The static buffer reshuffling and retrieval problem for autonomous mobile robots

Max Disselnmeyer[1][0009−0008−5689−2235], Thomas Bömer[2][0000−0003−4979−7455], Jakob Pfrommer[2][0000−0003−2492−0621], and Anne Meyer[1][0000−0001−6380−1348]

[1] Karlsruhe Institute of Technology, Zirkel 2, 76131 Karlsruhe, Germany
`{max.disselnmeyer,anne.meyer}@kit.edu`
[2] TU Dortmund University, Leonhard-Euler-Straße 5, 44227 Dortmund, Germany
`{thomas.boemer,jakob.pfrommer}@tu-dortmund.de`

Abstract. Buffer zones for moveable shelves or pallets with intermediate products are ubiquitous in production systems to decouple production stages. These buffer zones, usually organized by humans in dense grids, pose a challenge for direct access to all unit loads. To access the retrieval unit load, it often becomes necessary to reshuffle blocking unit loads. When autonomous mobile robots supply production machines without human intervention, their responsibilities encompass both the reshuffling of buffer zones and the transportation of unit loads to production cells adhering to the sequence prescribed by the production plan. This paper introduces the buffer reshuffling and retrieval problem, which is a variant of the block relocation problem known from container terminals at shipyards. We propose an integer programming formulation designed for the static variant of the problem. We created an instance generator with customizable layout sizes and found optimal solutions to solve these instances.

Keywords: block relocation problem · vehicle routing problem · warehouse optimization · integer programming

1 Introduction

Buffer zones are a common way to compensate for differences in material flow velocity between two process steps, such as different machine speeds in production supply. They act as a temporary storage area for materials or products, allowing them to accumulate until they can be processed in the next step. This helps smooth the flow of materials and products through the production process, reducing the risk of bottlenecks and disruptions.

The rapid growth of the autonomous mobile robot (AMR) market reflects the industry's urgent need for more efficient warehouse and production supply solutions. According to Grand View Research [24], the AMR market is projected to expand at a compound annual growth rate of 15.5% between 2023 and 2030. This underscores the potential for AMRs to transform intralogistics operations, including the management of buffer zones in production environments, general warehousing and distribution centers, and e-commerce fulfillment centers, where efficient retrieval and reshuffling are critical for meeting customer demands and optimizing operational efficiency.

Buffer zones traditionally relied on human operators and forklifts for management. Decisions like incoming unit load placement, retrieval prioritization, and relocation of obstructing unit loads were made by drivers, rule-based systems, or heuristics. To enable autonomous operation of buffer zones by AMRs, these decision-making processes must also be automated. Previous research has addressed the unit-load pre-marshalling problem [22] [23], which involves optimizing buffer bay reshuffling based on retrieval priority. Our work tackles a related challenge: the retrieval of unit loads, which includes their removal from the warehouse at runtime.

We introduce the buffer reshuffling and retrieval problem (BRR) in its static variant. Modeled as a combination of the block relocation problem (BRP) (familiar from container terminals) and a vehicle routing problem, the BRR incorporates routing and dispatching decisions for AMRs. For the static variant of the BRR, where the focus is solely on retrieval and reshuffling of existing unit loads, we propose an integer programming model aimed at minimizing AMR travel distance (with a single AMR assumed).

To gain insights into potential solution patterns and inform future heuristic development for larger, real-world instances, we also visualize solutions for small test cases.

2 The buffer reshuffling and retrieval problem

The buffer reshuffling and retrieval problem focuses on finding the shortest routes for AMRs operating in a buffer zone while ensuring the timely retrieval of unit loads. In addition, AMRs must strategically reshuffle the buffer to improve access to blocked unit loads. A unit load x is considered blocked if another unit load y is placed in front of it and the AMR has to relocate y to access x.

The BRR draws inspiration from the Unit-load Pre-Marshalling Problem (UPMP) and utilizes the BRP, where containers in a harbor are stacked on top of each other and must be reshuffled for retrieval with a freight crane. Similarly to container stacks accessible only from the top, unit loads in a buffer zone often form lines with a single access point (see Figure 1). This similarity allows us to leverage research on the BRP and adapt it by incorporating a vehicle routing problem to model the movements of an AMR in a buffer zone. Throughout this work, we will refer to the rows in a buffer that resemble container stacks as lanes (see also UPMP). For the purpose of solving this problem, we assume that these lanes remain constant and unchanging. This means that we will neglect scenarios where moving unit loads could theoretically open access from other directions. The storage slots available in a lane will be called *slots*.

The goal of the buffer reshuffling and relocation problem lies in finding optimal decisions for the AMR to retrieve all unit loads from the buffer according to their time windows. Therefore, retrieval and relocation decisions for the unit

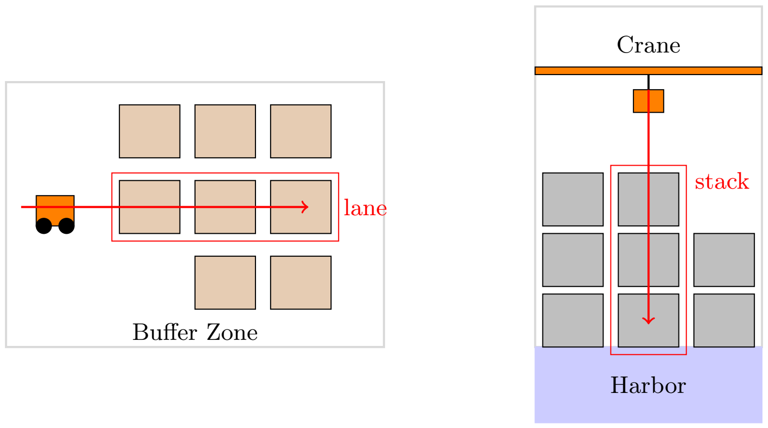


Fig. 1: Access for the AMR in a buffer compared to the access of a crane in a container terminal

loads as well as repositioning decisions for the AMR have to be taken to ensure smooth operation of the buffer zone.

