1
$\ell(p_6)$	2	2	1	1	3	2	1	0

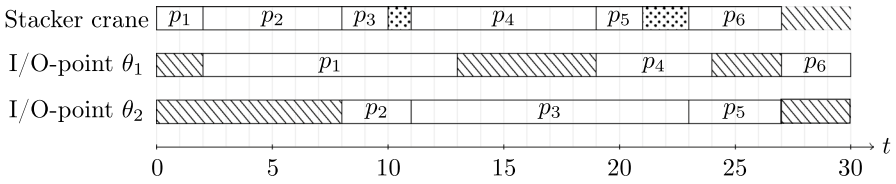


Fig. 2 Example schedule of PRPP_{Bl}

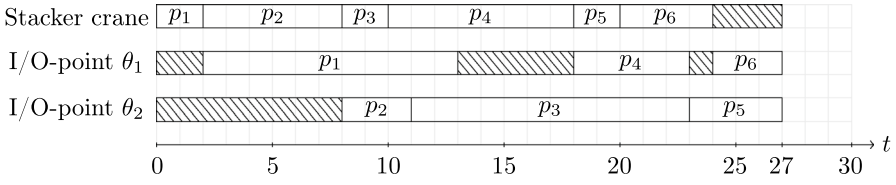


Fig. 3 Example schedule of PRPP_{Bu}

In the robust variants of the PRPP that we denote by PRPP^{R}, each pallet $p \in P$ has a nominal processing time $\bar{t}(p)$ and a worst-case processing time $\bar{t}(p) + \hat{t}(p)$, where $\hat{t}(p)$ denotes the potential delay of p. Following the "budgeted uncertainty" concept initially introduced by Bertsimas and Sim (2004), we assume that at most $\Gamma \leq n$ pallets are allowed to take their worst-case processing times at once. Thus, our uncertainty set is given by

Table 2 Example processing times and potential delays

j	1	2	3	4	5	6
$\bar{t}(p_j)$	11	3	12	5	4	3
$\hat{t}(p_j)$	8	4	2	7	1	1

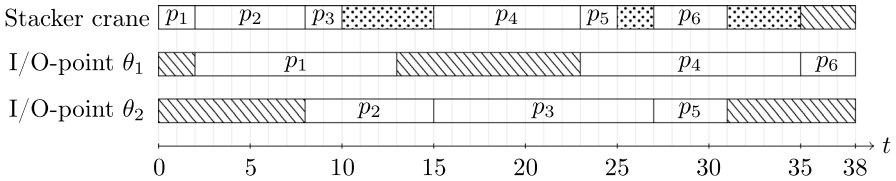


Fig. 4 Example schedule of PRPP_{Bl} showing the adversary's decisions

$$U(\Gamma) = \left\{ t \in \mathbb{R}^n \;\middle|\; t(p_j) = \bar{t}(p_j) + \delta_j \cdot \hat{t}(p_j); \delta_j \in \{0,1\}; j = 1, \ldots, n; \sum_{j=1}^{n} \delta_j \leq \Gamma \right\}.$$

After the uncertainty set $U(\Gamma)$ has been defined, the robust counterpart of our nominal problem can be written as

$$\min_{(\pi,\alpha)\in\Pi\times\Theta^n} \max_{t\in U(\Gamma)} \mathrm{C}_{\max}(\pi,\alpha,t), \tag{7}$$

where we first have to determine a pallet retrieval sequence π and an assignment α of pallets to I/O-points before the uncertain pallet processing times become known. Second, the actual start times for the pallet processing at their chosen I/O-points have to be fixed. This inner problem can be viewed as an adversary who aims to select a worst-case scenario of pallet delays for given π and α such that the makespan $\mathrm{C}_{\max}$ is maximized.

Example 2 Recall the example instance of the PRPP shown in Table 1 and the nominal solutions with $\pi = (p_1, p_2, p_3, p_4, p_5, p_6)$ and $\alpha = (\theta_1, \theta_2, \theta_2, \theta_1, \theta_2, \theta_1)$ as shown in Fig. 2 and 3 for PRPP_{Bl} and PRPP_{Bu}, respectively. For the robust variants $\text{PRPP}^{\text{R}}_{\text{Bl}}$ and $\text{PRPP}^{\text{R}}_{\text{Bu}}$, we assume $\Gamma = 2$ and the nominal processing times and their potential delays as listed in Table 2, where the nominal values are the same as the processing times already shown in Table 1a. In the case of $\text{PRPP}^{\text{R}}_{\text{Bl}}$, the adversary would delay the pallets p_2 and p_4, resulting in the schedule depicted in Fig. 4 with the new makespan $\mathrm{C}_{\max} = 38$. Note that delaying pallet p_2 immediately increases the makespan as the stacker crane blocks before starting the retrieval of pallet p_4. However, in the case of $\text{PRPP}^{\text{R}}_{\text{Bu}}$, the adversary would delay the pallets p_1 and p_4, leading to the schedule visualized in Fig. 5 with the makespan $\mathrm{C}_{\max} = 36$. Here, the delay in the pallet processing times first fills the idle times on the corresponding I/O-points before causing delays in the processing of the consecutive pallets.

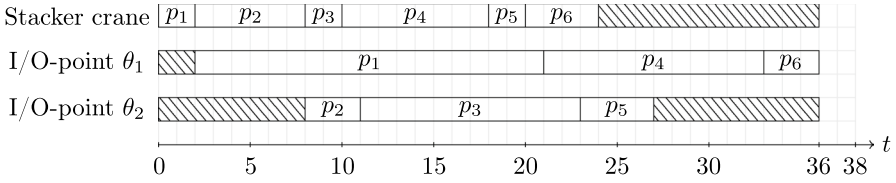


Fig. 5 Example schedule of PRPP_{Bu} showing the adversary's decisions

3 Single I/O-point

This section aims to solve our problem variants more efficiently for the case with a single I/O-point compared to the general case with multiple I/O-points by exploiting specific properties. When having just a single I/O-point $\theta_{\text{Depot}} \in \Theta$, each pallet has to be retrieved and processed there, and hence we only have to determine the pallet retrieval sequence π, while the assignment α of pallets to I/O-points is fixed. Moreover, the travel time durations $d(p) = \tau[\ell(\theta_{\text{Depot}}), \ell(p)] + \tau[\ell(p), \ell(\theta_{\text{Depot}})]$ required by the stacker crane to retrieve a pallet $p \in P$ are only dependent on the pallet itself.

As we will discuss next, the problem variants 1PRPP_{Bl} and 1PRPP_{Bu} with a single I/O-point are equivalent to the flow-shop scheduling problems $F2|blocking|\text{C}_{\max}$ and $F2||\text{C}_{\max}$, respectively. In these flow-shop scheduling problems, a set of jobs has to be scheduled on two machines. Each job has two operations, where the first operation has to be executed on the first machine and must be completed before the second operation can start on the second machine. The goal is to find a sequence of these jobs resulting in the smallest makespan, which is defined as the completion time of the last job.

Comparing $F2|blocking|\text{C}_{\max}$ and $F2||\text{C}_{\max}$ to our problem variants 1PRPP_{Bl} and 1PRPP_{Bu}, the jobs correspond to the pallets, the first machine is represented by the stacker crane, and the second machine matches the single I/O-point. The processing times for the jobs $p \in P$ on the first and second machine are equal to the stacker crane travel durations $d(p)$ and the pallet processing times $t(p)$, respectively. In the variant $F2|blocking|\text{C}_{\max}$, as in the case of 1PRPP_{Bl}, the first machine (stacker crane) blocks when the first operation of a job (pallet retrieval) is completed, but the second operation (pallet processing) cannot start while the second machine (I/O-point) is busy. In contrast, there are no such blocking restrictions in the variant $F2||\text{C}_{\max}$ as in the case of 1PRPP_{Bu}.

Since the problem $F2||\text{C}_{\max}$ with n jobs is well-known to be solvable in $O(n \cdot \log n)$ using the algorithm presented by Johnson (1954), the nominal variant 1PRPP_{Bu} can also be solved efficiently with the same runtime. However, Levorato et al (2022) state that the complexity of the robust counterpart of $F2||\text{C}_{\max}$ where the processing times on both machines are subject to budgeted uncertainty and each machine has its own uncertainty budget remains an open question.

Currently, the complexity of the more special problem $1\text{PRPP}^{\text{R}}_{\text{Bu}}$ with only uncertain processing times on the single I/O-point also remains unknown.

The problem $F2|blocking|\text{C}_{\max}$ can be transformed to a special case of the well-known *traveling salesman problem* (TSP), called *large* TSP (LTSP), as shown by Reddi and Ramamoorthy (1972). The LTSP was initially formulated by van Dal et al (1993), where a complete directed graph $G = (V, A)$ is given, and each node $v \in V$ is associated with two non-negative numbers $a(v)$ and $b(v)$. For an arc $(v, w) \in A$, its costs are given by $c(v, w) = \max\{a(v), b(w)\}$, and the aim is to find a Hamiltonian cycle on G with minimum total costs. Given that the LTSP with n nodes can be solved in $O(n \cdot \log n)$ by the algorithm of Gilmore and Gomory (1964), the variant 1PRPP_{Bl} can be solved in the same way.

In the following, we will show that the robust variant $1\text{PRPP}^{\text{R}}_{\text{Bl}}$ can also be solved efficiently by proving that a specific robust counterpart of the LTSP is solvable in polynomial time. In order to derive this counterpart, we first show how variant 1PRPP_{Bl} can be transformed to the LTSP. Although the equivalent problem $F2|blocking|\text{C}_{\max}$ is already known to be a special case of the LTSP (Reddi and Ramamoorthy 1972), we repeat the idea of the transformation to make our paper self-contained.

For a given instance of 1PRPP_{Bl} with n pallets to be retrieved, a corresponding LTSP instance has $\eta = n + 1$ nodes $V = \{v_0, v_1, \ldots, v_n\}$, where v_0 is a dummy node, and for $j = 1, \ldots, n$, node v_j matches pallet $p_j \in P$. Assuming that the pallets are processed in the sequence $\pi = (p_1, \ldots, p_n)$, for successive pallets p_j and p_{j+1}, the makespan increases due to the blocking condition by the maximum of the pallet processing time $t(p_j)$ and the stacker crane travel duration $d(p_{j+1})$ (the formation of the maximum terms is illustrated in Fig. 6). As non-negative numbers for the nodes, we hence have $a(v_0) = b(v_0) = 0$ for the dummy node v_0, and $a(v_j) = t(p_j)$ as well as $b(v_j) = d(p_j)$ for each $p_j \in P$. Therefore, to cope with variant $1\text{PRPP}^{\text{R}}_{\text{Bl}}$ having uncertain pallet processing times $t(p)$ for all $p \in P$, we define LTSP^{R} as a specific robust counterpart of the LTSP, where the values $a(v)$ for $v \in V$ are subject to budgeted uncertainty, while the values $b(v)$ remain certain. Due to the discussion above, the problems $1\text{PRPP}^{\text{R}}_{\text{Bl}}$ and LTSP^{R} are equivalent.

In LTSP^{R}, we denote for a node $v \in V$ by $\overline{a}(v)$ the nominal first value, and by $\overline{a}(v) + \hat{a}(v)$ its worst-case value, where $\hat{a}(v)$ denotes the first value's deviation. Then, each arc $(v, w) \in A$ has the nominal arc costs $\overline{c}(v, w) = \max\{\overline{a}(v), b(w)\}$, and its maximum cost value is $\overline{c}(v, w) + \hat{c}(v, w) = \max\{\overline{a}(v) + \hat{a}(v), b(w)\}$. Therefore, we get the arc cost deviation $\hat{c}(v, w) = \max\{\overline{a}(v) + \hat{a}(v), b(w)\} - \max\{\overline{a}(v), b(w)\}$.

