Logistics Research

23

Unidimensional puzzle-based storage systems

Héctor J. Carlo and Andrés F. Blanco-Quintana
Department of Industrial Engineering, University of Puerto Rico Mayagüez, Mayagüez, Puerto Rico
Fabiola E. Robles-Vega
Department of Computer Science and Engineering, University of Puerto Rico Mayagüez, Mayagüez, Puerto Rico, and
Joaquín E. Nieves-Báez
Department of Industrial Engineering, University of Puerto Rico Mayagüez, Mayagüez, Puerto Rico

Received 14 January 2025
Revised 18 March 2025
5 May 2025
Accepted 26 May 2025

Abstract

Purpose – A paradigm for Puzzle-Based Storage Systems (PBS) is that all grid cells have the capacity to move loads horizontally and vertically. This research challenges this paradigm by considering unidimensional PBS (UPBS), where some grid cells are limited to either horizontal or vertical movements. UPBS simultaneously reduce the investment cost and operational complexity of PBS, at the expense of a potentially lower system throughput.
Design/methodology/approach – A linear program (LP) formulation is presented to determine the optimal retrieval path for a load using a single escort (i.e. an open cell in the grid that allows load movement) in a UPBS. The LP is solved recursively to understand the tradeoffs of unidimensional designs for a 4×4 system. Formulations to solve the single-load multi-escort problem are also proposed. Lastly, an existing decentralized control PBS algorithm was used to develop managerial insights for designing UPBS, considering the simultaneous retrieval of multiple loads using multiple escorts and input/output points.
Findings – It is concluded that UPBS with three unidimensional cells would reduce the PBS cost by 1.8 times the capital cost of a bidimensional cell, at the expense of an 8.34% average reduction in throughput. It is argued that multi-escort UPBS designs should be similar to the ones empirically produced for single-escort UPBS. Lastly, it is concluded that UPBS may provide a favorable capital investment cost-to-throughput tradeoff.
Originality/value – This study is a pioneer in considering UPBS. The empirical evidence of a favorable system cost-to-throughput ratio of UPBS improves the business case of implementing PBS in practice.

Keywords Puzzle-based storage, High-density storage, Automated warehouse, Optimization
Paper type Research paper

1. Introduction

Puzzle-Based Storage Systems (PBS), also known as single-level live-cube systems, are very high-density parts-to-person storage systems for unit loads. The PBS design is based on an automated grid composed of cells capable of sliding loads horizontally and vertically. Each cell in a PBS holds one unit load, which is only moved when the load needs to be retrieved or if it needs to be moved out of the way to gain access to another load. The structure of a PBS is based on the classic 15-puzzle children's game, where 15 numbered tiles need to be slid in a 4×4 grid (15 tiles and an empty position) until all the tiles are placed in numerical order. In PBS, the desired loads are moved along the grid by continuously sliding them to an empty position (i.e. an open cell). Since loads need to be slid into open cells, the open cells are

This work was supported in part by the National Science Foundation under Grant No. HRD-2008186. The authors are grateful to Dr. Kevin Gue for proposing the research problem.

Logistics Research
Vol. 18 No. 1/2, 2025
pp. 23-45
Emerald Publishing Limited
e-ISSN: 1865-0368
p-ISSN: 1865-035X
DOI 10.1108/LORE-01-2025-0006

commonly referred to as escorts. In PBS, cells may only be moved in the north/south or east/west directions via an escort.

The literature describes two main PBS designs: *pop-up* and *shuttle*. In the pop-up design, each cell is equipped with a bidirectional conveyor for sliding loads in one dimension (e.g. vertically) and a bidirectional pop-up conveyor to achieve movements in the other dimension (e.g. horizontally). A typical example of a pop-up PBS includes roller conveyors with pop-up rollers in each cell. The pop-up rollers provide orthogonal movement with respect to the roller conveyor. On the other hand, the shuttle design uses independent material handling equipment (MHE) to move the loads. The MHE may be dedicated to cells or may work in collaboration to pick-up and move the loads upon request. This study focuses on PBS with the pop-up design. Pop-up designs achieve much faster retrieval speeds at the expense of a more rigid configuration (e.g. fixed grid shape and cell size).

Consider the 4×4 PBS grid depicted in Figure 1a with cells identified by their column and row numbers –in this order. Each of the sixteen cells of the grid either holds a load or is an escort. This PBS has a single escort (white) that is initially located at the input/output (I/O) point on the southwest corner with grid coordinates (1,1). The input (I) point is where loads enter the PBS and the output (O) point is where loads exit the PBS. In this example, the input and output points coincide. The requested load is represented by the black square in the grid with coordinates (3,3). The remaining fourteen cells, identified in gray, contain other stored loads. Moving the desired load (black) closer to the I/O requires placing the escort in a cell either immediately west or south of the desired load. Although technically the escort does not physically move (as the escort is an empty cell so there is nothing to move), it is conceptually easier to refer to "moving the escort" when describing the process of moving loads to open a cell. In this example, as illustrated by the arrows in Figure 1a, the escort is to be moved to the cell south of the desired load by moving the loads originally in cells (2,1) and (3,1) simultaneously west and then moving the load in cell (3,2) south. The numerals in Figure 1 identify the sequence in which the cell movements are performed. This type of simultaneous movement is known in the PBS literature as block or tandem movements. Notice that it is not possible to move the load in cell (3,2) south at the same time the load in that cell is moving west due to interference. After moving the desired load south, the escort is now the desired load's original location with coordinates (3,3), as in Figure 1b. The remaining images in Figure 1 depict the sequence of cell movements required to retrieve the desired load via the I/O point.