The block relocation problem has been shown to be NP-hard by [4]. Like the block relocation problem, the BRR involves determining the optimal sequence of movements under similar constraints in order to minimize an objective function. This shared combinatorial structure strongly suggests that the BRR is also NP-hard.

3 Previous Work

The Block Relocation Problem (BRP) was formally introduced by [14] in 2006. However, research on this problem had already been conducted, such as by [19], who investigated the Slab Stack Shuffling Problem (SSS), which was later classified as a variant of the BRP. In the literature, the BRP is also known as the Container Relocation Problem (CRP), as most research on this topic focuses on container terminals in shipyards, where this problem arises during the loading and unloading of container ships [18].

A related problem to the BRP is the Pre-Marshalling Problem (PMP). In the PMP, the focus shifts away from retrieval and exclusively targets the pre-sorting of unit loads or containers. This rearrangement ensures a relocation-free retrieval sequence, optimizing efficiency [18]. Building upon this concept, [22] and [23] adapted the PMP to intralogistics as the UPMP. Their approach involves dividing storage bays into virtual lanes, mirroring container stack organization in harbor logistics. Unit loads within these bays are then strategically reshuffled according to the virtual lanes, establishing an optimal configuration for later retrieval. Another related problem, encountered in container shipyards, is the Yard Crane Scheduling Problem (YCSP) [18]. In contrast to the CRP, which emphasizes the optimal container relocation sequence, the YCSP prioritizes scheduling yard cranes to perform various operations within a container terminal. Additionally, the YCSP considers a broader range of constraints, including physical limitations of the cranes, in contrast to the CRP's focus on container relocation constraints [16]. Lastly, another related problem is the Parallel Stack Loading Problem (PLSP), which considers the retrieval sequence already during the storing process and attempts to optimize the stacking of unit loads into the storage [1].

Table 1: Variants of the Block Relocation Problem

Paper	**Unrestricted**	**Slab Stack Shuffling**	**Priorities**	**Time Windows**	**Stochastic**	**Multiple Cranes**	**Dynamic**	**Objective**	**Modelling**
Lixin Tang and Yang [19]	○	●	◐	○	○	○	○	Reshuffles	IP
Kim and Hong [14]	○	○	◐	○	○	○	○	Moves	DP
Wan et al. [28]	○	○	○	○	○	○	○	Reshuffles	IP
Tang and Ren [26]	○	●	◐	○	○	◐	○	Workload	IP,DP
Caserta et al. [3]	○	○	○	○	○	○	○	Relocations	IP
Ünlüyurt and Aydın [27]	○	○	○	○	○	○	○	Time	DP
Hakan Akyüz and Lee [11]	○	○	○	○	○	○	●	Relocations	IP
Borjian et al. [2]	●	○	●	●	○	○	●	Time	IP
Ji et al. [12]	○	○	○	○	○	◐	○	Rehandles	IP
Ku and Arthanari [17]	○	○	●	●	●	○	○	Reshuffles	DP
López-Plata et al. [21]	●	○	●	●	○	○	○	WT	IP
de Melo da Silva et al. [8]	○	○	◐	○	●	○	○	Relocations	DP
Galle et al. [10]	○	○	●	○	●	○	○	Relocations	IP
Chu et al. [5]	○	●	○	○	○	◐	○	Time	MIP
da Silva Firmino et al. [7]	○	○	○	○	○	○	○	Time	IP
Jovanovic et al. [13]	○	○	○	○	○	○	○	Moves	CP
Feng et al. [9]	○	○	●	●	●	○	○	Reshuffles (WT)	DP
Lu et al. [20]	●	○	◐	○	○	○	○	Relocations	MIP
Tanaka and Voß [25]	○	○	○	○	○	○	○	Relocations	IP
Kimms and Wilschewski [15]	●	○	○	○	○	◐	○	Relocations	MIP

Legend	**Priorities**	**Other Features**
○	distinct	not included
◐	duplicate	-
●	both	included

[18] recently conducted a comprehensive survey of optimization methods for the BRP, classifying its variants, which will be used in this publication. Primarily, BRPs differ in the prioritization of unit loads to be retrieved. Unit loads can

have distinct priorities, implying a predefined retrieval sequence, or duplicate priorities, where a group of unit loads can be retrieved in any order. The BRP is then classified into the restricted BRP (rBRP) and the unrestricted BRP (uBRP). Although the uBRP allows for any feasible relocation, the rBRP only allows relocations of unit loads that directly obstruct the retrieval of the target unit load.

Traditionally, the number of moves for a crane in a shipyard terminal while loading or unloading a ship was minimized. Today, alternative objects such as minimizing the total working time [27] or minimizing the crane working time by optimizing the crane trajectories [6] [7] can be found in the literature. [2] introduced the *time-based CRP*, which falls under the variant with time windows. Instead of assigning fixed priorities to containers, which can be directly translated into a retrieval sequence, time windows are calculated for each container, specifying the time interval within which the container must be retrieved. Other publications on the variant with time windows were formulated by e.g. [17], [21] or [10], who added a stochastic component to this variant, where the time windows for retrieval are not fully known in advance. [9] extended the objective considering not only the number of reshuffles but also the total working time (WT).

The aforementioned Slab Stack Shuffling Problem deals with slab yards, commonly found in the steel industry, where slabs are stored in stacks. Unlike other variants of the BRP, the relocated slabs are placed back after retrieval. These yards have existed longer than the necessary computer programs to track the relocated slabs, which led to the practice of putting back the slabs from where they were taken. Additionally, the varying sizes of slabs introduce the constraint that larger slabs cannot be placed on top of smaller slabs for stability reasons. [19] With the advent of sophisticated computer programs capable of tracking slab relocations, [26] introduced an optimization approach that minimized the total workload in the slab yard. Their novel approach incorporated multiple cranes into the optimization framework, considering the distinct operation areas assigned to each crane. The focus of their work was on identifying the optimal set of slabs for retrieval, in order to minimize overall workload.

[12] and [13] considered the stowage plan of a container ship while loading container ships, while the former also considered multiple cranes in the optimization model, however, these cranes served a single bay each. [8] introduced the Block Retrieval Problem (BRTP), where only a subset of the containers has to be retrieved, while keeping the remaining items sorted for loading the next ship.