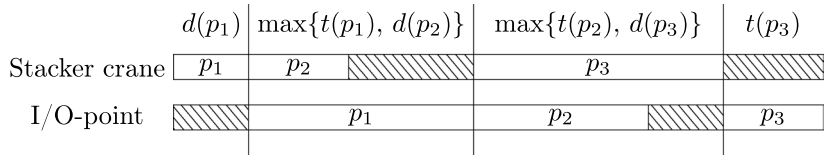


Fig. 6 Formation of the maximum terms in 1PRPP_{Bl}

Note that in the LTSP the total tour costs remain the same when traversing a Hamiltonian cycle in the opposite direction, and therefore the roles of the values $a(v)$ and $b(v)$ are interchangeable. This means that LTSP$^{\mathrm{R}}$ as defined above is the same as the robust problem with certain values $a(v)$ and uncertain values $b(v)$.

Next, we will derive for LTSP$^{\mathrm{R}}$ a result similar to Bertsimas and Sim (2003), who proved for a large class of combinatorial optimization problems that their robust counterparts with budgeted uncertainty belong to the same complexity class as their nominal variants. More precisely, for these combinatorial optimization problems, they showed that an optimal solution for the robust counterpart can be determined by solving several instances of the nominal problem with a modified cost structure where the number of instances to be considered is linear in the input size. Nevertheless, this result of Bertsimas and Sim (2003) is not directly applicable to LTSP$^{\mathrm{R}}$, because its input size is linear in the number of nodes, but not in the number of arcs for which the cost structure has to be modified. However, we will show in Lemma 1 that an optimal solution of LTSP$^{\mathrm{R}}$ can be computed by solving a quadratic number of nominal TSP instances with modified costs.

Lemma 1 *Let $I_{\mathrm{LTSP^R}}$ be an* LTSP$^{\mathrm{R}}$ *instance with η nodes. Then an optimal solution of $I_{\mathrm{LTSP^R}}$ can be determined by solving $\eta^2 + 1$ nominal* TSP *instances I'_{TSP} with each arc $(v, w) \in A$ having the specific costs $c'(v, w) = \overline{c}(v, w) + \max\{0, \hat{c}(v, w) - \lambda\}$ and $\lambda \geq 0$.*

Proof In each solution of $I_{\mathrm{LTSP^R}}$, at most Γ nodes $v \in V$ are allowed to take their worst-case values $\overline{a}(v) + \hat{a}(v)$, and we can hence formulate LTSP$^{\mathrm{R}}$ as

$$\min_{x \in X} \quad \max_{\substack{\delta \in \{0,1\}^{\eta} \\ \sum_{v \in V} \delta_v \leq \Gamma}} \quad \sum_{(v,w) \in A} (\overline{c}(v, w) + \hat{c}(v, w) \cdot \delta_v) \cdot x_{vw}, \tag{8}$$

where $X \subseteq \{0, 1\}^{\eta \times \eta}$ and each $x \in X$ forms a Hamiltonian cycle. Here, the inner problem tackled by the adversary can be written as the linear program

$$\max \quad \sum_{(v,w) \in A} (\overline{c}(v, w) + \hat{c}(v, w) \cdot \delta_v) \cdot x_{vw} \tag{9}$$

$$\text{s.t.} \quad \sum_{v \in V} \delta_v \leq \Gamma \tag{10}$$

$$0 \leq \delta_v \leq 1 \quad v \in V, \tag{11}$$

where the continuous relaxation is used for the variables δ_v for all $v \in V$, as the constraint matrix is totally unimodular. In addition, for all arcs $(v, w) \in A$ the values x_{vw} are already fixed in the outer stage, and considering the nominal arc costs hence simply results in an additive constant in the objective function. Thus, for the linear program of the inner problem, we get the dual linear program

$$\min \quad \sum_{(v,w)\in A} \overline{c}(v,w)\cdot x_{vw} + \Gamma\cdot\lambda + \sum_{v\in V}\rho_v \tag{12}$$

$$\text{s.t.} \quad \lambda + \rho_v \geq \sum_{w\in V} \hat{c}(v,w)\cdot x_{vw} \quad v\in V \tag{13}$$

$$\rho_v \geq 0 \quad v\in V \tag{14}$$

$$\lambda \geq 0. \tag{15}$$

By inserting the dual of the inner problem into the outer problem, we get

$$\min_{x\in X,\, \rho\in\mathbb{R}_+^\eta,\, \lambda\in\mathbb{R}_+} \quad \sum_{(v,w)\in A} \overline{c}(v,w)\cdot x_{vw} + \Gamma\cdot\lambda + \sum_{v\in V}\rho_v \tag{16}$$

$$\text{s.t.} \quad \lambda + \rho_v \geq \sum_{w\in V} \hat{c}(v,w)\cdot x_{vw} \quad v\in V. \tag{17}$$

For all $v\in V$, we have $\rho_v \geq 0$, and hence constraints (17) can be rewritten as

$$\rho_v = \max\left\{0,\ \sum_{w\in V} \hat{c}(v,w)\cdot x_{vw} - \lambda\right\} \tag{18}$$

$$= \sum_{w\in V} \max\{0,\ \hat{c}(v,w) - \lambda\}\cdot x_{vw}, \tag{19}$$

since each $x\in X$ forms a Hamiltonian cycle, and thus for each $v\in V$ there exists exactly one $w\in V$ with $x_{vw}=1$. As in the case of Bertsimas and Sim (2003), all possibilities where one of constraints (17) becomes tight can be enumerated in order to find an optimal solution. We have such a constraint for every node $v\in V$, and for each one, we need to consider all possibilities making it tight. An optimal solution for instance $I_{\text{LTSP}^{\text{R}}}$ can thus be obtained by solving the η^2+1 problems

$$\min_{x\in X} \sum_{(v,w)\in A} \overline{c}(v,w)\cdot x_{vw} + \Gamma\cdot\lambda + \sum_{(v,w)\in A} \max\{0,\hat{c}(v,w)-\lambda\}\cdot x_{vw}, \tag{20}$$

with the values $\lambda\in\{\hat{c}(v,w)\,|\,(v,w)\in A\}\cup\{0\}$ and taking the minimum. Considering these values λ is sufficient, because for each $v\in V$, we have exactly one $w\in V$ with $x_{vw}=1$, and hence only one value $\hat{c}(v,w)$ is taken into account in each sum term of constraints (17). Since the value $\Gamma\cdot\lambda$ is only an additive constant in the objective function, an optimal solution for each of these η^2+1 problems can be found by solving a nominal TSP instance I'_{TSP} with the arc costs $c'(v,w)=\overline{c}(v,w)+\max\{0,\ \hat{c}(v,w)-\lambda\}$ for all $(v,w)\in A$. □

Note that the nominal TSP instances used to solve LTSP$^{\text{R}}$ according to Lemma 1 have a very specific cost structure, leaving their complexity unclarified

at a first glance. In Lemma 2, we demonstrate that for each of these resulting TSP instances, there exists an equivalent LTSP instance that can be solved efficiently.

Lemma 2 *Let I'_{TSP} be a* TSP *instance with the specific arc costs*

$$\begin{aligned} c'(v,w) &= \overline{c}(v,w) + \max\{0,\ \hat{c}(v,w) - \lambda\} \\ &= \max\{\overline{a}(v),\ b(w)\} + \max\{0,\ \max\{\overline{a}(v) + \hat{a}(v),\ b(w)\} - \max\{\overline{a}(v),\ b(w)\} - \lambda\} \end{aligned}$$

for all $(v,w) \in A$ and a constant $\lambda \geq 0$. Then, the LTSP *instance I''_{LTSP} with $a''(v) = \overline{a}(v) + \max\{0,\ \hat{a}(v) - \lambda\}$ and $b''(v) = b(v)$ for all $v \in V$ is equivalent to the* TSP *instance I'_{TSP}.*

Proof In the LTSP instance I''_{LTSP}, each arc $(v,w) \in A$ has the costs

$$c''(v,w) = \max\{\overline{a}(v) + \max\{0,\ \hat{a}(v) - \lambda\},\ b(w)\}.$$

It therefore remains to show that the arc costs $c'(v,w)$ of I'_{TSP} are the same as the arc costs $c''(v,w)$ of I''_{LTSP} for all $v, w \in V$, which we do using a case distinction.

- If $b(w) < \overline{a}(v)$, we have

$$\begin{aligned} c'(v,w) &= \overline{a}(v) + \max\{0,\ \overline{a}(v) + \hat{a}(v) - \overline{a}(v) - \lambda\} \\ &= \overline{a}(v) + \max\{0,\ \hat{a}(v) - \lambda\} \\ &= c''(v,w). \end{aligned}$$

- If $b(w) \geq \overline{a}(v)$ and $b(w) < \overline{a}(v) + \hat{a}(v)$, we have

$$\begin{aligned} c'(v,w) &= b(w) + \max\{0,\ \overline{a}(v) + \hat{a}(v) - b(w) - \lambda\} \\ &= \max\{b(w),\ b(w) + \overline{a}(v) + \hat{a}(v) - b(w) - \lambda\} \\ &= \max\{b(w),\ \overline{a}(v) + \hat{a}(v) - \lambda\} \\ &= \max\{b(w),\ \overline{a}(v) + \max\{0,\ \hat{a}(v) - \lambda\}\} \\ &= c''(v,w), \end{aligned}$$

  where $\hat{a}(v) - \lambda$ is equal to $\max\{0,\ \hat{a}(v) - \lambda\}$ if $\hat{a}(v) \geq \lambda$, and both $\max\{b(w),\ \overline{a}(v) + \hat{a}(v) - \lambda\}$ as well as $\max\{b(w),\ \overline{a}(v) + \max\{0,\ \hat{a}(v) - \lambda\}\}$ are equal to $b(w)$ if $\hat{a}(v) < \lambda$.
- If $b(w) \geq \overline{a}(v) + \hat{a}(v)$, we have

$$\begin{aligned} c'(v,w) &= b(w) + \max\{0,\ b(w) - b(w) - \lambda\} \\ &= b(w) \\ &= \max\{b(w),\ \overline{a}(v) + \max\{0,\ \hat{a}(v) - \lambda\}\} \\ &= c''(v,w). \end{aligned}$$

□

Recall that the nominal LTSP with η nodes is solvable in $O(\eta \cdot \log \eta)$, and thus Theorem 1 follows by combining Lemmas 1 and 2.

Theorem 1 *The* LTSP^{R} *with* η *nodes is solvable in* $O(\eta^3 \cdot \log \eta)$.

Proof According to Lemma 1, an optimal solution for an LTSP^{R} instance with η nodes can be computed by solving at most $\eta^2 + 1$ different nominal TSP instances with a specific cost structure, where according to Lemma 2 each one can be transformed to an equivalent nominal LTSP instance. The nominal LTSP with η nodes is solvable in $O(\eta \cdot \log \eta)$, resulting in an overall runtime of $O(\eta^3 \cdot \log \eta)$ to solve LTSP^{R}. □

Given Theorem 1, we immediately conclude Corollary 1 due to the equivalence of LTSP^{R} and $1\text{PRPP}^{\text{R}}_{\text{Bl}}$.

Corollary 1 $1\text{PRPP}^{\text{R}}_{\text{Bl}}$ *is solvable in* $O(n^3 \cdot \log n)$.