Arriving loads access the PBS via a set of predefined input point(s) and depart the PBS through a set of predefined output point(s), which may not coincide. As expected, the I/O point (s) in the PBS must be on the edge of the PBS grid. Unlike most storage systems that have predefined aisles, PBS achieve very high-density by being aisle-less. In general, aisles in storage systems are undesirable as nothing may be stored in them. Aisles in traditional storage systems occupy approximately 35% of all the area available for storage (Gue, 2006). In PBS, the escorts may be conceived as movable aisles.

PBS may be controlled by centralized or decentralized control systems. Centralized control PBS require that each cell is connected to a central computer to make decisions considering all the information in the PBS. On the other hand, decentralized control PBS operate by allowing cells to communicate and negotiate as agents with neighboring cells. Centralized control PBS are able to

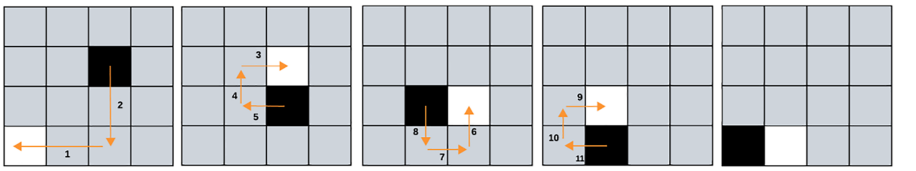

Figure 1. Sequence of PBS movements to retrieve a load with one escort. **Source:** Authors' own creation

determine optimal control strategies for PBS, at the expense of demanding intensive computing power with efficient algorithms and requiring interconnectivity with all cells. In contrast, decentralized control systems do not require as much computation power and limit information exchange to be on-demand and with neighboring cells. In general, centralized control PBS need to be hardwired and are limited to simpler scenarios (e.g. low-throughput systems with fewer I/O points), whereas decentralized control PBS allow plug-and-play functionality and are scalable to more complex scenarios (e.g. high-throughput systems with multiple I/O points).

The main implementation challenge for PBS is the investment (capital) cost. It is observed that reducing the number of pop-up conveyors would simultaneously reduce the investment cost and operational complexity of PBS, at the expense of a lower system throughput. Interestingly, the cost ratio between unidimensional and bidimensional cells is approximately 2:5 (K. Furmans, private communication, June 21, 2023). Hence, this study challenges the premise that every cell in a PBS must have a pop-up conveyor. The tradeoff of eliminating a pop-up conveyor is that load movements would be limited to one dimension (horizontally or vertically), which could affect the expected retrieval time for a load. Cells without a pop-up conveyor are hereafter referred to as *unidimensional cells* since movements are limited to bidirectional travel in only one dimension. A PBS that includes unidimensional cells is henceforth termed *unidimensional PBS* (UPBS).

If the PBS in Figure 1 had unidimensional cells, the proposed retrieval path might not be feasible. For example, if cell in (3,1) was unidimensional only allowing horizontal moves, then the proposed retrieval path would not be feasible. An alternative path would retrieve the load by using the second row and then move south to reach the I/O. Notice that having cell (3,1) as unidimensional also creates a complication for the escort, which would now have to travel from its original position (1,1) $\rightarrow$ (2,1) $\rightarrow (2,2) \rightarrow (3,2)$ for the first load movement or it may take an alternative (and longer) path via (1,1) $\rightarrow$ (4,1) $\rightarrow (4,2) \rightarrow (3,2)$.

The research problem addressed in this paper is to determine how to design UPBS (*i.e.* PBS containing some cells that are limited to movements in the north/south or east/west). This study is a pioneer in considering UPBS. The basic premise in the study is that limiting some grids to unidimensional movement will improve the business case for implementing PBS in terms of the system cost-to-throughput ratio. The remaining of this paper is organized as follows. Section 2 summarizes the relevant academic literature associated with very high-density storage systems. Section 3 presents a mathematical formulation to determine the best path to retrieve a single load using a single escort with a single I/O point for a centralized control UPBS. The optimization problem is solved iteratively to compute the expected optimal retrieval distance for a UPBS design considering a random load and escort location for a 4×4 UPBS. UPBS design insights and cost-to-throughput analyses are derived from these results. In addition, mathematical formulations that extend the problem to consider multiple escorts are presented. Section 4 studies a more complex and practical decentralized control UPBS that retrieves simultaneous loads to multiple I/O points using multiple escorts. A simulation study is performed to evaluate different UPBS designs and gain valuable design insights. Lastly, Section 5 presents the conclusions and future work of this study.

2. Literature review

This section presents and discusses the pertinent PBS academic literature by discussing: centralized control PBS, decentralized control PBS, and three-dimensional (3D) PBS.

2.1 Centralized PBS

The first mention of a system that resembles PBS was in Gue (2006) as an example of a very high-density (VHD) storage system. The paper emphasizes the importance of effectively using space and characterizes the best layouts for VHD storage systems. Gue and Kim (2007) is the first publication specifically addressing PBS. The authors estimate the minimum number of

moves required to retrieve a load to a corner I/O point using a single escort without considering tandem movement. Taylor and Gue (2008) extend the work of Gue and Kim (2007) by quantifying the effects of using different number and location of escorts on the average retrieval time. It is concluded that having more escorts reduces the expected retrieval time of a single load. Gue and Taylor (2014) introduce a type of aisle-less storage concept labeled concentric rings and compare it to the PBS from Gue and Kim (2007) in terms of retrieval time performance. Kota *et al*. (2015) extend the work of Gue and Kim (2007) by developing closed-form expressions to minimize the time a PBS takes to retrieve a single load that is randomly located using one and two escorts without tandem movement.