Finally, the Dynamic Container Relocation Problem deserves attention. Here, new incoming unit loads need to be stored within the storage area while the retrieval process is ongoing. [11] and [2] among others investigated this variant. Table 1 summarizes publications on the BRP with the variants used in each article, the modeling approach used, and the optimized objective.

4 Integer Programming Model

The integer programming model developed here is based on the model proposed by [2] for the BRP and is further developed to address the specific challenges of our problem. First, we introduce the necessary sets for the integer programming model: Let $\mathcal{N} = \{1, \ldots, N\}$ represent the set of unit loads. Let $\mathcal{I} = \{1, \ldots, I\}$ represent the set of lanes, and let $\mathcal{J}_i = \{1, \ldots, J_i\}$ represent the set of positions within lane $i \in \mathcal{I}$. A specific slot is defined as $[i, j]$ for $i \in \mathcal{I}$ and $j \in \mathcal{J}_i$. The slots are counted from the innermost to the outermost. The sink is referenced by $[I, 1]$, where I denotes the lane containing the sink, and we model this lane with a single slot. We differentiate between the set $\mathcal{I}$ and $\mathcal{I}'$, with the latter defined as $\mathcal{I}' = \mathcal{I} \setminus \{I\}$, thus excluding the sink. Let $\mathcal{T} = \{1, \ldots, T\}$ represent the set of discrete time steps used to model time within the problem. This discretization allows for computational tractability. Traditionally, most of the research on BRP has employed a staged approach, where each stage corresponds to a single action performed by the crane in the container terminal. However, a time-based model offers several advantages over a staged model. First, it facilitates the straightforward incorporation of time windows for retrieval. Second, it enables the extension of the model to handle multiple AMRs operating simultaneously, which would be significantly more challenging using a staged approach. Lastly, we define d_{ijkl} as the distance between the slots $[i, j]$ and $[k, l]$.

With the definition of these sets, we refer to Table 2 for additional notations for the unit loads and their retrieval windows, the storage slots and their distances, and the moves performed by the AMRs.

The decision variables for the IP model will be declared as follows:

$$b_{ijnt} = \begin{cases} 1 & \text{if unit load } n \text{ is in slot } [i,j] \text{ at time } t, \\ 0 & \text{otherwise;} \end{cases} \tag{1}$$
$$\forall i \in \mathcal{I}', \forall j \in \mathcal{J}_i, \forall n \in \mathcal{N}, \forall t \in \mathcal{T}$$

$$x_{ijklnt} = \begin{cases} 1 & \text{if unit load } n \text{ is relocated from } [i,j] \text{ to } [k,l] \text{ at time } t, \\ 0 & \text{otherwise;} \end{cases} \tag{2}$$
$$\forall i, k \in \mathcal{I}', \forall j \in \mathcal{J}_i, \forall l \in \mathcal{J}_k, \forall n \in \mathcal{N}, \forall t \in \mathcal{T}$$

$$y_{ijnt} = \begin{cases} 1 & \text{if unit load } n \text{ is retrieved from } [i,j] \text{ at time } t, \\ 0 & \text{otherwise;} \end{cases} \tag{3}$$
$$\forall i \in \mathcal{I}', \forall j \in \mathcal{J}_i, \forall n \in \mathcal{N}, \forall t \in \mathcal{T}$$

$$g_{nt} = \begin{cases} 1 & \text{if unit load } n \text{ has been retrieved at time } t' \in \{1, ..., t-1\}, \\ 0 & \text{otherwise;} \end{cases} \tag{4}$$
$$\forall n \in \mathcal{N}, \forall t \in \mathcal{T}$$

$$e_{ijklt} = \begin{cases} 1 & \text{if the AMR repositions from slot } [i,j] \text{ to slot } [k,l] \text{ at time } t, \\ 0 & \text{otherwise;} \end{cases} \tag{5}$$
$$\forall i, k \in \mathcal{I}, \forall j \in \mathcal{J}_i, \forall l \in \mathcal{J}_k, \forall n \in \mathcal{N}, \forall t \in \mathcal{T}$$

	Notation	**Description**
Unit load	(u_n, r_n)	A unit load with label u_n. The time-step at which the retrieval time window opens at r_n.
Slot	$[i, j]$	The slot notation $[i, j]$ represents the slot in lane i and position j within the lane. This slot indexing scheme is used to efficiently locate and manage items within the buffer zone. Slots are numbered from the back of the lane.
Outermost slot in a lane	$[i, J_i]$	The outermost slot J_i in lane i.
Sink	$[I, 1] = [I, J_I]$	The sink will be modelled as a slot in the lane with the highest value for the index i.
Relocation move	$[i, j] \to [k, l]$	A relocation move from slot $[i, j]$ to slot $[k, l]$.
Retrieval move	$[i, j] \to [I, J]$	Retrieving a unit load from slot $[i, j]$.
Repositioning move	$[i, j] \hookrightarrow [k, l]$	Repositioning the vehicle from slot $[i, j]$ to $[k, l]$
Retrieval window	$[r_n, r_n + \rho_n]$	Time window for the retrieval of unit load u_n with ρ_n being the maximum amount of delay of u_n.
Distance between slots	d_{ijkl}	The distance between the slots with the indices $[i, j]$ and $[k, l]$.
Travel time between slots	τ_{ijkl}	The time difference between the slots with the indices $[i, j]$ and $[k, l]$.

Table 2: Notation for Integer Programming Model

$$c_{ijt} = \begin{cases} 1 & \text{if the AMR is at slot } [i,j] \text{ at time } t, \\ 0 & \text{otherwise;} \end{cases} \qquad (6)$$
$$\forall i \in \mathcal{I}, \forall j \in \mathcal{J}_i, \forall t \in \mathcal{T}$$

These decision variables are all binary and will be used to represent the decisions taken for the AMR on how to retrieve and reshuffle the unit loads. (4) introduces an additional index t', which denotes all the time steps that occurred before time step t. For instance, if $t = 5$, t' would indicate all time steps $0 < t' < t$. Therefore, the variable g_{nt} in (4) with $t = 5$ would have a value of 1, if the unit load with the index n was retrieved during any of the time steps $\{1, 2, 3, 4\}$. We furthermore extended the model formulated by [2] with additional variables e_{ijklt} and c_{ijt} which allow us to also model the repositioning decisions for the AMR and consider repositioning moves in the total distance traveled.