Overall, our results of the single I/O-point case from this section are summarized in Table 3 (both the equivalent problems and their complexities), grouped by the blocking and buffering variant as well as the nominal and robust case.

4 Multiple I/O-points

This section deals with the two general variants $\text{PRPP}^{\text{R}}_{\text{Bl}}$ and $\text{PRPP}^{\text{R}}_{\text{Bu}}$ with an arbitrary number of I/O-points. The two corresponding nominal variants PRPP_{Bl} and PRPP_{Bu} are already NP-hard as will be demonstrated next, showing immediately that this is also the case for their robust counterparts. Both PRPP_{Bl} and PRPP_{Bu} involve finding a feasible stacker crane tour as well as a partition of the pallets to the I/O-points (making it impractical to enumerate all possible solutions even on very small instances as there are totally $n! \cdot m^n$ different solutions). On the one hand, when ignoring the pallet processing times (i.e., $t(p) = 0$ for all $p \in P$), we obtain the *Retrieval Optimization Problem* (ROP) studied by Buckow et al (2024), which is a generalization of the TSP and thus already NP-hard. On the other hand, when ignoring the stacker crane travel times (i.e., $\tau[\ell_i, \ell_j] = 0$ for all $\ell_i, \ell_j \in L$), only the pallet processing times on the I/O-points are relevant, resulting in the NP-hard scheduling problem $\text{P}||\text{C}_{\max}$. This stresses the complexity of even the nominal variants with multiple I/O-points, given that they are generalizations of two strongly NP-hard problems.

Table 3 Summarized results of the single I/O-point case

	Nominal		Robust	
Variant	Equivalent problem	Complexity	Equivalent problem	Complexity
Blocking	$F2\|blocking\|\text{C}_{\max}$	$O(n \log n)$	LTSP^{R}	$O(n^3 \log n)$
Buffering	$F2\|\|\text{C}_{\max}$	$O(n \log n)$	Open	Open

Subsections 4.1 and 4.2 deal with the blocking and buffering case when having multiple I/O-points, respectively.

4.1 Blocking variant

We first reveal a mathematical model of the nominal blocking variant PRPP_{Bl} in Sect. 4.1.1, before presenting a dynamic programming algorithm for the worst-case evaluation of its robust counterpart $\text{PRPP}^{\text{R}}_{\text{Bl}}$ in Sect. 4.1.2. Finally, in Sect. 4.1.3, a mathematical model of $\text{PRPP}^{\text{R}}_{\text{Bl}}$ integrating the dynamic programming approach is presented.

4.1.1 Nominal case

Next, we present a formulation of the nominal problem PRPP_{Bl} as *mixed-integer linear program* (MIP), extending the formulation introduced by Buckow et al (2024) for the ROP by additionally incorporating the pallet processing times. The way the stacker crane travel times are modeled is based on the MIP formulation presented by Goerigk et al (2013) for a bus evacuation problem. Our model has four different variable types. First, the binary position variables x_{jki} receive the value 1 if pallet $p_j \in P$ is retrieved at position $k \in \{1, \dots, n\}$ at the I/O-point $\theta_i \in \Theta$, and 0 otherwise. Second, for $k = 1, \dots, n$, the variables d^k_{to} and d^k_{from} represent the stacker crane's durations to approach the kth pallet retrieved, and bringing it from there to its assigned I/O-point, respectively. Moreover, for $k = 0, \dots, n$, variable s_k indicates the starting time of the processing of the pallet at position k at its assigned I/O-point, with $k = 0$ meaning that no pallet is retrieved at all. Finally, variable $\text{C}_{\max}$ corresponds to the makespan of the resulting schedule.

$$(\text{MIP-PRPP}^{\text{R}}_{\text{Bl}})$$

$$\min \text{C}_{\max} \tag{21}$$

$$\text{s.t.} \sum_{j=1}^{n} \sum_{i=1}^{m} x_{jki} = 1 \quad k = 1, \dots, n \tag{22}$$

$$\sum_{k=1}^{n} \sum_{i=1}^{m} x_{jki} = 1 \quad j = 1, \dots, n \tag{23}$$

$$d^1_{\text{to}} = \sum_{j=1}^{n} \sum_{i=1}^{m} (\tau[\ell(\theta_{\text{Depot}}), \ell(p_j)] \cdot x_{j1i}) \tag{24}$$

$$d^k_{\text{to}} \geq \tau[\ell(\theta_i), \ell(p_j)] \cdot \left(\sum_{j'=1}^{n} x_{j',k-1,i} + \sum_{i'=1}^{m} x_{jki'} - 1 \right) \quad \begin{matrix} k = 2, \ldots, n; \\ j = 1, \ldots, n; \\ i = 1, \ldots, m \end{matrix} \tag{25}$$

$$d^k_{\text{from}} = \sum_{j=1}^{n} \sum_{i=1}^{m} (\tau[\ell(p_j), \ell(\theta_i)] \cdot x_{jki}) \quad k = 1, \ldots, n \tag{26}$$

$$\text{C}_{\max} \geq s_k + \sum_{j=1}^{n} \sum_{i=1}^{m} (\bar{t}(p_j) \cdot x_{jki}) \quad k = 1, \ldots, n \tag{27}$$

$$s_k \geq s_{k'} + \sum_{j=1}^{n} (\bar{t}(p_j) \cdot x_{jk'i}) - M \cdot \left(2 - \sum_{j=1}^{n} (x_{jki} + x_{jk'i}) \right) \quad \begin{matrix} k = 2, \ldots, n; \\ k' = 1, \ldots, k-1; \\ i = 1, \ldots, m \end{matrix} \tag{28}$$

$$s_k \geq s_{k-1} + d^k_{\text{to}} + d^k_{\text{from}} \quad k = 1, \ldots, n \tag{29}$$

$$x_{jki} \in \{0, 1\} \quad \begin{matrix} j, k = 1, \ldots, n; \\ i = 1, \ldots, m \end{matrix} \tag{30}$$

$$d^k_{\text{to}}, d^k_{\text{from}} \geq 0 \quad k = 1, \ldots, n \tag{31}$$

$$\text{C}_{\max} \geq 0 \tag{32}$$

$$s_k \geq 0 \quad k = 0, \ldots, n \tag{33}$$

Given the four variable types presented above, we have the MIP formulation (21)-(33). The objective function (21) minimizes the resulting makespan. Furthermore, constraints (22) and (23) ensure that the stacker crane travel sequence is properly defined, i.e., exactly one pallet is processed at each position and each pallet has to be processed exactly once. Constraint (24) defines the stacker crane travel duration to move from the depot I/O-point θ_{Depot} to the initial location of the first pallet retrieved. Constraints (25) describe the stacker crane travel durations to move from the previous I/O-point visited to the initial location of each of the remaining pallets. Analogously, constraints (26) define the stacker crane travel durations to move from each pallet to its assigned I/O-point. Constraints (27) force the makespan to be larger than all pallet completion times. Additionally, constraints (28) and (29) ensure that the pallet starting times are defined correctly by considering for each pallet both the completion time of the predecessor pallet at the same I/O-point and the stacker crane travel duration. Finally, the variable domains are defined by (30)-(33).

The big-M conditions ensure that constraints (28) are only active if both pallets involved are processed on the same I/O-point. Note that hereby all pallets previously processed on the same I/O-point are considered, instead of only using the direct

predecessors. While this results in some redundant constraints, we need less big-M conditions. Overall, only constraints (28) controlling the objective value contain the big-M parameter. Therefore, the objective value of each feasible solution is always an upper bound for parameter M, and it can be set to

$$M = \sum_{p \in P} \bar{t}(p) + \sum_{\ell_i, \ell_j \in L} \tau[\ell_i, \ell_j]. \tag{34}$$

4.1.2 Dynamic programming

For the worst-case evaluation of $\text{PRPP}^{\text{R}}_{\text{Bl}}$, we need to determine which pallets the adversary chooses to delay for a fixed stacker crane tour characterized by a pallet retrieval sequence π and an assignment α of pallets to I/O-points. For this purpose, we present a dynamic programming approach, where the key observation is that a pallet processing can only start after it is retrieved by the stacker crane and the processing of the previous pallet at the same I/O-point is completed. Due to the blocking condition, the stacker crane can only retrieve the current pallet after the processing of the pallet previously retrieved by the stacker crane (at any I/O-point) is started. From the resulting network with $O(n)$ states, we then create Γ copies to incorporate the potential pallet delays, leading to a dynamic programming approach with a runtime of $O(n \cdot \Gamma)$.

The dynamic program has states $z_{\gamma k} \in \mathbb{R}_+$ corresponding to the processing start time of the pallet retrieved at position $k \in \{0, \dots, n\}$ (where $k = 0$ means that no pallet is retrieved at all) in the case that at most $\gamma \in \{0, \dots, \Gamma\}$ pallets can be delayed. In addition, states $z_{\gamma, n+1}$ indicate the resulting makespan $\text{C}_{\max}$ if at most $\gamma \in \{0, \dots, \Gamma\}$ pallets can take their worst-case processing times. In the following, let set $Pred(n+1) \subseteq \{1, \dots, n\}$ contain the positions of all pallets that are processed last on any I/O-point. Moreover, for $k = 1, \dots, n$, we denote by $d(\pi_k)$ the stacker crane travel duration to retrieve pallet π_k (i.e., the time to move from its previous location to the initial location of pallet π_k and then to the assigned I/O-point $\alpha(\pi_k)$), which is fixed for given π and α. We initialize $z_{\gamma 0} = 0$ for all $\gamma = 0, \dots, \Gamma$, and the recursion is given as follows.

- For $\gamma = 0$ and $k = 1, \dots, n$, we have $z_{0k} = \max\{z_{0,k-1} + d(\pi_k),\ z_{0,pred(k)} + \bar{t}(\pi_{pred(k)})\}$, because no pallet is delayed, and the processing of pallet π_k can only start after it is retrieved by the stacker crane and its I/O-point predecessor is completed, respectively.
- For $\gamma = 1, \dots, \Gamma$ and $k = 1, \dots, n$, we additionally have to consider the case that the predecessor pallet $\pi_{pred(k)}$ is delayed, leading to

$$\begin{aligned} z_{\gamma k} = \max\{&z_{\gamma,k-1} + d(\pi_k), \\ &z_{\gamma,pred(k)} + \bar{t}(\pi_{pred(k)}), \\ &z_{\gamma-1,pred(k)} + \bar{t}(\pi_{pred(k)}) + \hat{t}(\pi_{pred(k)})\}. \end{aligned}$$

- For $\gamma = 0$ and $k = n+1$ we have $z_{0,n+1} = \max_{k \in Pred(n+1)}\{z_{0k} + \bar{t}(\pi_k)\}$, as the makespan C_{max} is defined as the largest completion time of a pallet processing.
- For $k = n+1$ and $\gamma = 1, \ldots, \Gamma$, we have $z_{\gamma,n+1} = \max_{k \in Pred(n+1)}\{z_{\gamma k} + \bar{t}(\pi_k), z_{\gamma-1,k} + \bar{t}(\pi_k) + \hat{t}(\pi_k)\}$ to additionally incorporate the case that a pallet processed last on an I/O-point is delayed.