Zhang *et al*. (2013) propose controlling PBS by implementing the analogous of a class-based storage policy with three classes. Mirzaei *et al*. (2017) study multiple load retrievals in a PBS using a single escort and find that a method that combines loads for tandem retrieval significantly reduced the average retrieval time. Yalcin *et al*. (2019) optimally develop retrieval plans for a single load using multiple escorts with tandem movements.

Ma *et al*. (2022) introduce a beam search algorithm to minimize the total number of moves required for the single-load retrieval problem using multiple escorts. Bukchin and Raviv (2022) develop a 0/1 formulation based on a time-expanded-graph (TEG) for single load retrieval problems involving simultaneous block and load movements. Based on experimental results, they suggest that placing the I/O point in the middle of the edge or employing multiple I/O points is advantageous. Bukchin and Raviv (2023) extend the TEG formulation to include multiple simultaneous retrievals as a variation of the multi-commodity network flow problem.

Other authors considered centralized algorithms for shuttle-based PBS: Raviv *et al*. (2023) propose shuttle-based PBS using autonomous mobile robots (AMRs), and Sgarbossa *et al*. (2023) present a puzzle-based movable rack system for picking that allows Euclidian travel.

It is observed that most existing models in the scientific literature consider as input parameters some pre-determined repositioning movements that might not be feasible in UPBS. For example, some papers implicitly assume that escorts move progressively between moves via a 3- or 5-move repositioning. These moves may not be feasible in UPBS. On the other hand, TEG-based formulations, such as the ones in Bukchin and Raviv (2022) and Bukchin and Raviv (2023), may be manipulated by excluding some arcs from their network to resemble UPBS. Preliminary work related to the single-escort UPBS formulation presented in Section 3.1 was included in Carlo and Blanco (2023) without stating how to compute the required travel distances, without detailing how the formulation may be utilized to generated UPBS design insights, or how to quantify the throughput-to-cost tradeoffs of UPBS.

All papers described above assume that the PBS operates with a centralized control system. The next sub-section presents papers assuming decentralized control systems, where the information is requested and passed on-demand from one cell to another, rather than collecting all the information on a centralized system.

2.2 Decentralized PBS

The main advantage of decentralized control PBS is the ability to develop plug-and-work modules that serve as grid cells. Notice that centralized controls require connecting each cell to a central decision support system. Furmans *et al*. (2010) describe new plug-and-work material handling systems that operate using decentralized decisions, including a PBS. Mayer and Furmans (2010) and Gue *et al*. (2014) describe PBS systems consisting of modularized conveyor-units (equivalent to grid cells) that make decentralized decisions to handle loads while avoiding deadlocks. Similar systems are studied by Krühn *et al*. (2010) and Krühn *et al*. (2016) in terms of using decentralized controls based on the status of neighboring modules and developing algorithms to detect and avoid deadlocks. Furmans *et al*. (2013) extend the work of Gue *et al*. (2014) by introducing an adapted control algorithm to manage a PBS and prevent deadlocks when a conveyor module fails.

Gue and Uludag (2012) and Shekari Ashgzari and Gue (2021) study PBS for picking operations where each cell may contain totes with multiple loads (i.e. mixed totes). Seibold

et al. (2013) focus on using PBS to sort goods under different layouts, assuming a constant work in process (WIP), while Sohrt *et al*. (2014) incorporate time-windows for PBS used for sorting.

Shirazi and Zolghadr (2021) present an algorithm that allows replenishments from all sides of the PBS grid. Assuming the number of requested loads remains constant, the authors conclude via simulation that depending on the grid size and the number of requested loads, increasing the number of escorts might make the retrieval process more complicated. Alahmad and Ishii (2021) use an A* algorithm to, among other things, demonstrate that square grids perform better than other rectangular shapes. He *et al*. (2023) use a deep reinforcement algorithm for the multi-load PBS retrieval problem with multiple escorts and randomly placed I/O points.

Other authors have proposed decentralized control for shuttle-based PBS. Alfieri *et al*. (2012) introduce a version of a shuttle PBS that uses a limited number of Automated Guide Vehicle (AGV) tractors to facilitate rack movements. In their system, the AGVs are not dedicated to the cells, which provide some of the benefits of PBS, without investing too much in automation. Schwab (n.d.) introduces a decentralized control algorithm, LivePath, designed for PBS (referred to as GridFlow systems) operated by AGVs to handle large loads. Instead of relying on path reservations, each AGV determines its next move based solely on the current system state, using a set of priority-based rules to keep the system free of both deadlocks (circular waiting) and livelocks (circular movement). LivePath demonstrated suitable throughput when evaluated against a centralized benchmark, using a mixed-integer programming MIP formulation capable of computing optimal movement sequences for single-AGV scenarios. Trenkle *et al*. (2013) describe an autonomous system consisting of several individual vehicles that mimic PBS. Gue (2016) propose a PBS for handling containers at a rail-to-rail hub for the Physical Internet (PI or π). Sohrt and Overmeyer (2020) study a routing algorithm designed for decentralized and controlled modular conveyors synchronized through the network, with the aim to reserve routes by exchanging messages. Seibold *et al*. (2022) present a decentralized controls system based on the principle of logical time.