The variables b_{ijnt}, x_{ijklnt} and y_{ijnt} are defined for all slots except the sink. This distinction is crucial for modeling the flow of unit loads and the movement of the AMR, as the sink does not hold unit loads but serves as a departure point for retrieved loads.

With the retrieval window of a unit load defined as $[r_n, r_n+\rho_n]$, the maximum number of time steps T allowed to retrieve all the unit loads in the problem setting can be expressed as follows:

$$T = \max_{n\in\mathcal{N}, r_n<\infty} \{r_n + \rho_n\} \tag{7}$$

To simplify the model, we assume that the travel time for the AMR is equal to the travel distance. This eliminates the need to calculate the travel time τ from the travel distance d, which could be done using (8). This equation could be adapted in future work to include acceleration and deceleration, as well as different velocities for the AMRs. Furthermore, we assume here a minimum of 1 for the travel time, to prevent multiple repositioning decisions for an AMR during a single time step.

$$\tau_{ijkl} = \max(1, d_{ijkl}) \tag{8}$$

The objective is to minimize the distance traveled by the AMR during the reshuffling and retrieval processes (9). This is achieved by multiplying all decisions taken by the corresponding travel distances and summing all those decisions and their distances up. While traditionally research on block relocation problems focuses on optimizing the number of relocations, crane movements, or similar measures, we argue that the distance traveled by autonomous mobile robots (AMRs) is a more critical performance indicator for this problem. First, AMRs consume energy while operating, and thus keeping the distance travelled minimized will also reduce the energy consumption and the associated energy costs. Additionally, a lower distance traveled will lead to an improved efficiency, as AMRs can complete their tasks more quickly, leading to increased throughput and productivity. Finally, AMRs will experience less wear and tear on their hardware when moving shorter distances, which could extended lifespans of the components.

Objective function

$$\begin{aligned}\min \sum_{i\in\mathcal{I}'}\sum_{j\in\mathcal{J}_i}\sum_{n\in\mathcal{N}}\sum_{t\in\mathcal{T}}\Big(y_{ijnt}\cdot d_{ijI1} + \sum_{k\in\mathcal{I}'}\sum_{l\in\mathcal{J}_k} x_{ijklnt}\cdot d_{ijkl}\Big) \\ + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}_i}\sum_{k\in\mathcal{I}}\sum_{l\in\mathcal{J}_k}\sum_{t\in\mathcal{T}} e_{ijklt}\cdot d_{ijkl}\end{aligned} \tag{9}$$

Subject to:

$$\sum_{i\in\mathcal{I}'}\sum_{j\in\mathcal{J}_i} b_{ijnt} + g_{nt} \le 1, \qquad \forall n\in\mathcal{N}, \forall t\in\mathcal{T} \tag{10}$$

$$\sum_{n\in\mathcal{N}} b_{ijnt} \le 1, \qquad \forall i\in\mathcal{I}', \forall j\in\mathcal{J}_i, \forall t\in\mathcal{T} \tag{11}$$

$$\sum_{n\in\mathcal{N}} b_{ijnt} \ge \sum_{n\in\mathcal{N}} b_{ij+1nt}, \qquad \forall i\in\mathcal{I}', \forall j\in\mathcal{J}_i\setminus J_i, \forall t\in\mathcal{T} \tag{12}$$

$$\sum_{k\in\mathcal{I}'}\sum_{l\in\mathcal{J}_k}\sum_{n\in\mathcal{N}} x_{ijklnt} + \sum_{k\in\mathcal{I}}\sum_{l\in\mathcal{J}_k} e_{ijklt} + \sum_{n\in\mathcal{N}} y_{ijnt} = c_{ijt}, \qquad \forall i\in\mathcal{I}', \forall j\in\mathcal{J}_i, \forall t\in\mathcal{T} \quad (13)$$

$$\sum_{k\in\mathcal{I}}\sum_{l\in\mathcal{J}_k} e_{I1klt} = c_{I1t}, \qquad \forall t\in\mathcal{T} \quad (14)$$

$$y_{ijnt} + \sum_{k\in\mathcal{I}'}\sum_{l\in\mathcal{J}_k} x_{ijklnt} \le b_{ijnt}, \qquad \forall i\in\mathcal{I}', \forall j\in\mathcal{J}_i, \forall n\in\mathcal{N}, \forall t\in\mathcal{T} \quad (15)$$

$$b_{ijnt} = b_{ijn(t-1)} + \sum_{k\in\mathcal{I}'}\sum_{l\in\mathcal{J}_k}\Big(x_{klijn(t-\tau_{klij})} - x_{ijkln(t-1)}\Big) - y_{ijn(t-1)}, \qquad \forall i\in\mathcal{I}', \forall j\in\mathcal{J}_i, \forall n\in\mathcal{N}, \forall t\in\mathcal{T}\setminus 1 \quad (16)$$

$$c_{ijt} = c_{ij(t-1)} - \sum_{n\in\mathcal{N}} y_{ijn(t-1)} + \sum_{k\in\mathcal{I}'}\sum_{l\in\mathcal{J}_k}\sum_{n\in\mathcal{N}} x_{klijn(t-\tau_{klij})} + \sum_{k\in\mathcal{I}}\sum_{l\in\mathcal{J}_k} e_{klij(t-\tau_{klij})} - \sum_{k\in\mathcal{I}'}\sum_{l\in\mathcal{J}_k}\sum_{n\in\mathcal{N}} x_{ijkln(t-1)} - \sum_{k\in\mathcal{I}}\sum_{l\in\mathcal{J}_k} e_{ijkl(t-1)}, \qquad \forall i\in\mathcal{I}', \forall j\in\mathcal{J}_i, \forall t\in\mathcal{T}\setminus 1 \quad (17)$$