Example 3 Recall the schedule displayed in Fig. 2 for the $PRPP^{R}_{Bl}$ instance with $\Gamma = 2$ from Tables 1 and 2. The corresponding dynamic programming process for the worst-case evaluation is shown in Fig. 7. The nodes represent the states, and the arcs correspond to the different values considered in the recursion. The stacker crane travel durations considered in the recursion are represented by dashed arcs, while the pallet processing times are illustrated by solid arcs. The dotted arcs thereby correspond to the case that the processing of a pallet is delayed. A longest path from

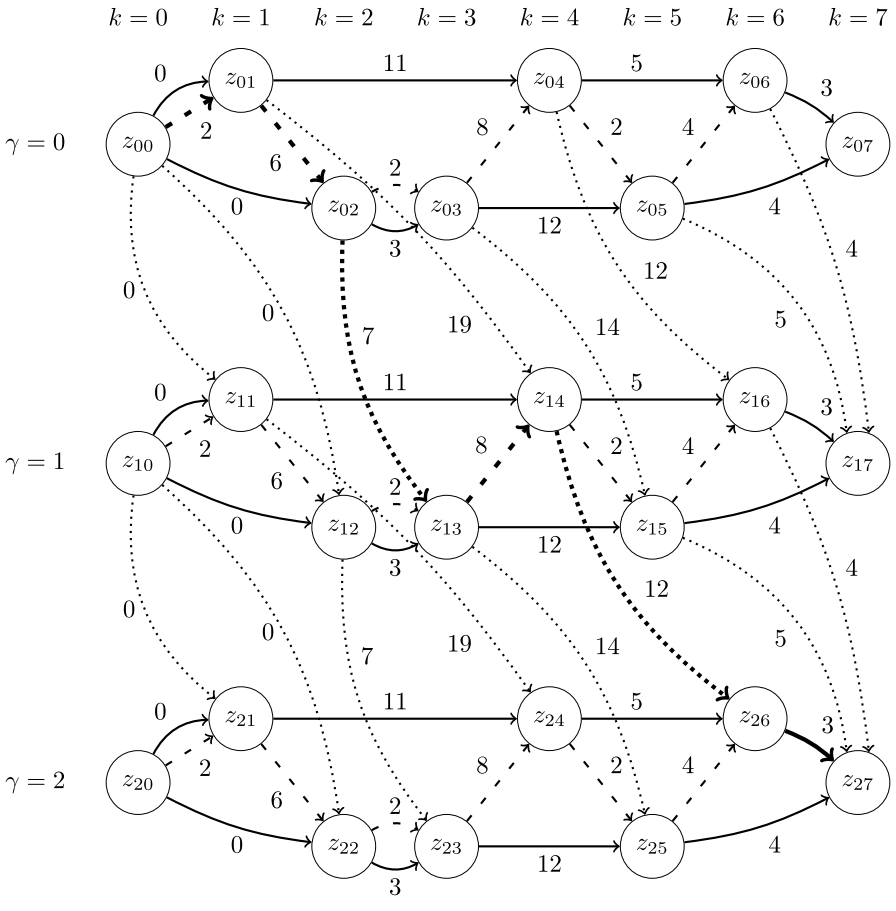


Fig. 7 Example of dynamic programming for $PRPP^{R}_{Bl}$

state z_{00} to z_{27} with a length of 38 is highlighted in bold, and the adversary would delay pallets p_2 and p_4.

4.1.3 Robust case

To formulate the robust variant $\text{PRPP}^{\text{R}}_{\text{Bl}}$ as MIP, we need to integrate the dynamic program for the worst-case evaluation as described in Subsection 4.1.2 into the MIP for the corresponding nominal variant PRPP_{Bl} as specified in Subsection 4.1.1. On the one hand, for all $\gamma = 0, \dots, \Gamma$ and $k = 0, \dots, n+1$, we insert a variable $z_{\gamma k}$ representing the corresponding state in the dynamic program. On the other hand, we remove variable $C_{\max}$ and the starting time variables s_k for $k = 1, \dots, n$, as these values are already represented by the inserted state variables.

$$(\text{MIP-PRPP}^{\text{R}}_{\text{Bl}})$$

$$\min z_{\Gamma, n+1} \tag{35}$$

$$\text{s.t. } (22) - (26)$$

$$z_{\gamma k} \geq z_{\gamma, k-1} + d^k_{\text{to}} + d^k_{\text{from}} \quad \begin{array}{l} \gamma = 0, \dots, \Gamma; \\ k = 1, \dots, n \end{array} \tag{36}$$

$$z_{\gamma k} \geq z_{\gamma k'} + \sum_{j=1}^{n} (\bar{t}(p_j) \cdot x_{jk'i}) - M \cdot \left(2 - \sum_{j=1}^{n} (x_{jki} + x_{jk'i}) \right) \quad \begin{array}{l} \gamma = 0, \dots, \Gamma; \\ k = 2, \dots, n; \\ k' = 1, \dots, k-1; \\ i = 1, \dots, m \end{array} \tag{37}$$

$$\begin{aligned} z_{\gamma k} \geq z_{\gamma-1, k'} &+ \sum_{j=1}^{n} ((\bar{t}(p_j) + \hat{t}(p_j)) \cdot x_{jk'i}) \\ &- M \cdot \left(2 - \sum_{j=1}^{n} (x_{jki} + x_{jk'i}) \right) \end{aligned} \quad \begin{array}{l} \gamma = 1, \dots, \Gamma; \\ k = 2, \dots, n; \\ k' = 1, \dots, k-1; \\ i = 1, \dots, m \end{array} \tag{38}$$

$$z_{\gamma, n+1} \geq z_{\gamma k} + \sum_{j=1}^{n} \sum_{i=1}^{m} (\bar{t}(p_j) \cdot x_{jki}) \quad \begin{array}{l} \gamma = 0, \dots, \Gamma; \\ k = 1, \dots, n \end{array} \tag{39}$$

$$z_{\gamma, n+1} \geq z_{\gamma-1, k} + \sum_{j=1}^{n} \sum_{i=1}^{m} ((\bar{t}(p_j) + \hat{t}(p_j)) \cdot x_{jki}) \quad \begin{array}{l} \gamma = 1, \dots, \Gamma; \\ k = \gamma, \dots, n \end{array} \tag{40}$$

$$x_{jki} \in \{0,1\} \quad \begin{matrix} j,k = 1,\dots,n; \\ i = 1,\dots,m \end{matrix} \tag{41}$$

$$d^k_{\text{to}}, d^k_{\text{from}} \geq 0 \quad k = 1,\dots,n \tag{42}$$

$$z_{\gamma k} \geq 0 \quad \begin{matrix} \gamma = 0,\dots,\Gamma; \\ k = 0,\dots,n+1 \end{matrix} \tag{43}$$

By the modifications described above, the MIP formulation (35)-(43) results, where constraints (22)-(26) are copied from the nominal MIP. The objective (35) minimizes the makespan by considering variable $z_{\Gamma,n+1}$ corresponding to the final state in the dynamic program. The dynamic programming recursion is implemented by constraints (36)-(40), where some cases of the maximum terms are summarized as they express the same. Constraints (36) ensure that the stacker crane travel durations $d(\pi_k) = d^k_{\text{to}} + d^k_{\text{from}}$ to retrieve pallet π_k for $k = 1,\dots,n$ are properly considered in the recursion.

The nominal and worst-case pallet processing times for the pallet predecessors at the I/O-points are taken into account by constraints (37) and (38), respectively. Due to the big-M conditions, these constraints are only active if the two considered jobs are processed on the same I/O-point. Similar to the nominal case, the objective value of each feasible solution serves as an upper bound for parameter M, since only constraints (37) and (38) which control the objective value contain big-M conditions, allowing it to be set to

$$M = \sum_{p \in P} (\bar{t}(p) + \hat{t}(p)) + \sum_{\ell_i, \ell_j \in L} \tau[\ell_i, \ell_j]. \tag{44}$$

Constraints (39) and (40) guarantee that the makespan for each value $\gamma \in \{1,\dots,\Gamma\}$ is defined as the largest completion time of a pallet. Finally, the variable domains are defined by constraints (41)-(43).

4.2 Buffering variant

This subsection tackles the buffering case with multiple I/O-points. Starting with a mathematical model of the nominal variant PRPP_{Bu} in Sect. 4.2.1, the worst-case evaluation for the robust counterpart $\text{PRPP}^{\text{R}}_{\text{Bu}}$ is solved by dynamic programming in Sect. 4.2.2. Eventually, Sect. 4.2.3 incorporates this dynamic programming approach into the mathematical model of the nominal case.

4.2.1 Nominal case

By adapting the MIP formulation (21)-(33) of PRPP_{Bl} as described in Subsection 4.1.1, we next derive the model (45)-(52) for PRPP_{Bu}. In this model, we keep the objective function (45), the constraints (22)-(26), as well as all four variable

types (49)-(52). Note that we also keep constraints (27) and (28) that are relevant for the makespan calculation and repeat them in (46) and (47) for better readability. Furthermore, we add constraints (48) to ensure that each pallet is only processed after the stacker crane has brought it to the assigned I/O-point, where no idle times are considered due to sufficient buffer space. Note that in constraints (52) defining the starting time variables, no variable s_0 is needed compared to inequalities (33), as we have constraints (48) rather than constraints (29). Again, since only constraints (47) controlling the objective value contain big-M conditions, parameter M can also be set according to equation (34).

$$(\text{MIP-PRPP}_{\text{Bu}})$$

$$\min \mathrm{C}_{\max} \tag{45}$$

$$\text{s.t. } (22) - (26)$$

$$\mathrm{C}_{\max} \geq s_k + \sum_{j=1}^{n} \sum_{i=1}^{m} (\bar{t}(p_j) \cdot x_{jki}) \quad k = 1, \ldots, n \tag{46}$$

$$s_k \geq s_{k'} + \sum_{j=1}^{n} (\bar{t}(p_j) \cdot x_{jk'i}) - M \cdot \left(2 - \sum_{j=1}^{n} (x_{jki} + x_{jk'i}) \right) \quad \begin{array}{l} k = 2, \ldots, n; \\ k' = 1, \ldots, k-1; \\ i = 1, \ldots, m \end{array} \tag{47}$$

$$s_k \geq \sum_{k'=1}^{k} (d_{\text{to}}^{k'} + d_{\text{from}}^{k'}) \quad k = 1, \ldots, n \tag{48}$$

$$x_{jki} \in \{0, 1\} \quad \begin{array}{l} j, k = 1, \ldots, n; \\ i = 1, \ldots, m \end{array} \tag{49}$$

$$d_{\text{to}}^{k}, d_{\text{from}}^{k} \geq 0 \quad k = 1, \ldots, n \tag{50}$$

$$\mathrm{C}_{\max} \geq 0 \tag{51}$$

$$s_k \geq 0 \quad k = 1, \ldots, n \tag{52}$$

4.2.2 Dynamic programming

In the following, we present a dynamic programming approach for the worst-case evaluation of $\text{PRPP}_{\text{Bl}}^{\text{R}}$, meaning to determine the pallets the adversary chooses to delay once the stacker crane movement characterized by π and α is fixed. Due to sufficient buffer space in the case of $\text{PRPP}_{\text{Bl}}^{\text{R}}$, a pallet $p \in P$ is ready for processing

as soon as the stacker crane has brought it to its assigned I/O-point, and we denote this point in time as the pallet's release date $r(p)$. Due to the release dates and the objective function of minimizing the makespan C_{max} that is determined only by one I/O-point, there always exists an optimal solution in which the adversary only delays pallets processed at one specific I/O-point. Therefore, the problem to compute the delayed pallets for given π and α in $PRPP^{R}_{Bl}$ can be handled for each I/O-point independently and by taking one with the largest resulting makespan C_{max}.