Furmans and Gue (2018) provide a general framework for material handling modules, which can store relevant information, contributing to decision-making, and then controlling the execution of these decisions.

2.3 Three-dimensional (3D) PBS

The PBS concept has been further extended to include a third dimension with up/down movements. Zaerpour *et al*. (2010) and Zaerpour *et al*. (2012) are the first to study PBS in three dimensions, called live-cubes, which can be seen as placing several PBS one on top of the other, connected by a lift. Zaerpour *et al*. (2017a) propose an evaluative framework to compare different live-cubes system configurations, while Zaerpour *et al*. (2017b) and Zaerpour *et al*. (2017c) implement a class-based policy for a live-cubes system. Yu and Koster (2012) investigate storage and retrieval sequencing problems in 3D PBS. Hao *et al*. (2015) develop models for designing 3D compact storage systems. Other studies have focused on automated parking systems (e.g. Wu *et al*. (2019), Siddique *et al*. (2021)).

2.4 Research gap

The only published work that implicitly assumes UPBS is Furmans *et al*. (2013), which modifies the decentralized adapted controls algorithm from Gue *et al*. (2014) to continue operating even if some north/south or east/west conveyors fail. In this study the system is not designed to be a UPBS, but rather behaves as one when conveyors fail. The authors report, as an example, that based on an agent-based simulation two failed north/south conveyor failures in the same row have a significantly greater impact on the throughput than two north/south conveyor failures in the same column. The main difference between Furmans *et al*. (2013) and this paper is that they have an algorithm to cope with unreliable conveyors, whereas this study seeks to determine the placement of unidimensional cells in a UPBS. A more detailed literature review on PBS may be found in Blanco (2023).

3. Design of UPBS for a single load with a single escort

Consider an $m \times n$ UPBS grid operating with a centralized control system. The problem under study seeks to determine the best path to retrieve a predetermined load from its original location in the grid to the I/O point with the least amount of horizontal and vertical load movements. The following modeling assumptions are made:

(1) The initial location of the requested load is known

(2) There is only one I/O point, located in any location around the edges of the grid

(3) The travel distance between locations is known and corresponds to rectilinear travel considering unidimensional cells

(4) Each horizontal or vertical movement is of one distance unit (DU)

(5) Loads may only move via escorts

The modeling assumptions made are realistic for a PBS if the performance metric is associated with travel distances, as in our case. However, from a temporal perspective it would be important to incorporate the time delay experienced by loads as a consequence of the conveyor pop-up times (i.e. the time it takes pop-up conveyors to be risen or lowered to enable orthogonal movements). For example, the FlexConveyor in Next Intralogistics (n.d.) requires 0.3 s to lift up or lower down (i.e. 0.6 s per complete lift-return cycle). Clearly, from a temporal perspective, the orientation of the PBS becomes relevant and the problem becomes more complicated.

The UPBS retrieval problem may be formulated as a mathematical program. Section 3.1 presents a linear program (LP) formulation for the single-escort problem and Section 3.2 presents a 0/1 formulation for the multi-escort problem.

3.1 Single-escort single-load formulation

The following nomenclature is defined for the LP formulation:

Sets

$\boldsymbol{N}$ = set of all (grid) cell locations, $\boldsymbol{N} = \{1, \ldots, |\boldsymbol{N}|\}$

$\boldsymbol{J}_k$ = set of neighborhood locations for cell location k, $\boldsymbol{J}_k = \{1, \ldots, |\boldsymbol{J}_k|\}, k \in \boldsymbol{N}$

Indices

i = location of the escort at the beginning of a movement, $i \in \boldsymbol{N}$

k = location of the escort at the end of a movement; which coincides with the location of the load to be retrieved, $k \in \boldsymbol{N}$

h, j = last cell location visited by the escort before reaching location k, $(h, j \in \boldsymbol{J}_k)$

Parameters

s = initial location of the load to be retrieved, $s \in \boldsymbol{N}$

r = location of the I/O, $r \in \boldsymbol{N}$

d_{ijk} = minimum distance for the escort to move from location $i \in \boldsymbol{N}$, using intermediate location $j \in \boldsymbol{N}$, to reach the load located at $k \in \boldsymbol{N}$. The intermediate location is necessary to ensure the minimum distance does not include going directly from location i to location k, as this would imply that the load is moved backwards in the process.

d_{js} = minimum distance for the escort to move from its initial location to the load's initial location $s \in \boldsymbol{N}$ through neighborhood location $j \in \boldsymbol{N}$ in DUs (note the index of the escort's initial location is suppressed)

Decision variables

y_j = Binary variable equal to 1 if escort moves from its initial location to $s \in \boldsymbol{N}$ through location $j \in \boldsymbol{J}_s$, zero otherwise. This variable is relaxed in the formulation.

x_{ijk} = Binary variable equal to 1 if escort moves from location $i \in \boldsymbol{N}$ to the load's location $k \in \boldsymbol{N}$ via intermediate location $j \in \boldsymbol{J}_k$, zero otherwise. This variable is relaxed in the formulation.