$$c_{I1t} = c_{I1(t-1)} + \sum_{i\in\mathcal{I}'}\sum_{j\in\mathcal{J}_i}\sum_{n\in\mathcal{N}} y_{ijn(t-\tau_{ijI1})} + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}_i}\big(e_{ijI1(t-\tau_{ijI1})} - e_{I1ij(t-1)}\big), \qquad \forall t\in\mathcal{T}\setminus 1 \quad (18)$$

$$g_{nt} = \sum_{i\in\mathcal{I}'}\sum_{j\in\mathcal{J}_i}\sum_{t'=1}^{t-1} y_{ijnt'}, \qquad \forall n\in\mathcal{N}, \forall t\in\mathcal{T} \quad (19)$$

$$\sum_{i\in\mathcal{I}'}\sum_{j\in\mathcal{J}_i}\sum_{t=1}^{r_n-\tau_{ijI1}} y_{ijnt} = 0, \qquad \forall n\in\mathcal{N} \quad (20)$$

$$\sum_{i\in\mathcal{I}'}\sum_{j\in\mathcal{J}_i}\sum_{t=r_n-\tau_{ijI1}}^{r_n+\rho_n-\tau_{ijI1}} y_{ijnt} = 1, \qquad \forall n\in\mathcal{N} \quad (21)$$

$$\sum_{i\in\mathcal{I}'}\sum_{j\in\mathcal{J}_i}\sum_{t=r_n+\rho_n-\tau_{ijI1}+1}^{T} y_{ijnt} = 0, \qquad \forall n\in\mathcal{N} \quad (22)$$

Constraints (10) ensure that the model accounts for all unit loads, preventing loss or misrouting. These constraints enforce that a unit load must be either in the storage area or marked as retrieved. Constraints (11) restrict the storage area to a maximum of one unit load per slot at any given time. these constraints prevent overloading of slots and ensure efficient utilization of storage space. Constraints (12) enforce a full storage area, prohibiting empty slots in the lanes. These constraints ensure compact storage and prevent hollow spaces from forming. Constraints (13) restrict each available vehicle to a single retrieval or relocation action per time step, preventing simultaneous movements. Similarly, constraints (14) limit each available vehicle at the sink to one repositioning action per time step. Constraints (15) ensure that unit loads can only be moved from a slot in the storage area if they are present in this slot. Constraints (16) serve as a tracking mechanism for unit load movements. They update the storage area configuration in t based on the previous configuration in $t-1$ and the previous decisions taken. These constraints maintain a record of the unit load positions and retrievals. We adapted these constraints from the model proposed in [2] and modified them to also incorporate the repositioning procedures of the AMR. Constraints (17) offer the same update mechanism as (16) for the position of the AMR by checking if the AMR was at a specific location in the previous time step and updating this value by the previous decisions taken. Constraints (18) handle these updates for the sink. Constraints (19) establish a connection between the configuration variables g and their corresponding retrieval variables y. These constraints ensure that the retrieval actions align with the updated storage configurations. Constraints (20), (21), and (22) collectively mandate that unit loads are retrieved within their designated retrieval time windows. These constraints prevent premature or belated retrievals and ensure timely fulfillment of customer requests.

Finally, the starting points for the unit loads and the AMR must be set. This can be achieved by configuring the values of b_{ijnt} and c_{ijt} for $t = 1$ according to the instance's specifications (see constraints (23) and (24)).

$$b_{ijn1} = \begin{cases} 1, & \text{if unitload } n \text{ starts in slot } [i,j] \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in \mathcal{I}', \forall j \in \mathcal{J}_i, \forall n \in \mathcal{N} \quad (23)$$

$$c_{ij1} = \begin{cases} 1, & \text{if the AMR starts in slot } [i,j] \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}_i \quad (24)$$

5 Computational Experiments

5.1 Instances

In our experiments, we use an instance generator with a symmetrical layout for the buffer zone, where the storage area is divided into same-size bays (see Figure 2). However, this problem and our proposed integer model are affected by the curse of dimensionality, making larger instances almost unsolvable in a realistic time frame. Therefore, in this paper, we focus on a single bay.

We employ a network flow model, adapted from [22] and [23], to partition the bay into virtual lanes. This approach prioritizes minimizing initial blockages within access points and lanes, thus enhancing retrieval efficiency. First, we define the access directions for the bay and calculate optimal access points and the virtual lanes behind them (see Figure 3). For an in-depth explanation of the process, we refer to [22] and [23].

This partitioning of the bay into virtual lanes enables us to apply our model to storage with multiple access directions, a layout often operated by AMRs. To further reduce complexity, we neglect the acceleration and deceleration processes of the AMR as well as the handling times for loading and unloading the unit loads. For the unit loads, we define an initial fill level for the buffer area and generate retrieval windows according to a normal distribution with a specified mean and standard deviation.

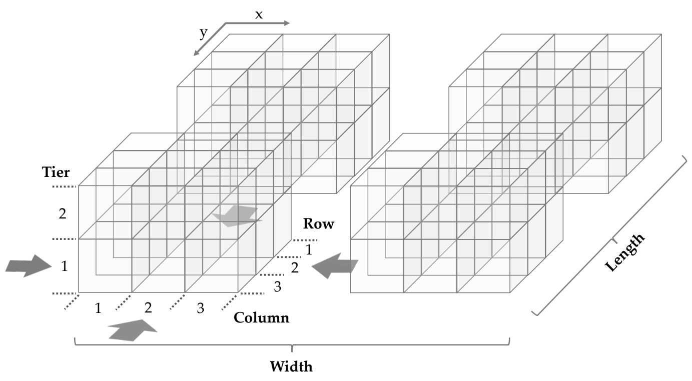


Fig. 2: Illustration of a buffer zone with multiple bays [23]