Due to the discussion above, a dynamic programming approach considering only one I/O-point at a time is sufficient for the worst-case evaluation. Recall that problem $PRPP_{Bl}$ with a single I/O-point is related to the two machine problem $F2||C_{max}$, where the stacker crane forms the first machine, and the single I/O-point corresponds to the second machine. To deal with the worst-case evaluation of problem $F2||C_{max}$ with budgeted uncertainty, Levorato et al (2022) present a dynamic programming approach. However, since they assume that the job processing times on both machines are uncertain, their approach is not directly applicable to our setting. In our case, the stacker crane travel times are certain, and only the pallet processing times at the I/O-points are subject to uncertainty. We therefore present an adapted variant of their dynamic programming approach to tackle our setting.

In our approach, we have states $z_{\gamma k} \in \mathbb{R}_+$ that indicate the worst-case makespan when at most $\gamma \in \{0, \dots, \Gamma\}$ pallets can be delayed, and only the pallets processed at positions $0, \dots, k$ on the current machine are taken into account. Note that for a given γ, the number of states can be reduced by considering only the values $k \in \{\gamma, \dots, n - \Gamma + \gamma\}$ because the remaining states are never part of a longest path in the resulting network. Our initialization is simply $z_{00} = 0$, since no pallet is processed for $\gamma = 0$ and $k = 0$. In the recursion, we distinguish the following three cases.

- For $\gamma = 0$ and $k = 1, \dots, n - \Gamma$, we have $z_{0k} = \max\{r(\pi_k), z_{0,k-1}\} + \bar{t}(\pi_k)$, because no pallet can be delayed at all, and we need to consider both the release date $r(\pi_k)$ of the pallet at the kth position as well as the completion time $z_{0,k-1}$ of the pallet at the previous position $k - 1$.
- For $\gamma = 1, \dots, \Gamma$ and $k = \gamma$, the kth pallet can always be assumed to be delayed, and we thus have $z_{\gamma k} = \max\{r(\pi_k), z_{\gamma-1,k-1}\} + \bar{t}(\pi_k) + \hat{t}(\pi_k)$.
- For $\gamma = 1, \dots, \Gamma$ and $k = \gamma + 1, \dots, n - \Gamma + \gamma$, we have to consider the two cases that the current pallet is delayed or not and take the maximum, leading to

$$\begin{aligned} z_{\gamma k} &= \max\{\max\{r(\pi_k),\, z_{\gamma,k-1}\} + \bar{t}(\pi_k),\, \max\{r(\pi_k),\, z_{\gamma-1,k-1}\} + \bar{t}(\pi_k) + \hat{t}(\pi_k)\} \\ &= \max\{r(\pi_k) + \bar{t}(\pi_k) + \hat{t}(\pi_k),\, z_{\gamma,k-1} + \bar{t}(\pi_j),\, z_{\gamma-1,k-1} + \bar{t}(\pi_j) + \hat{t}(\pi_k)\}. \end{aligned}$$

The presented dynamic programming approach has the overall runtime $O(\Gamma \cdot n)$, as we have states $z_{\gamma k}$ for $\gamma = 0, \dots, \Gamma$ and $k = \gamma, \dots, n - \Gamma + \gamma$. Since we have to apply the dynamic programming algorithm for each I/O-point independently and taking the maximum, the worst-case evaluation for $PRPP^{R}_{Bu}$ has the runtime $O(\Gamma \cdot n \cdot m)$.

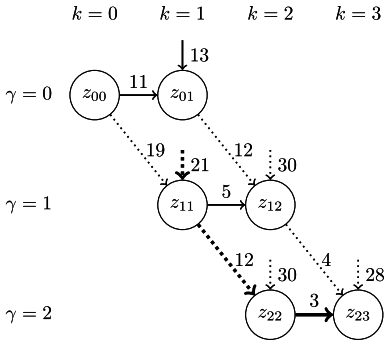


Fig. 8 Example of dynamic programming for $\text{PRPP}^{\text{R}}_{\text{Bu}}$

Example 4 Reconsider the example instance of $\text{PRPP}^{\text{R}}_{\text{Bu}}$ with $\Gamma = 2$ shown in Tables 1 and 2, and the corresponding nominal schedule depicted in Fig. 3. The dynamic programming process for the worst-case evaluation of I/O-point θ_1 is shown in Fig. 8, where pallets p_1, p_4 and p_6 are processed at the positions $k = 1$, 2 and 3, respectively. The nodes correspond to the states, and the arcs represent the different possibilities due to the recursion, with the arcs above each node indicating the case that the processing of the corresponding pallet starts at its release date. The dotted arcs indicate when a processing of a pallet is delayed, the arcs chosen by the recursion are highlighted in bold, and the resulting longest path has a length of 36. By retracing the recursion, it can be determined that pallets p_1 and p_4 at positions $k = 1$ and 2 are delayed in the corresponding solution.

4.2.3 Robust case

To also formulate the robust variant $\text{PRPP}^{\text{R}}_{\text{Bu}}$ as MIP, we need to alter the MIP (45)-(52) from Subsection 4.2.1 of the corresponding nominal variant PRPP_{Bu} by incorporating the dynamic programming approach presented in Subsection 4.2.2 for the worst-case evaluation. Recall that this algorithm can be performed for each I/O-point independently, and hence for all $\gamma = 0, \dots, \Gamma$, $k = \gamma, \dots, n - \Gamma + \gamma$ and $i = 1, \dots, m$, we insert variables $z_{\gamma ki}$ corresponding to state $z_{\gamma k}$ of the ith I/O-point in the dynamic programming scheme. At the time when creating the MIP, we do not know which pallet will be processed at which I/O-point, so in the dynamic programming algorithm, the states have to be created for all possible pallets, where some states later become disabled by big-M conditions. As in the previous models, these big-M conditions only need to be involved in constraints that control the objective value, meaning parameter M can be chosen as stated in equation (44). Moreover, we replace the starting time variables s_k in the nominal MIP by variables r_k corresponding to the release dates of the kth pallet retrieved for all $k = 1, \dots, n$. All other variables remain the same as in the nominal case, and the basic constraints (22)-(26) are copied from the corresponding nominal MIP.

By applying the changes described, the MIP (53)-(64) results for the robust case. Again, note that some cases of the maximum terms in the recursion express the same and they are hence summarized in the MIP. It is ensured by constraints (54) that the

makespan is at least as large as the worst-case completion time of each single I/O-point. In addition, it is guaranteed by constraints (55) and (56) that the release dates are respected when no pallet is delayed at all or if the current pallet is the first pallet to be delayed, respectively. Furthermore, constraints (57) represent the case that the pallet at position k takes its nominal processing time, while constraints (58) regard the case that it is delayed. Constraints (59) make sure that the release dates are properly defined as the time when the stacker crane has brought the current pallet to its designated I/O-point. Eventually, the variable domains are defined by (60)-(64).

$$(\text{MIP-PRPP}^{\text{R}}_{\text{Bu}})$$

$$\min \mathrm{C}_{\max} \tag{53}$$

$$\text{s.t. } (22) - (26)$$

$$\mathrm{C}_{\max} \geq z_{\Gamma ni} \quad i = 1, \ldots, m \tag{54}$$

$$z_{0ki} \geq r_k + \sum_{j=1}^{n} (\bar{t}(p_j) \cdot x_{jki}) - M \cdot \left(1 - \sum_{j=1}^{n} x_{jki}\right) \quad \begin{array}{l} k = 1, \ldots, n - \Gamma; \\ i = 1, \ldots, m \end{array} \tag{55}$$

$$\begin{aligned} z_{\gamma ki} \geq r_k + \sum_{j=1}^{n} ((\bar{t}(p_j) + \hat{t}(p_j)) \cdot x_{jki}) \\ - M \cdot \left(1 - \sum_{j=1}^{n} x_{jki}\right) \end{aligned} \quad \begin{array}{l} \gamma = 1, \ldots, \Gamma; \\ k = \gamma, \ldots, n - \Gamma + \gamma; \\ i = 1, \ldots, m \end{array} \tag{56}$$

$$z_{\gamma ki} \geq z_{\gamma, k-1, i} + \sum_{j=1}^{n} (\bar{t}(p_j) \cdot x_{jki}) \quad \begin{array}{l} \gamma = 0, \ldots, \Gamma; \\ k = \gamma + 1, \ldots, n - \Gamma + \gamma; \\ i = 1, \ldots, m \end{array} \tag{57}$$

$$z_{\gamma ki} \geq z_{\gamma - 1, k-1, i} + \sum_{j=1}^{n} ((\bar{t}(p_j) + \hat{t}(p_j)) \cdot x_{jki}) \quad \begin{array}{l} \gamma = 1, \ldots, \Gamma; \\ k = \gamma, \ldots, n - \Gamma + \gamma; \\ i = 1, \ldots, m \end{array} \tag{58}$$

$$r_k \geq \sum_{k'=1}^{k} (d^{k'}_{\text{to}} + d^{k'}_{\text{from}}) \quad k = 1, \ldots, n \tag{59}$$

$$x_{jki} \in \{0, 1\} \quad \begin{array}{l} j, k = 1, \ldots, n; \\ i = 1, \ldots, m \end{array} \tag{60}$$

$$d^k_{\text{to}}, d^k_{\text{from}} \geq 0 \quad k = 1, \ldots, n \tag{61}$$

$$C_{\max} \geq 0 \tag{62}$$

$$z_{\gamma ki} \geq 0 \quad \begin{array}{l} \gamma = 0, \ldots, \Gamma; \\ k = \gamma, \ldots, n - \Gamma + \gamma; \\ i = 1, \ldots, m \end{array} \tag{63}$$

$$r_k \geq 0 \quad k = 1, \ldots, n \tag{64}$$

5 Computational results

In this section, different aspects of the PRPP are evaluated experimentally. First, the test instances used in our experiments are described in Subsection 5.1. Afterwards, in Subsection 5.2, the benefits of the new integrated models for both PRPP_{Bl} and PRPP_{Bu} are evaluated by comparing them with the relaxations of the stacker crane travel times and pallet processing times, respectively. Similarly, Subsection 5.3 deals with the benefits of the robust models compared to the nominal models. In Subsection 5.4, we investigate the differences between variants PRPP_{Bl} and PRPP_{Bu}, i.e., the benefits of having a buffer at each I/O-point. Note that Subsections 5.2-5.4 mainly serve to evaluate the new models themselves, while their computational limits and the instance sizes they are able to solve can be found in Appendix A. Eventually, some heuristics and their computational results are revealed in Subsection 5.5 to also handle larger problem instances.

Our algorithms were coded in the C++ programming language, and all experiments were conducted on an Intel Core i9-10920X 3.5GHz machine with 64 Bit Ubuntu 20.04 LTS and 64GB RAM. To solve the four mathematical models MIP-PRPP_{Bl}, MIP-$\text{PRPP}^{\text{R}}_{\text{Bl}}$, MIP-$\text{PRPP}_{\text{Bu}}$ and MIP-$\text{PRPP}^{\text{R}}_{\text{Bu}}$ developed in Sect. 4, we used CPLEX 20.1, which was warm-started with a solution determined by a heuristic. We present and evaluate multiple heuristics for the PRPP in Subsection 5.5. Especially on smaller instances, an iterative improvement procedure turned out to be effective in determining good solutions in a small amount of computing time. Hence, we use this iterative improvement procedure for the warm-start of CPLEX. Note that this heuristic can also handle the nominal case by setting $\Gamma = 0$.

The objective values resulting from the iterative improvement procedure were also used to initialize the big-M parameter in the four mathematical models developed in Sect. 4. A time limit of one hour per run was set in all experiments involving CPLEX. The experiments with CPLEX were multi-threaded, allowing the use of up to ten cores simultaneously, while all other experiments were single-threaded. All test instances and the raw data of our results are available at http://www2.informatik.uos.de/kombopt/data/rop/.