(SE)

$$Min\ z_{SE} = \sum_{j \in \boldsymbol{J}_s} d_{js} y_j + \sum_{i \in \boldsymbol{N}} \sum_{j \in \boldsymbol{J}_k} \sum_{k \in \boldsymbol{N}} d_{ijk} x_{ijk} \tag{1}$$

s.t.

$$\sum_{j \in \boldsymbol{J}_s} y_j = 1 \tag{2}$$

$$\sum_{h \in \boldsymbol{J}_j} x_{shj} = y_j \ \forall j \in \boldsymbol{J}_s \tag{3}$$

$$\sum_{i \in \boldsymbol{N}} x_{ijk} = \sum_{h \in \boldsymbol{J}_j} x_{khj} + \sum_{i \in \boldsymbol{N}} x_{irk}$$

$$\forall k \in \boldsymbol{N} \backslash \{s\}, j \in \boldsymbol{J}_k \tag{4}$$

$$\sum_{i \in \boldsymbol{N}} \sum_{k \in \boldsymbol{N}} x_{irk} + y_r = 1 \tag{5}$$

$$0 \le y_j \le 1 \ \forall j \in \boldsymbol{J}_s \tag{6}$$

$$0 \le x_{ijk} \le 1 \ \forall i, k \in \boldsymbol{N}, j \in \boldsymbol{J}_k \tag{7}$$

The objective function of the single escort formulation (SE) in (1) minimizes the number of movements from the escort's initial location to the load's initial location (the first term in the objective function) and all the escort movements from there on until reaching the I/O. Referencing Figure 1, the first term in the objective function would capture the escort movements from its initial location (1,1) to the loads initial location (3,3). The remaining escort movements would be captured by the second term of the objective function. Since an escort movements eventually lead to a load movement, the objective function records the total number of movements required to retrieve the load. Further, since the horizontal and vertical move velocities are assumed constant, the objective function is associated with the total load retrieval time. Therefore, a lower number of total movements to perform a retrieval implies lower retrieval times, which in turn imply a higher UPBS system throughput, which is what this study ultimately intends to quantify. The formulation can consider the case where cell movement is limited to one at a time (a.k.a. load movement) or can allow tandem movements by manipulating the escort travel distances. Also, the formulation allows the I/O to be located at any grid cell. Constraint set (2) ensures that the escort moves from its initial location exactly once. The initial location of the escort is not explicitly specified but is implicitly included in d_{js}. It is noted that if the escort travels to any location via the I/O, the load will end up at the I/O, which concludes the retrieval process. Constraint set (3) ensures the escort repositions for the second load movement, which starts at the escort's ending location after the first load

movement (i.e. the load's initial location) and ends at the intermediate location used to reach location s in the load's first movement. Constraint set (4) is a flow conservation constraint that ensures that if the escort moves a load at location k, the next escort movement needs to start at k or the load must have been delivered at the I/O (i.e. intermediate location r was visited). Constraint set (5) covers a special case when the load is delivered to the I/O on the first move. Constraint sets (6) and (7) define the decision variables. Intuitively, the decision variables should be binary in (6) and (7); however, since the SE formulation is associated with the min-cost flow problem in a node-arc incidence matrix of a directed graph, as represented in Figure 2, the coefficients associated with the restrictions are known to create a Totally Unimodular (TU) matrix. Hence, the decision variables may be relaxed in the SE formulation.

The next sub-section describes the methodology recommended for determining how to obtain the distances for an UPBS.

3.2 Modified Floyd–Warshall algorithm to obtain distances

The SE formulation in Equations (1–7) assumes that values of d_{js} and d_{ijk} are known. These parameters correspond to the minimum distances (measured in number of movements) required to travel within the UPBS. To obtain these parameters a modified version of the Floyd–Warshall (F–W) algorithm [44, p. 251] may be run *a priori*. The F–W algorithm simultaneously finds the shortest path between all pairs of nodes ($\boldsymbol{V}$) with arcs ($\boldsymbol{E}$) in a network $\boldsymbol{G}(\boldsymbol{V},\boldsymbol{E})$ by efficiently solving Dijkstra's algorithm recursively. The complexity of the F–W algorithm is $O(n^3)$.

In this study, the UPBS grid is modeled as a directed graph $\boldsymbol{G}_{UPBS}(\boldsymbol{V},\boldsymbol{E})$. The nodes $i \in \boldsymbol{V}$ of the graph represent the grid cell locations and the $(i,j) \in \boldsymbol{E}$ edges determine if a movement between the cell locations is allowed for the UPBS. As shown in Figure 2, in a regular PBS all cells that are not at the edge of the grid will have four neighbors, one in each direction. Non-corner edge cells will have 3 neighbors, and corner cells will have only 2.

Further, since the distance parameter d_{ijk} represents the minimum distance to travel in the UPBS network from $i \to j \to k$ without traversing directly from $i \to k$, the F–W algorithm has to be modified. To understand the reasoning behind the required modification, assume that an escort in location 6 of the PBS in Figure 2 will be used to move a load in location 10. Clearly, by performing this movement, the load would travel southward from location 10 to location 6. After this movement, the load would end in location 6 and the escort in location 10. Further, assume the next movement of the load was determined to be once again southward to location 2. Then, to ensure a southward movement of the load, the escort will have to travel from location 10 to location 6, via location 2 ($10 \to 2 \to 6$). If one allows the escort to travel using arc $(10,6)$ (i.e. the $i \to k$ movement that we seek to prohibit), then as the escort travels down, the load travels up back to its original location 10. For this reason, the network in the F–W algorithm needs to be modified to exclude the arc (i,k) when computing d_{ijk}.