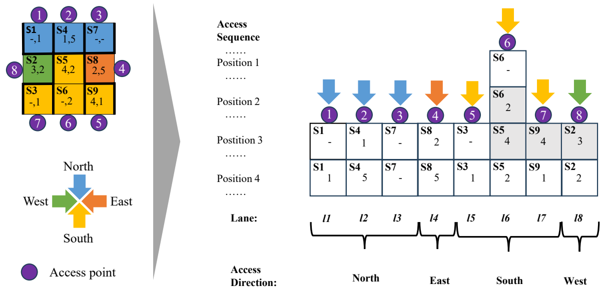


Fig. 3: Partitioning the bay into virtual lanes (adapted from [23])

5.2 Example Solutions

Our experiments were carried out on an Intel(R) Xeon(R) w5-2445 CPU with Gurobi Optimizer version 11.0.1. Runtimes varied significantly depending on the instance size and selected parameters, such as fill level and latest time window. For small instances (3x3 bay, low fill level of 30%), we could find optimal solutions in under two minutes in most cases. However, for larger instances with either higher fill levels or larger bays, no solution was found within two hours of runtime.

Figure 4 illustrates reshuffling and retrieval decisions for a 3x3 bay instance. This solution was found in 400 seconds with a MIPGap of 4.6%. We color-code each slot based on access direction (south access in yellow, east access in red). The illustration highlights the static buffer reshuffling and retrieval problem's similarities to scheduling problems, especially its need to optimize retrieval sequences and potentially schedule reshuffling for efficient retrieval. In this variant, the optimal solution depends less on specific unit load storage locations (initial or relocated) and more on the retrieval sequence and timing of relocations. Consequently, the focus shifts away from complex storage and dispatching toward minimizing AMR travel distance for reshuffling actions. This emphasizes the

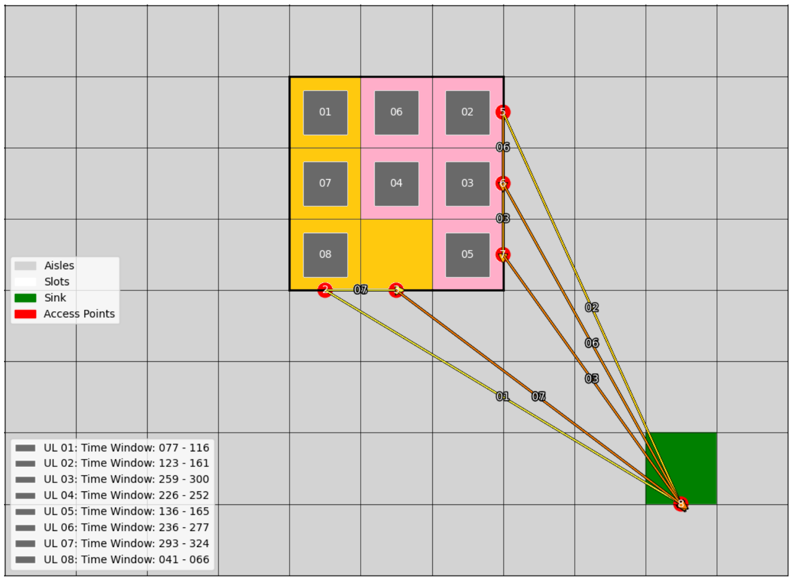
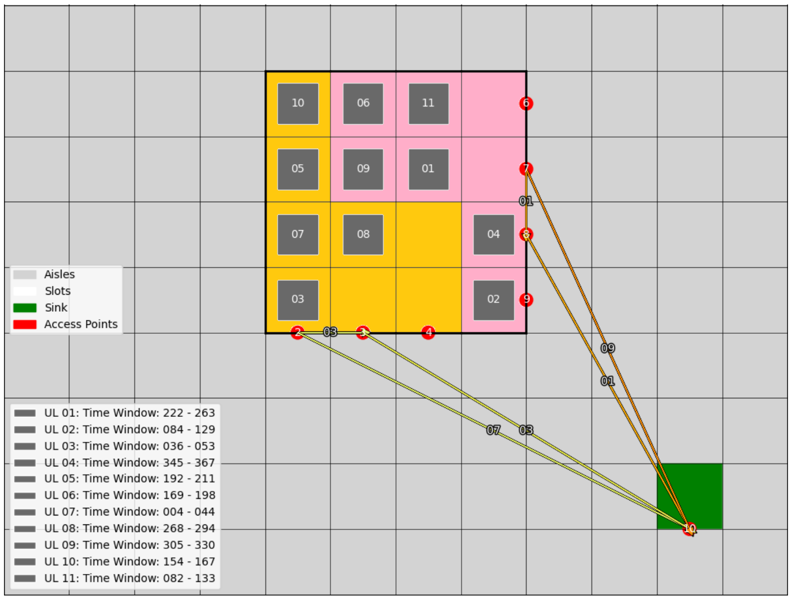

01
06
02
07
04
03
08
05
Aisles
Slots
Sink
Access Points
UL 01: Time Window: 077 - 116
UL 02: Time Window: 123 - 161
UL 03: Time Window: 259 - 300
UL 04: Time Window: 226 - 252
UL 05: Time Window: 136 - 165
UL 06: Time Window: 236 - 277
UL 07: Time Window: 293 - 324
UL 08: Time Window: 041 - 066

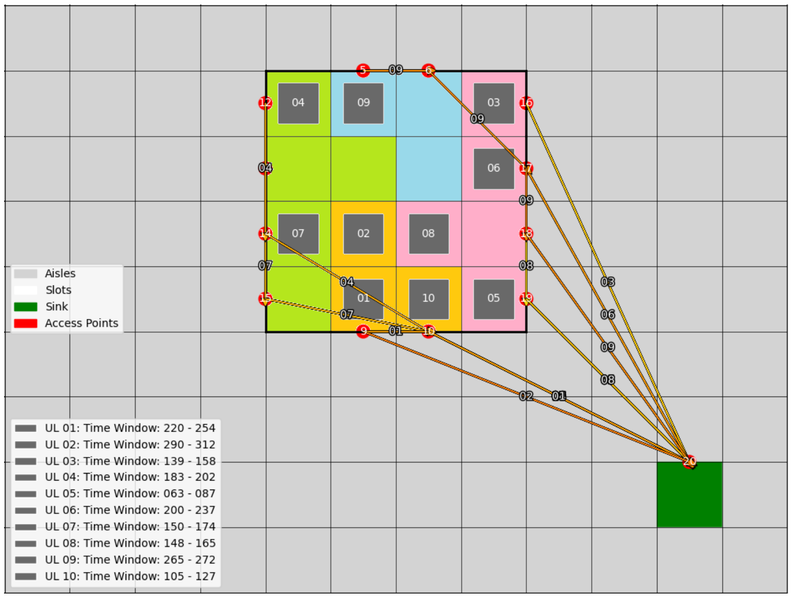
Fig. 4: Example solution for a 3x3 bay showcasing multiple reshuffling moves

need for further research to include multiple AMRs and accommodate incoming unit loads that require storage within the buffer.