5.1 Test instances

For our computational study, we randomly generated a total of 270 test instances $I(n, m)$ with different numbers of pallets and I/O-points. The coordinates of the locations corresponding to the pallets and I/O-points are randomly sampled as integers in a square area having an edge length of 100. Travel times between these locations are calculated by using the Chebyshev metric, i.e., taking the maximum of the horizontal and vertical distance. In addition, pallet processing times are determined by uniformly choosing two integers in the interval [1, 200], with the smaller integer corresponding to the nominal processing time and the difference between these two integers being the potential processing time delay. Moreover, the numbers of pallets used are $n \in \{5, 6, 7, 8, 9, 10, 20, 50, 100\}$, and the numbers of I/O-points chosen are $m \in \{1, 2, 3\}$, having particularly many instances with few pallets for comparisons needing optimal solutions. The uncertainty budget was set to $\Gamma = \max\{3, \lceil n/10 \rceil\}$, implying that $\Gamma = 3$ holds for all generated instances with $n \leq 20$ pallets, while the uncertainty budget increases with the number of pallets for the remaining instances. Ten replications were made for each combination of parameters n and m.

Our generated instances include the practical parameter values relevant to the aforementioned company, where nearly $n = 100$ pallets have to be retrieved from a warehouse with a total of $m = 3$ I/O-points in order to prepare one shift. As is common in warehouses with AS/RS, their automated stacker crane can move independently in horizontal and vertical directions, and its travel costs are hence also based on the Chebyshev metric.

5.2 Benefits of new integrated models

As already mentioned above, the PRPP combines the ROP and $P||C_{max}$ by taking into account both the stacker crane travel times and the pallet processing times. In our first experiment, we evaluated the benefits of our two integrated mathematical models MIP-$PRPP_{Bl}$ and MIP-$PRPP_{Bu}$ presented in Sects. 4.1.1 and 4.2.1 compared to ignoring the pallet processing times or the stacker crane travel times, resulting in ROP and $P||C_{max}$ instances, respectively. On the one hand, the ROP instances were solved in the same way as by Buckow et al (2024), i.e., by using their transformation of the ROP to the TSP and then applying their branch-and-cut solver to it. On the other hand, the $P||C_{max}$ instances were solved by using a simple MIP model. Note that each ROP solution already fully determines a PRPP solution, while a $P||C_{max}$ solution only yields an assignment α of pallets to I/O-points. As shown by Buckow et al (2024), the problem of determining an optimal pallet retrieval sequence for a fixed assignment α of pallets to I/O-points is already NP-hard when only considering stacker crane travel times. Thus, we heuristically determined a pallet retrieval sequence π in the case of $P||C_{max}$ by lexicographically sorting the pallets by their starting times in the relaxation and by their indices.

Only small instances with $n \leq 10$ were included in this experiment in order to be able to obtain nearly optimal solutions. All instances with $n \leq 8$ could be verified as

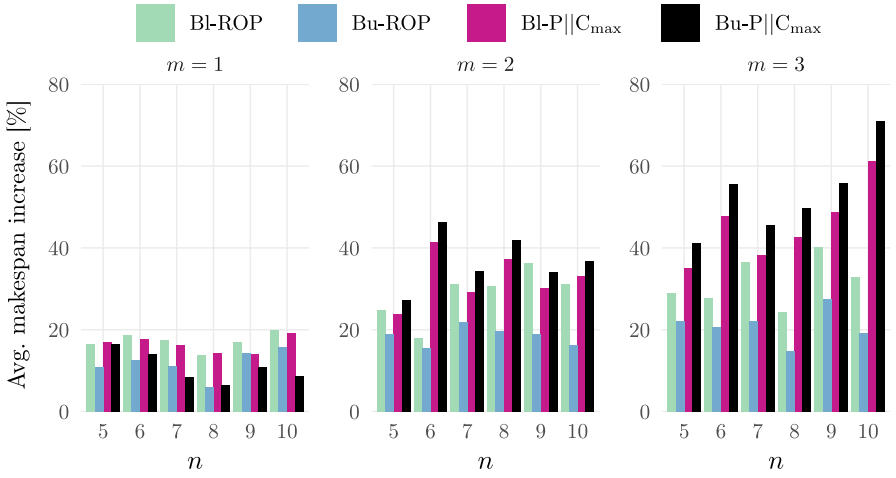


Fig. 9 Average percentage makespan increases of the relaxations compared to the integrated models, differentiated by parameters n and m

optimally solved within the time limit of one hour, while some instances with $n \geq 9$ could not be verified within the time limit. Further details on the integrated MIP results can be found in Appendix A. Remember that enumerating all possible solutions is no viable alternative to the MIP models even for small instances, since we have $n! \cdot m^n$ different solutions. Moreover, all resulting instances of ROP and $P||C_{max}$ could be solved optimally in a negligible amount of computing time. In the following, we call Bl-ROP and Bu-ROP the relaxations of $PRPP_{Bl}$ and $PRPP_{Bu}$ where only the stacker crane travel times are considered. In contrast, let Bl-$P||C_{max}$ and Bu-$P||C_{max}$ be the relaxations where only the pallet processing times are taken into account. The resulting average percentage makespan increases of the relaxations compared to the integrated mathematical models are shown in Fig. 9, differentiated by parameters n and m, that is, the higher the value, the bigger the benefit of the integrated approach.

Figure 9 discloses that the results are generally similar in the blocking and buffering case. Moreover, even with a single I/O-point ($m = 1$), there are materially benefits of the integrated models, predominantly in the double-digit percentage range. These benefits become even larger when having multiple I/O-points ($m = 2$ or $m = 3$), reaching makespan increases of up to 70%. However, the results show no clear impact of the number n of pallets. Furthermore, the integrated models of PRPP yield a little bit larger benefits in relation to $P||C_{max}$ compared to ROP. Overall, these results highlight the importance of considering both the stacker crane travel times and the pallet processing times to solve the PRPP properly. Since our integrated models consider both, they are required to tackle the PRPP appropriately.

5.3 Benefits of robust models

In a second experiment, we investigated the benefits of the robust integrated mathematical models MIP-PRPP$^{R}_{Bl}$ and MIP-PRPP$^{R}_{Bu}$ presented in Subsections 4.1.3 and 4.2.3 compared to the nominal integrated models MIP-PRPP$_{Bl}$ and MIP-PRPP$_{Bu}$ presented in Subsections 4.1.1 and 4.2.1. Only small instances with $n \leq 10$ pallets are included in this experiment in order to mainly compare optimal results. Recall that an uncertainty budget of $\Gamma = 3$ was assumed by default in all instances with $n \leq 10$ pallets. All instances with $n \leq 8$ pallets could be verified as optimally solved within the one hour time limit, whereas some instances with $n \geq 9$ pallets were not verified (see Appendix A for more details). Again, recall that enumerating all possible solutions is not a practical alternative to the MIP models due to the large solution space. Note that fewer instances were verified as optimally solved by the robust models than with the nominal models. To properly compare the nominal and robust results, the nominal solutions resulting from the mathematical models MIP-PRPP$_{Bl}$ and MIP-PRPP$_{Bu}$ were evaluated by using the dynamic programming algorithms for the worst-case evaluation presented in Subsections 4.1.2 and 4.2.2.

Figure 10 shows the average percentage makespan increases of the nominal models compared to the robust models, distinguished by the blocking and buffering case as well as the instance parameters n and m. These makespan increases refer to the percentage gaps of the objective values of the nominal solutions after applying the dynamic programming algorithm compared to the robust objective values. The results shown in Fig. 10 reveal that the benefits of the robust models are quite high, with makespan increases of up to 30% in some combinations. The more I/O-points we have, the larger the makespan increases are, ranging from single-digit percentages for $m = 1$ I/O-point to double-digit makespan increases for $m = 2$ or $m = 3$ I/O-points. In contrast, the number of pallets n apparently has no clear influence on the

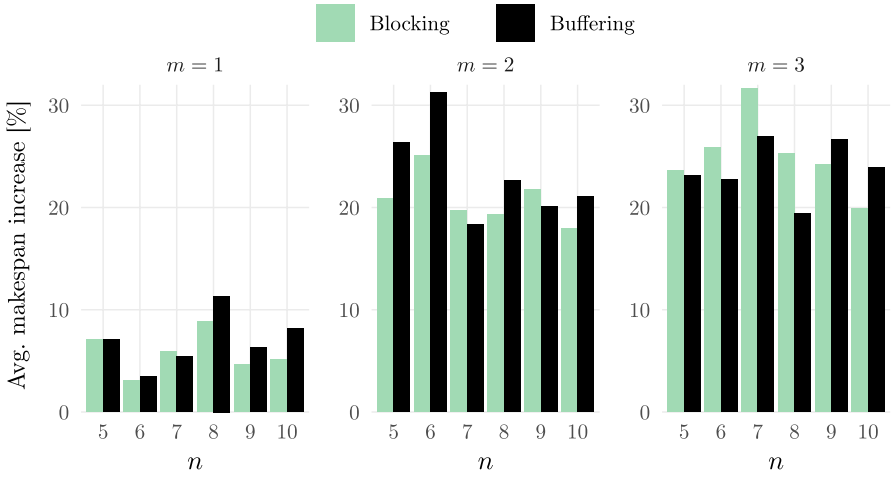


Fig. 10 Average percentage makespan increases of the nominal models compared to the robust models, separated by the blocking and buffering variant and by parameters n and m

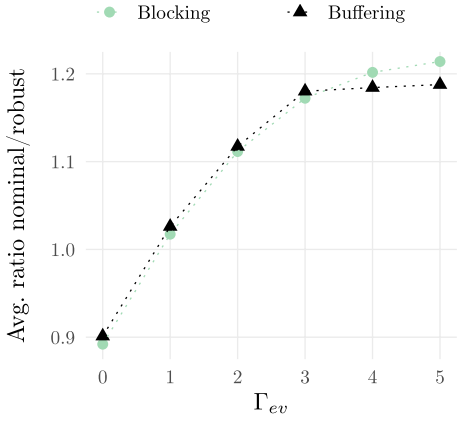


Fig. 11 Average ratios of the nominal ($\Gamma_{al} = 0$) to the robust ($\Gamma_{al} = 3$) makespans, separated by the blocking and buffering variant, and evaluated for different Γ_{ev}

results. Furthermore, the makespan increases are basically the same in the blocking and buffering case, apart from a few differences in some combinations. Since the results depicted in Fig. 10 disclose that the robust models for the PRPP with budgeted uncertainty lead to considerably better results than the nominal models, we conclude that the robust case needs its own specific models to be tackled appropriately.

To also investigate how sensitive our robust MIP models react to changes in the uncertainty budget Γ, we conducted a sensitivity analysis in another experiment. In that experiment, we distinguish between two types of Γ for the sake of clarity. On the one hand, we have Γ_{al} which is actually used in the algorithms to solve the models, and on the other hand, we have Γ_{ev} for the worst-case evaluation of the resulting solutions using the dynamic programming algorithms. For all instances with $n \leq 10$ pallets, we evaluated the solutions obtained by both the nominal models ($\Gamma_{al} = 0$) and the robust models ($\Gamma_{al} = 3$) for different values of $\Gamma_{ev} \in \{0, 1, 2, 3, 4, 5\}$. Figure 11 depicts for both the blocking and buffering variants the average ratios of the nominal to the robust objective values, differentiated by Γ_{ev}. Ratios smaller than 1.0 mean that the nominal models obtained better solutions, while ratios larger than 1.0 indicate that the robust models generated superior solutions.