The corresponding network graph for UPBS would also exclude the arcs not allowable for unidimensional cells, as shown in the example in Figure 3. In Figure 3, the bi-dimensional grids are presented in gray and the unidimensional grids only include the allowable move

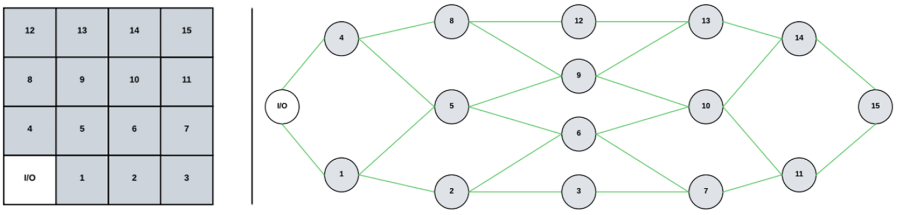

Figure 2. Sample network graph for a 4 × 4 PBS. **Source:** Authors' own creation

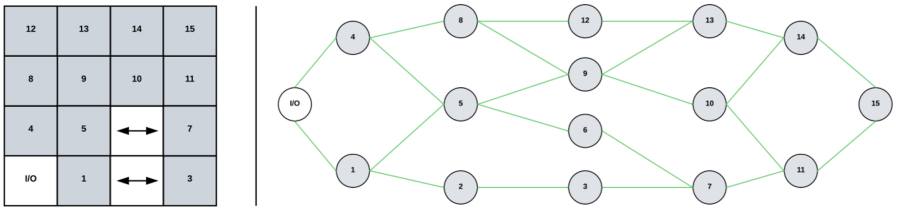

Figure 3. Sample network graph for a 4 × 4 UPBS. **Source:** Authors' own creation

direction. Note that in the UPBS example provided excluded arc $(2,6)$ as grid 2 is unidimensional allowing movement only in the horizontal dimension.

To find d_{ijk}, the F–W algorithm was run for each (j,k) pair with a special network $\boldsymbol{G}_{(j,k)}(\boldsymbol{V},\boldsymbol{E}) \doteq \boldsymbol{G}_{UPBS}(\boldsymbol{V},\boldsymbol{E}) \backslash (k,j)$ (i.e. the UPBS network excluding arc between $k \rightarrow j$). As described before, if edge (k,j) is included in the network, there is a possibility that the shortest path from $i \rightarrow j$ goes through k; implying that the escort passes through the load location in order to be positioned, and in the process moves the load backwards away from the I/O. Let the shortest path solution from $i \rightarrow j$ the F–W algorithm in network $\boldsymbol{G}_{(j,k)}(\boldsymbol{V},\boldsymbol{E})$ be $d_{ij}^{(j,k)}$. Then,

$$d_{ijk} = \begin{cases} d_{ij}^{(j,k)} + 1 \; if \; j \rightarrow k \; \text{exists} \; in \; \boldsymbol{G}_{UPBS}(\boldsymbol{V},\boldsymbol{E}) \\ \infty otherwise \end{cases} \tag{8}$$

3.3 Experimental results single escort problem

This sub-section uses the SE formulation iteratively to obtain design insights for UPBS. The F–W algorithm modification and the SE formulation were coded in Python, which calls the Gurobi Optimizer v9.1.2® to solve the LP. The experiments consider a 4 × 4 UPBS grid. The PBS grid is mapped with location 0 in the lower left corner with coordinates (1,1), with location numbering increasing to the east, then north. For example, location 1 has coordinates (2,1) and location 4 has coordinates (1,2). The I/O is assumed in location 0. This assumption is consistent with papers, although Bukchin and Raviv (2022) suggest that the I/O should be located in the middle of an edge as it reduces the expected retrieval distance for loads.

The experimental methodology starts assuming the requested load is in location 1. The SE formulation is then solved considering that the escort is in each of the 15 possible locations (16 total cells, minus the load's initial location). The average of the retrieval distances represents the expected retrieval distance for a PBS, given the requested load is in location 1. For the load in location 1, the expected load retrieval distance (E[RD]) was 2.80 DU, as shown in Table 1. This process was repeated for each possible load location (15 total as the load in the I/O was excluded). A total of 15 × 15 = 225 instances were run. The average of the 225 instances is the average expected load retrieval distance ($\overline{\text{E[RD]}}_{PBS}$) for the PBS. Table 1 summarizes the results for the traditional (bidimensional) PBS which serves as a baseline for the number of moves required to retrieve the load. The average expected load retrieval distance for the PBS ($\overline{\text{E[RD]}}_{PBS}$) is 9.31 DU.