Figure 5 demonstrates part of the nearly optimal solution for a 4x4 bay instance with access from the east and south side. We illustrate only loaded AMR movements (retrieving or relocating unit loads) to enhance readability. Movement arrows include the carried unit load and are colored darker the later they are executed. We colored the virtual lanes behind the access points 2 and 7 to illustrate unit loads 01 and 03 blocking unit loads 09 and 07 respectively. This example suggests a potential reshuffling strategy: unit loads (e.g., UL 01 and UL 03) are relocated closer to the sink shortly before their due dates and when blocking other unit loads, offering first insights for developing heuristics. The solver took 40,000 seconds to find this solution with a 6% MIPGap, highlighting the problem's complexity and the curse of dimensionality, which hinders solving larger instances.

Lastly, Figure 6 shows the solution found for a 4x4 bay with access from all four directions. This solution has a MIPGap of 6.2% to the optimal, found after 40,000 seconds of runtime. This instance indicates that increased access directions result in shorter virtual lanes and fewer unit load blockages. Consequently, unit load relocations here primarily serve to enable faster retrievals in later time steps by moving them closer to the sink. This highlights an alternative problem-solving approach: carefully designing buffer layouts to minimize blockages can reduce the need for intensive computation. However, limitations exist – the shape of unit loads and available warehouse space may not always allow retrieval from multiple access directions.


Aisles
Slots
Sink
Access Points
UL 01: Time Window: 222 - 263
UL 02: Time Window: 084 - 129
UL 03: Time Window: 036 - 053
UL 04: Time Window: 345 - 367
UL 05: Time Window: 192 - 211
UL 06: Time Window: 169 - 198
UL 07: Time Window: 004 - 044
UL 08: Time Window: 268 - 294
UL 09: Time Window: 305 - 330
UL 10: Time Window: 154 - 167
UL 11: Time Window: 082 - 133
Fig. 5: Example solution for a 4x4 bay with 2 access directions


Aisles
Slots
Sink
Access Points
UL 01: Time Window: 220 - 254
UL 02: Time Window: 290 - 312
UL 03: Time Window: 139 - 158
UL 04: Time Window: 183 - 202
UL 05: Time Window: 063 - 087
UL 06: Time Window: 200 - 237
UL 07: Time Window: 150 - 174
UL 08: Time Window: 148 - 165
UL 09: Time Window: 265 - 272
UL 10: Time Window: 105 - 127
Fig. 6: Example solution for a 4x4 bay with 4 access directions

6 Conclusion

This work introduced the BRR, a novel combinatorial optimization problem merging concepts from the BRP and vehicle routing while drawing inspiration from the UPMP. We illustrated the problem's real-world applications, established its relationship to block relocation, and surveyed the relevant literature on variants and modeling approaches. Building upon this foundation, we formulate an integer programming model for the static BRR, extending the BRP with vehicle routing considerations for autonomous mobile robot optimization.

Computational results on small instances illustrated the model's functionality and provided insights for developing heuristics or approximation algorithms to achieve solutions within practical time constraints. However, the complexity of the problem makes larger instances unsolvable in realistic timespans. Future research directions should include accommodating dynamic unit load arrivals, incorporating multiple AMR, and developing efficient solution methods for larger, more complex instances.

Acknowledgments. This research was funded by the European Union - NextGenerationEU - funding code 13IK032I and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - project number 276879186

References

[1] Sven Boge and Sigrid Knust. The parallel stack loading problem minimizing the number of reshuffles in the retrieval stage. *European Journal of Operational Research*, 280(3):940–952, 2020. `https://doi.org/10.1016/j.ejor.2019.08.005`.

[2] Setareh Borjian, Vahideh H. Manshadi, Cynthia Barnhart, and Patrick Jaillet. Managing relocation and delay in container terminals with flexible service policies. 2015. `https://doi.org/10.48550/arXiv.1503.01535`.

[3] Marco Caserta, Silvia Schwarze, and Stefan Voß. A mathematical formulation and complexity considerations for the blocks relocation problem. *European Journal of Operational Research*, 219(1):96–104, 2012. `https://doi.org/10.1016/j.ejor.2011.12.039`.

[4] Marco Caserta, Silvia Schwarze, and Stefan Voß. A mathematical formulation and complexity considerations for the blocks relocation problem. *European Journal of Operational Research*, 219(1):96–104, 2012. ISSN 0377-2217. `https://doi.org/10.1016/j.ejor.2011.12.039`.

[5] Feng Chu, Junkai He, Feifeng Zheng, and Ming Liu. Scheduling multiple yard cranes in two adjacent container blocks with position-dependent processing times. *Computers Industrial Engineering*, 136:355–365, 2019. ISSN 0360-8352. `https://doi.org/10.1016/j.cie.2019.07.013`.

[6] Andresson da Silva Firmino, Ricardo Martins de Abreu Silva, and Valeria Cesario Times. An exact approach for the container retrieval problem to reduce crane's trajectory. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 933–938, 2016. `https://doi.org/10.1109/ITSC.2016.7795667`.