It can be seen in Fig. 11 that the ratios are below 1.0 only for $\Gamma_{ev} = 0$, meaning that the solution quality of the nominal models only exceeds the solution quality of the robust models if there is no uncertainty at all. Moreover, the solution quality of the nominal and robust models is nearly the same for $\Gamma_{ev} = 1$, since the ratios are only a little above 1.0. The ratios reach more than 1.1 for $\Gamma_{ev} = 2$, and continue to rise even more with growing Γ_{ev}, with the ratios rising faster in the blocking case, while the buffering case reaches saturation earlier. The benefits of the blocking model thus seem to be moderately larger than those of the buffering model, as the ratios are generally a bit larger. However, the buffering model seems to be slightly more resilient to changes in Γ_{ev}, as the increase in the ratios becomes rather small at a certain point. Overall, the robust models yield a stable gain compared to the nominal models for all different values of Γ_{ev} except 0, showing that our robust models are quite insensitive to changes in the uncertainty budget Γ.

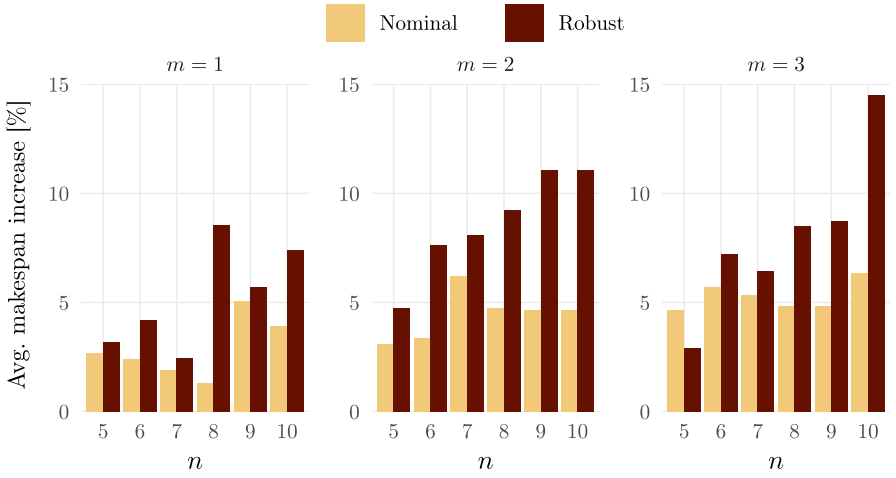


Fig. 12 Average percentage makespan increases of the blocking compared to the buffering variant, separated by the nominal and robust case and by parameters n and m

5.4 Benefits of buffers

In the next experiment, we examined the benefits of having a buffer at each I/O-point by comparing the blocking variant PRPP_{Bl} with the buffering variant PRPP_{Bu}. Note that for any given instance, the optimal makespan in the buffering variant is at least as good as in the blocking variant, since the buffering variant relaxes the blocking constraint. The benefits of the buffer are measured by the resulting average percentage makespan increases of the blocking compared to the buffering variant, thus corresponding to the required makespan increases when having no buffer at all. Note that the higher the makespan increases, the larger the benefits of having buffers. To calculate these makespan increases, we used the results obtained from both the nominal models $\text{MIP-PRPP}_{\text{Bl}}$ and $\text{MIP-PRPP}_{\text{Bu}}$ (see Sects. 4.1.1 and 4.2.1) as well as the robust models $\text{MIP-PRPP}^{\text{R}}_{\text{Bl}}$ and $\text{MIP-PRPP}^{\text{R}}_{\text{Bu}}$ (see Sects. 4.1.3 and 4.2.3).

For the nominal and robust case, the average percentage makespan increases are displayed in Fig. 12 for all combinations of parameters n and m, separated by the nominal and robust case. It can be seen that most makespan increases are in the single-digit percentage range (remember that the larger the makespan increases, the bigger the benefits of the buffers), showing that having a buffer considerable reduces the makespan. These makespan increases become even larger in the robust case compared to the nominal case, caused by the fact that solutions for the blocking variant are more vulnerable to increases in the pallet processing times as these cannot be buffered. Generally, the results show that it is very worthwhile to use a buffer, particularly in the robust case. This shows practitioners that buffers should be installed at the I/O-points whenever possible.

In preliminary experiments, we also analyzed the solution structure in the case of PRPP_{Bu}, showing that the buffer space is actually used regularly. For the smaller instances with $n \leq 10$ pallets, one or two pallets are often stored at one

I/O-point simultaneously. However, we also analyzed heuristic solutions for the larger instances with $n \geq 50$ pallets, where sometimes up to 20 pallets are stored at one I/O-point simultaneously. Moreover, the better the solution quality is, the more often the buffer tends to be used, explaining why the resulting makespans for PRPP_{Bu} are on average much lower than for PRPP_{Bl}.

5.5 Heuristics

Preliminary tests have shown that CPLEX without warm-start takes up to a minute to set up the mathematical models for the larger instances with $n = 100$ pallets, meaning it cannot quickly find even feasible solutions. In order to also solve larger instances properly in a reasonable amount of computing time, we implemented some heuristics in the last experiment. These heuristics can be used for both $\text{PRPP}^{\text{R}}_{\text{Bl}}$ and $\text{PRPP}^{\text{R}}_{\text{Bu}}$, as they work with the same solution representation consisting of both the pallet retrieval sequence π and the assignment α of pallets to I/O-points. Moreover, even if these heuristics are designed to handle the robust case, note that they can also be used to tackle the nominal case simply by setting $\Gamma = 0$. Overall, we implemented the following four heuristics.

- **Greedy heuristic (GD)**. Starting with an empty solution, the pallets are greedily appended to it by checking all pairs of I/O-points and remaining pallets. In each step, a pair is chosen that leads to the smallest makespan of the resulting partial solution. Such partial solutions are evaluated by applying the dynamic programming algorithms presented in Sect. 4.
- **Random heuristic (RA)**. By randomly choosing both the pallet retrieval sequence π and the assignment α of pallets to I/O-points, solutions are sampled until a computation time limit is reached. Finally, the best of all sampled solutions is returned.
- **Iterative improvement (II)**. Initially, a feasible solution is determined using the greedy heuristic presented above. This initial solution is then iteratively improved by searching two neighborhoods one after the other until a local optimum is reached. In the first neighborhood, two pallets in the retrieval sequence are swapped, and new best I/O-points are assigned to both swapped pallets by checking all possibilities. In the second neighborhood, three pallets in the retrieval sequence are reordered in a best possible way by checking all six permutations of these three pallets and also assigning them to best I/O-points. In each iteration, the second neighborhood is searched only if the first neighborhood yields no further improvements. This procedure terminates prematurely even if no local optimum has been reached after a given computation time limit.
- **Tabu search (TS)**. The tabu search also starts with an initial solution generated by the greedy heuristic. This solution is gradually altered using the first neighborhood described in the iterative improvement algorithm, i.e., two pallets are swapped in the retrieval sequence and new best I/O-points are chosen for them.

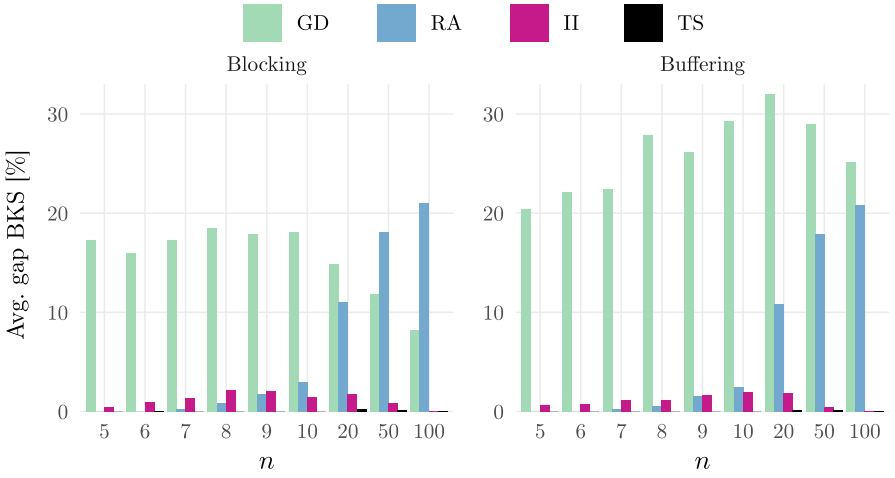


Fig. 13 Average percentage gaps of the heuristics to BKS values, differentiated by the blocking and buffering variant and by parameter n

The first improving neighbor solution found is always taken, and if no improving neighbor solution exists, a best neighbor solution is taken for which the transition to it is not marked as tabu in the tabu list. The transition to a neighbor solution is considered as tabu if the triple consisting of the two positions of both pallets swapped and the objective value of the generated neighbor solution is already contained as an entry in the tabu list. New tabu list entries are appended to the end of the tabu list, and if hereby the maximum tabu list size $\psi = 500$ (a value that appeared effective in preliminary experiments) is exceeded, the first entry in the tabu list is removed. The tabu search terminates after a given computation time limit and returns the best solution found so far.

Figure 13 depicts the results of all four heuristics for the robust problem variants $\text{PRPP}^{\text{R}}_{\text{Bl}}$ (left) as well as $\text{PRPP}^{\text{R}}_{\text{Bu}}$ (right) and different numbers of pallets n. These results are reported by average percentage gaps to *best known solutions* (BKS) of the respective instances, which correspond to the best solutions found during all experiments performed, including our tests of the robust models MIP-$\text{PRPP}^{\text{R}}_{\text{Bl}}$ and MIP-$\text{PRPP}^{\text{R}}_{\text{Bu}}$ presented in Sects. 4.1.3 and 4.2.3. In particular, note that the BKS values for all instances with $n \leq 8$ correspond to optimal objective values. Furthermore, a computing time limit of one minute per run was imposed in this experiment, even if heuristics GH and II required much smaller computing times on most instances.

Overall, Fig. 13 reveals that the differences in the heuristic results between the blocking and buffering case are generally rather small. Nonetheless, it is evident that heuristic TS performs best, as its average gaps are well below 1% in all tested combinations. What is particularly remarkable about these results is that heuristic TS reached gaps of almost 0% for all instances with $n \leq 10$, showing that it solved these small instances nearly optimally. Heuristic II performs second best, having average gaps under 3% in all tested combinations. The results of heuristic RA are less

consistent, there are very small gaps for instances with $n \leq 8$, while for the large instances there are considerable gaps of up to 20%. The construction heuristic GD performs worst with mostly double-digit percentage gaps even for small instances. However, this is mainly because construction heuristic GD required much less computing time than the other heuristics, taking less than one second on any given instance.

Since heuristics II and TS have very small gaps in all tested combinations, the PRPP seems to be well solvable on instances for larger, practically relevant sizes of up to $n = 100$ pallets as in the case of the aforementioned company. Although knowing that the pallet processing times are still relevant, the company concentrates on the stacker crane travel times in their current planning. Nevertheless, discussions with practitioners at the company revealed that they underestimated the optimization potential gained by additionally considering the pallet processing times.

Note that preliminary tests showed that heuristics II and TS are much superior in solving large instances compared to the MIP models presented in Sect. 4 and using CPLEX without warm-start. As already mentioned above, heuristic II is hence used to warm-start the MIP models. For several larger instances with $n \geq 50$ pallets, heuristic II did not reach a local optimum within the one minute time limit. Nevertheless, for the smaller instances with $n \leq 10$ pallets which were also used to test the MIP models, heuristic II required very little computing time to reach a local optimum, taking less than one second on any of these instances. As heuristic II is able to find good solutions on the smaller instances in a very short amount of computing time, it appears to be best suited for MIP warm-starts.