The experiment continues by considering incorporating one unidimensional cell at a time to the system. Incorporating one unidimensional cell at a time is an iterative greedy approach to find a good (yet not necessarily optimal) solution. Only 12 of the 16 cells are considered as potential unidimensional cells given that corner cells may not be unidimensional because it would be impossible for a single escort to remove a load from a corner in those scenarios. For example, the first potential unidimensional cell to be considered is the one in location 1 (with coordinates (2,1)) in the direction that precludes loads from moving in the east/west dimension. (Recall that the I/O is in location 0.) For this UPBS the average expected load

Table 1. Expected retrieval distance E[RD] for a load at each specific location (in DUs)

Load location	$E[RD]$
1	2.80
2	6.40
3	10.00
4	2.80
5	5.40
6	8.33
7	11.93
8	6.40
9	8.33
10	10.93
11	13.87
12	10.00
13	11.93
14	13.87
15	16.60
$\overline{E[RD]}_{PBS}$	9.31

Source(s): Authors' own creation

retrieval distance is obtained as before by allowing the escort to be in all other cells. The second potential unidimensional cell is the same cell location (location 1) but limiting the north/south dimension. The third potential unidimensional cell is location 2 with limited dimension east/west. In a 4 × 4 PBS grid there are 12 cells that may be unidimensional (four corners excluded), and the unidimensional cells may be in one of two orientations, so a total of 24 layouts are considered. The average expected optimal retrieval distance for the 24 layouts is compared. The unidimensional cell (and orientation) with the minimum expected optimal retrieval distance is considered the most appropriate one to incorporate as the lowest $\overline{E[RD]}$ will provide the highest UPBS throughput among all possibilities.

Given that the best unidimensional cell is implemented, the experiment continues to explore incorporating a second unidimensional cell. At this point, only 22 unidimensional cells are considered (corners and the first unidimensional cell are excluded) for the second unidimensional cell to implement. After the second (Taha, 2006) unidimensional cell was selected based on their $\overline{E[RD]}$, a third unidimensional cell was identified, and so on. Figure 4 summarizes the results of expected optimal retrieval distance as a function of the number of unidimensional cells incorporated. Figure 4 may be interpreted as, by implementing the best unidimensional cell, the expected retrieval distance increases by 2.53% over the original PBS (which is the baseline). After implementing two unidimensional cells, the average expected retrieval distance increases by 5.30% over the PBS. As stated earlier, the proposed strategy that incorporates one unidimensional cell at a time does not guarantee optimality of the resulting UPBS design as not all possible combinations of unidimensional cell designs are considered. On the other hand, the iterative greedy approach is expected to yield good solutions.

It can be observed from Figure 4 that implementing three (of the possible 12) unidimensional cells increases the expected optimal retrieval distance by less than 10%, when compared to the bidimensional PBS. Considering four unidimensional cells affects the expected optimal retrieval distance by less than 15%. With six unidimensional cells the expected optimal retrieval distance increase exceeds 20%.

Figure 5 depicts the final layout with as many unidimensional cells as possible. The bi-dimensional grids are presented in gray and the unidimensional grids only include the allowable move direction. The boxed number included in the unidimensional cells indicates

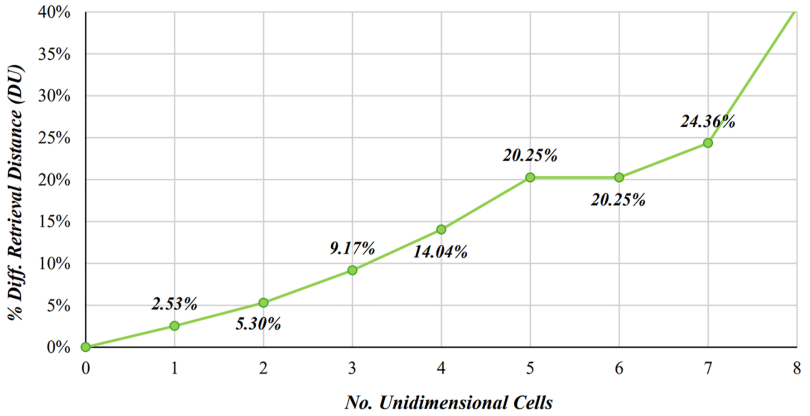

Figure 4. Expected optimal retrieval distance as a function of number of unidimensional cells. **Source:** Authors' own creation

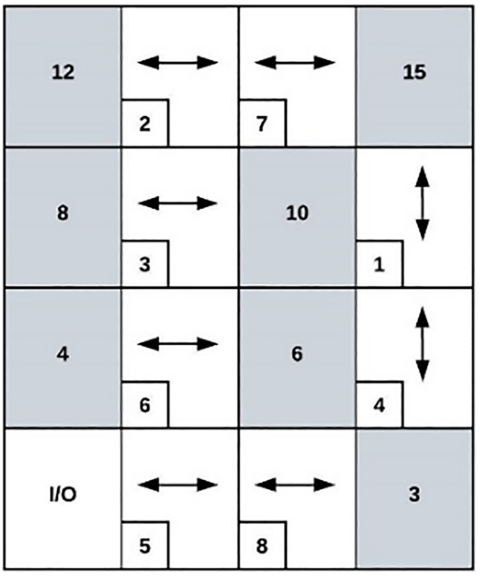

Figure 5. UPBS layout design progression. **Source:** Authors' own creation

the stage in which it was incorporated (1 being the first unidimensional cell added). This number also corresponds to the abscissa in Figure 4. The maximum number of unidimensional cells is 8. Adding a ninth unidimensional cell renders an infeasible design as the I/O would be unreachable from some cells. There were instances where two unidimensional cells resulted in the same expected retrieval distance; in those cases, the more intuitive choice was selected by hand. In all these cases it was observed that the unidimensional cell not selected was selected in the next stage.

The UPBS layout in Figure 5 appears to have created north-south corridors in the first and last columns and east-west corridors in the other columns. This layout may be used to guide the

design of UPBS considering the tradeoff between cost and expected retrieval distance from Figure 4.