[7] Andresson da Silva Firmino, Ricardo Martins de Abreu Silva, and Valéria Cesário Times. A reactive grasp metaheuristic for the container retrieval problem to reduce crane's working time. *Journal of Heuristics*, 25: 141–173, 2019. `https://doi.org/10.1007/s10732-018-9390-0`.

[8] Marcos de Melo da Silva, Güneş Erdoğan, Maria Battarra, and Vitaly Strusevich. The block retrieval problem. *European Journal of Operational Research*, 265(3):931–950, 2018. ISSN 0377-2217. `https://doi.org/10.1016/j.ejor.2017.08.048`.

[9] Yuanjun Feng, Dong-Ping Song, Dong Li, and Qingcheng Zeng. The stochastic container relocation problem with flexible service policies. *Transportation Research Part B: Methodological*, 141:116–163, 2020. ISSN 0191-2615. `https://doi.org/10.1016/j.trb.2020.09.006`.

[10] Virgile Galle, Vahideh H Manshadi, S Borjian Boroujeni, Cynthia Barnhart, and Patrick Jaillet. The stochastic container relocation problem. *Transportation Science*, 52(5):1035–1058, 2018. `https://doi.org/10.1016/j.trb.2020.09.006`.

[11] M Hakan Akyüz and Chung-Yee Lee. A mathematical formulation and efficient heuristics for the dynamic container relocation problem. *Naval Re-*

search Logistics (NRL), 61(2):101–118, 2014. https://doi.org/10.1002/nav.21569.

[12] Mingjun Ji, Wenwen Guo, Huiling Zhu, and Yongzhi Yang. Optimization of loading sequence and rehandling strategy for multi-quay crane operations in container terminals. *Transportation Research Part E: Logistics and Transportation Review*, 80:1–19, 2015. ISSN 1366-5545. https://doi.org/10.1016/j.tre.2015.05.004.

[13] Raka Jovanovic, Shunji Tanaka, Tatsushi Nishi, and Stefan Voß. A grasp approach for solving the blocks relocation problem with stowage plan. *Flexible Services and Manufacturing Journal*, 31:702–729, 2019. https://doi.org/10.1007/s10696-018-9320-3.

[14] Kap Hwan Kim and Gyu-Pyo Hong. A heuristic rule for relocating blocks. *Computers Operations Research*, 33(4):940–954, 2006. ISSN 0305-0548. https://doi.org/10.1016/j.cor.2004.08.005. Part Special Issue: Optimization Days 2003.

[15] Alf Kimms and Fabian Wilschewski. A new modeling approach for the unrestricted block relocation problem. *OR Spectrum*, pages 1–41, 2023. https://doi.org/10.1007/s00291-023-00728-w.

[16] Damla Kizilay and Deniz Türsel Eliiyi. A comprehensive review of quay crane scheduling, yard operations and integrations thereof in container terminals. *Flexible Services and Manufacturing Journal*, 33(1):1–42, 2021. https://doi.org/10.1007/s10696-020-09385-5.

[17] Dusan Ku and Tiru S. Arthanari. Container relocation problem with time windows for container departure. *European Journal of Operational Research*, 252(3):1031–1039, 2016. ISSN 0377-2217. https://doi.org/10.1016/j.ejor.2016.01.055.

[18] Charly Lersteau and Weiming Shen. A survey of optimization methods for block relocation and premarshalling problems. *Computers Industrial Engineering*, 172:108529, 2022. ISSN 0360-8352. https://doi.org/10.1016/j.cie.2022.108529.

[19] Aiying Rong Lixin Tang, Jiyin Liu and Zihou Yang. Modelling and a genetic algorithm solution for the slab stack shuffling problem when implementing steel rolling schedules. *International Journal of Production Research*, 40(7):1583–1595, 2002. https://doi.org/10.1080/00207540110110118424.

[20] Chao Lu, Bo Zeng, and Shixin Liu. A study on the block relocation problem: Lower bound derivations and strong formulations. *IEEE Transactions on Automation Science and Engineering*, 17(4):1829–1853, 2020. https://doi.org/10.1109/TASE.2020.2979868.

[21] Israel López-Plata, Christopher Expósito-Izquierdo, Eduardo Lalla-Ruiz, Belén Melián-Batista, and J. Marcos Moreno-Vega. Minimizing the waiting times of block retrieval operations in stacking facilities. *Computers Industrial Engineering*, 103:70–84, 2017. ISSN 0360-8352. https://doi.org/10.1016/j.cie.2016.11.015.

[22] Jakob Pfrommer, Anne Meyer, and Kevin Tierney. Solving the unit-load pre-marshalling problem in block stacking storage systems with multiple

access directions. *European Journal of Operational Research*, 2023. https://doi.org/10.1016/j.ejor.2023.08.044.

[23] Jakob Pfrommer, Thomas Bömer, Daniyar Akizhanov, and Anne Meyer. Sorting multibay block stacking storage systems. 2024. https://doi.org/10.48550/arXiv.2405.04847.

[24] Grand View Research. Autonomous mobile robots (amr) market size, share trends analysis report by component (hardware, software, services), by type, by battery type, by end-use, by region, and segment forecasts, 2023 - 2030. 2024.

[25] Shunji Tanaka and Stefan Voß. An exact approach to the restricted block relocation problem based on a new integer programming formulation. *European Journal of Operational Research*, 296(2):485–503, 2022. https://doi.org/10.1016/j.ejor.2021.03.062.

[26] Lixin Tang and Huizhi Ren. Modelling and a segmented dynamic programming-based heuristic approach for the slab stack shuffling problem. *Computers Operations Research*, 37(2):368–375, 2010. ISSN 0305-0548. https://doi.org/10.1016/j.cor.2009.05.011.

[27] Tonguç Ünlüyurt and Cenk Aydın. Improved rehandling strategies for the container retrieval process. *Journal of Advanced Transportation*, 46(4):378–393, 2012. https://doi.org/10.1002/atr.1193.

[28] Yat-wah Wan, Jiyin Liu, and Pei-Chun Tsai. The assignment of storage locations to containers for a container stack. *Naval Research Logistics (NRL)*, 56(8):699–713, 2009. https://doi.org/10.1002/nav.20373.