While the MIP models were only capable of solving some small instances with at most ten pallets, preliminary experiments also revealed that even heuristics II and TS reached their computational limits for instances with more than 100 pallets. For such large instances, the search for an improving neighbor solution takes considerable computing times, resulting in very few (if any at all) iterations being completed in a reasonable amount of time, highlighting how difficult the PRPP is to solve.

6 Conclusions

In this paper, we studied and evaluated different variants of the PRPP, an integrated problem considering both the retrieval and processing of pallets in warehouses with multiple I/O-points. We differentiate between the blocking and buffering problem variant, depending on whether there is sufficient buffer space at the I/O-points for temporarily storing pallets or not. Additionally, to protect against uncertainties in pallet processing times, we apply robust optimization with budgeted uncertainty sets.

When having just a single I/O-point, the blocking and the buffering variants are equivalent to well-known two-machine flow-shop scheduling problems, allowing to solve the single I/O-point case of these two variants without robustness in polynomial time. Moreover, we proved that even the robust single I/O-point blocking

variant can be solved in polynomial time, while the complexity of the robust buffering variant with only one I/O-point remains an open question. For the general case with an arbitrary number of I/O-points, mathematical models were presented for both the blocking and the buffering variant. Furthermore, dynamic programming algorithms for the worst-case evaluation of the variants with uncertainty were developed, which were then integrated into the nominal mathematical models to obtain robust models.

The computational study disclosed that our new integrated models for the PRPP achieved much better results than existing models for pallet retrieval optimization or parallel machine scheduling problems. This shows that both the stacker crane travel times as well as the pallet processing times must be taken into account to solve the PRPP adequately. Compared to the nominal integrated models, our robust models turned out to be considerably better at hedging against uncertainties in the pallet processing times. The comparison between the blocking and buffering variants shows that in both the nominal and robust case the makespan can be reduced substantially by setting up buffer space at the I/O-points. To solve larger instances appropriately in a short amount of computing time, an iterative improvement procedure and a tabu search performed well.

Finally, our integrated models have shown to be suitable for addressing the new complex problem involving both pallet retrieval and processing. The additional integration of robust optimization leads to more resilient schedules, hedging against uncertainties. In some solutions obtained in the buffering case on the larger instances, up to 20 pallets are stored at one I/O-point simultaneously, where an infinite buffer capacity was presumed. Therefore, future research could explore a more general variant mixing the blocking and buffering case by assuming an arbitrary finite buffer capacity.

Appendix A. Detailed MIP results

Tables 4 and 5 show the results of the four mathematical models presented in Sect. 4 obtained by CPLEX within an one hour computational time limit. The results in Table 4 refer to the nominal integrated models MIP-PRPP$_{\text{Bl}}$ and MIP-PRPP$_{\text{Bu}}$ introduced in Sects. 4.1.1 and 4.2.1, while the results in Table 5 refer to the robust integrated models MIP-PRPP$^{\text{R}}_{\text{Bl}}$ and MIP-PRPP$^{\text{R}}_{\text{Bu}}$ presented in Sects. 4.1.3 and 4.2.3. In both tables, the results are differentiated by the parameters n and m as well as the blocking (left) and buffering case (right). For each given combination, the number of solutions verified as optimally solved, the average optimality gaps reported by CPLEX, and the average computational times required are listed.

It can be seen in Tables 4 and 5 that all instances with $n \leq 8$ pallets were verified as optimally solved, whereas some instances with $n \geq 9$ pallets could not be verified within the time limit. Overall, the computing times rise substantially with increasing numbers of pallets n and I/O-points m. The general trends are the same in the nominal and robust case, but the computing times and optimality gaps increase a bit faster in the robust case. Overall, an instance size of around $n = 10$ pallets seems

Table 4 Nominal MIP results

		Blocking			Buffering		
n	*m*	#Opt	Avg. gap [%]	Avg. time [s]	#Opt	Avg. gap [%]	Avg. time [s]
5	1	10	0.0	0.0	10	0.0	0.0
	2	10	0.0	0.1	10	0.0	0.1
	3	10	0.0	0.1	10	0.0	0.1
6	1	10	0.0	0.0	10	0.0	0.0
	2	10	0.0	0.2	10	0.0	0.2
	3	10	0.0	0.4	10	0.0	0.3
7	1	10	0.0	0.1	10	0.0	0.1
	2	10	0.0	1.4	10	0.0	1.3
	3	10	0.0	4.6	10	0.0	2.7
8	1	10	0.0	0.4	10	0.0	1.0
	2	10	0.0	13.0	10	0.0	12.4
	3	10	0.0	292.9	10	0.0	165.9
9	1	10	0.0	1.4	10	0.0	4.0
	2	10	0.0	54.5	10	0.0	119.4
	3	9	0.6	682.5	9	1.3	773.4
10	1	10	0.0	14.8	10	0.0	47.5
	2	8	1.6	1 307.3	6	2.3	1 842.6
	3	6	3.0	2 452.6	8	2.5	1 919.8

Table 5 Robust MIP results

		Blocking			Buffering		
n	*m*	#Opt	Avg. gap [%]	Avg. time [s]	#Opt	Avg. gap [%]	Avg. time [s]
5	1	10	0.0	0.0	10	0.0	0.0
	2	10	0.0	0.1	10	0.0	0.1
	3	10	0.0	0.2	10	0.0	0.2
6	1	10	0.0	0.1	10	0.0	0.1
	2	10	0.0	0.6	10	0.0	0.4
	3	10	0.0	1.4	10	0.0	0.9
7	1	10	0.0	0.2	10	0.0	0.2
	2	10	0.0	2.5	10	0.0	2.2
	3	10	0.0	10.6	10	0.0	8.0
8	1	10	0.0	0.6	10	0.0	0.4
	2	10	0.0	23.0	10	0.0	17.5
	3	10	0.0	188.3	10	0.0	177.3
9	1	10	0.0	3.3	10	0.0	2.3
	2	10	0.0	444.1	10	0.0	219.9
	3	8	1.6	1 329.1	9	1.8	1 132.3
10	1	10	0.0	23.5	10	0.0	16.2
	2	3	8.0	3 013.6	5	4.5	2 563.4
	3	1	17.7	3 354.6	3	7.5	2 969.1

to be the computational limit beyond which instances can no longer be solved well using the given MIP models.

Acknowledgements The authors are grateful to the editors and two anonymous referees for their helpful and constructive comments.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availibility All instances and results can be found at http://www2.informatik.uos.de/kombopt/data/rop/.

References

Allahverdi A (2015) The third comprehensive survey on scheduling problems with setup times/costs. European Journal of Operational Research 246(2):345–378

Allahverdi A, Gupta J, Aldowaisan T (1999) A review of scheduling research involving setup considerations. Omega 27(2):219–239

Allahverdi A, Ng C, Cheng T et al (2008) A survey of scheduling problems with setup times or costs. European Journal of Operational Research 187(3):985–1032

Ang M, Lim Y, Sim M (2012) Robust storage assignment in unit-load warehouses. Management Science 58(11):2114–2130

Ben-Tal A, Goryashko A, Guslitzer E et al (2004) Adjustable robust solutions of uncertain linear programs. Mathematical Programming 99(2):351–376

Ben-Tal A, El Ghaoui L, Nemirovski A (2009) Robust optimization, vol 28. Princeton University Press

Bertsimas D, den Hertog D (2022) Robust and adaptive optimization. Dynamic Ideas LLC

Bertsimas D, Sim M (2003) Robust discrete optimization and network flows. Mathematical Programming 98(1):49–71

Bertsimas D, Sim M (2004) The price of robustness. Operations Research 52(1):35–53

Bold M, Goerigk M (2021) A compact reformulation of the two-stage robust resource-constrained project scheduling problem. Computers & Operations Research 130:105232

Boysen N, Stephan K (2016) A survey on single crane scheduling in automated storage/retrieval systems. European Journal of Operational Research 254(3):691–704

Bruni M, Di Puglia Pugliese L, Beraldi P et al (2017) An adjustable robust optimization model for the resource-constrained project scheduling problem with uncertain activity durations. Omega 71:66–84

Buckow JN, Goerigk M, Knust S (2024) Retrieval optimization in a warehouse with multiple input/output-points. OR Spectrum. https://doi.org/10.1007/s00291-024-00775-x

Chen ZL, Hall N (2022) Supply chain scheduling, International Series in Operations Research & Management Science, vol 323. Springer

de Koster R, Le-Duc T, Zaerpour N (2012) Determining the number of zones in a pick-and-sort order picking system. International Journal of Production Research 50(3):757–771

Gilmore P, Gomory R (1964) Sequencing a one state-variable machine: A solvable case of the traveling salesman problem. Operations Research 12(5):655–679

Goerigk M, Hartisch M (2024) An introduction to robust combinatorial optimization, International Series in Operations Research & Management Science, vol 361. Springer

Goerigk M, Grün B, Heßler P (2013) Branch and bound algorithms for the bus evacuation problem. Computers & Operations Research 40(12):3010–3020
Gong Y, de Koster R (2011) A review on stochastic models and analysis of warehouse operations. Logistics Research 3(4):191–205
Hosseini A, Otto A, Pesch E (2024) Scheduling in manufacturing with transportation: Classification and solution techniques. European Journal of Operational Research 315(3):821–843
Jiang X, Sun L, Zhang Y et al (2022) Order batching and sequencing for minimising the total order completion time in pick-and-sort warehouses. Expert Systems with Applications 187:115943
Jiu S (2022) Robust omnichannel retail operations with the implementation of ship-from-store. Transportation Research Part E: Logistics and Transportation Review 157:102550
Johnson S (1954) Optimal two-and three-stage production schedules with setup times included. Naval Research Logistics Quarterly 1(1):61–68
Levorato M, Figueiredo R, Frota Y (2022) Exact solutions for the two-machine robust flow shop with budgeted uncertainty. European Journal of Operational Research 300(1):46–57
Miyata H, Nagano M (2019) The blocking flow shop scheduling problem: A comprehensive and conceptual review. Expert Systems with Applications 137:130–156
Qiu R, Sun Y, Sun M (2022) A robust optimization approach for multi-product inventory management in a dual-channel warehouse under demand uncertainties. Omega 109:102591
Reddi S, Ramamoorthy C (1972) On the flow-shop sequencing problem with no wait in process. Journal of the Operational Research Society 23(3):323–331
Roodbergen K, Vis I (2009) A survey of literature on automated storage and retrieval systems. European Journal of Operational Research 194(2):343–362
Sun Y, Qiu R, Sun M (2024) A robust optimization approach for inventory management with limited-time discounts and service-level requirement under demand uncertainty. International Journal of Production Economics 267:109096
Teck S, Dewil R, Vansteenwegen P (2024) A simulation-based genetic algorithm for a semi-automated warehouse scheduling problem with processing time variability. Applied Soft Computing 160:111713
Thorsen A, Yao T (2015) Robust inventory control under demand and lead time uncertainty. Annals of Operations Research 257(1):207–236
van Dal R, van der Veen J, Sierksma G (1993) Small and large TSP: Two polynomially solvable cases of the traveling salesman problem. European Journal of Operational Research 69(1):107–120
Yanıkoğlu İ, Gorissen B, den Hertog D (2019) A survey of adjustable robust optimization. European Journal of Operational Research 277(3):799–813