To quantify the cost-to-throughput tradeoff of UPBS, let c_b be the fixed cost of each bidimensional cell (or cell module). Assuming a unidimensional: bidimensional cell cost ratio of approximately 2:5, the 4×4 grid design with 3 unidimensional cells will cost $1.8c_b$ less than the PBS, at the expense of an $\overline{E[RD]}$ increase of by 9.17%, or equivalently, a 8.34% average reduction in system throughput.

When extrapolating the design insights for larger PBS grids one would expect a similar pattern; corner cells bidimensional, non-corner edge cells would be the first to be considered unidimensional, and corridors will be created for loads and escorts to navigate. Figure 6 presents the hypothesized UBPS layout design progression for a single-escort 5×5 grid design.

For a 5×5 UPBS, the first two cells to become unidimensional should be the ones adjacent to the corner cells in the top row, consistent with the empirical results for the 4×4 design. Notice that the first unidimensional cell with coordinates (5,4) in Figure 6 barely has an effect on the retrieval of loads. Similarly, incorporating the second unidimensional cell with coordinates (2,5) only limits the retrieval flow of a few loads that now are forced to travel in the horizontal direction. Since cell with coordinates (2,5) becomes unidimensional, there is little value-added in allowing cell (2,4) to have vertical movement. The fourth cell to be unidimensional should be the one with coordinates (5,2) as the last column mainly becomes a vertical corridor. Preferring cell with coordinates (5,2) to the one in (5,3) is to allow the escort to escape the corridor more easily. Figure 6 presents the hypothesized sequence for incorporating unidimensional cells. As in the 4×4 case, more unidimensional cells may be incorporated, at the expense of throughput. These design insights may be extrapolated for designing larger single-escort UPBS.

3.4 Multi-escort problem formulation

Unfortunately, the LP for the SE formulation cannot be directly extended to the multi-escort problem. Instead, the 0/1 formulation based on a time-expanded graph (TEG) from Bukchin and Raviv (2023) may be modified to consider UPBS. The required modification eliminates from the TEG graph those edges corresponding to unidimensional cells. The formulation already allows individual or tandem movements and has the makespan (equivalent to the load

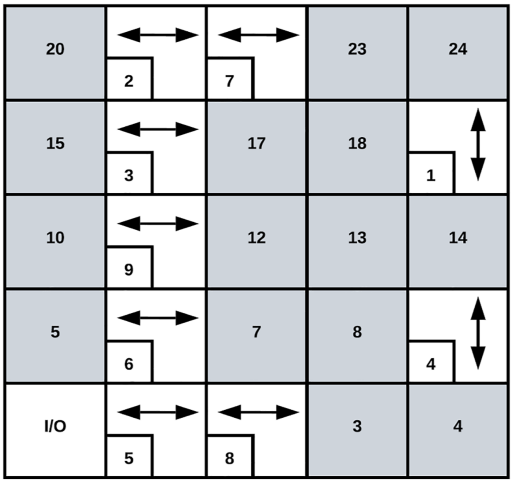

Figure 6. Hypothesized 5×5 UPBS layout design progression. **Source:** Authors' own creation

retrieval time for a single-load instance) as a weighted term in the objective function. The modified formulation is not included for practicality. Clearly, the TEG formulation may also be used for the single-escort problem. However, the SE formulation presented is much faster. The main limitation of the TEG formulation is its sensitivity to an input parameter T, which is an upper bound on the maximum number of steps to retrieve the load. If T is not large enough, the load would not have enough time to exit the system; if T is too large, the runtime to solve the TEG grows quickly. An alternative to obtain an upper bound for the T parameter in the TEG formulation is to solve the single escort (SE) formulation to determine the number of movements required to retrieve the load.

An alternative linear MIP formulation for the multi-escort problem is presented in Appendix A. The formulation, labeled ME, finds the optimal cell movements to retrieve a load, given that the retrieval path is known. The ME formulation may use the resulting retrieval path from the SE formulation, in which case the solution would not be guaranteed to be globally optimal. However, it would provide a faster solution than the TEG formulation. Unfortunately, experimental results such as the ones from Section 3.3 for the multi-escort problem would require that all combinations of the initial escort locations be considered and the required runtimes become impractical even with two escorts. Since the focus of this paper is on the design of UPBS, no experiments were performed to compare the two alternatives for solving the single-load multi-escort UPBS.

In general, UPBS should be designed assuming multiple escorts, as they are simply unused storage capacity. Therefore, the number of escorts in the UPBS is expected to change over time. It is hypothesized that UPBS designs based on insights from the single-escort case should to be effective for multi-escort UPBS. In theory, for multi-escort UPBS, the escorts may collaborate to retrieve a load from a unidimensional corner cell. However, no UPBS should have unidimensional corner cells as it might become inoperable when storage capacity is at maximum (i.e. a single escort is left). Interestingly, if multiple loads are retrieved simultaneously, then the distribution of escorts to retrieval loads becomes an important factor. Results from Mirzaei *et al.* (2017) suggest that retrieving multiple loads in tandem is effective. Also, results from Bukchin and Raviv (2022) suggest that retrieval times for retrieving a single load with multiple loads improve up to the fourth escort – beyond that it becomes negligible. Therefore, it is possible that the optimal retrieval strategy evenly distributes escorts among retrieval loads and then seek to merge them at some point in the retrieval path so all escorts may collaborate to retrieve them in tandem. Interestingly, the UPBS design proposed in Section 3.3 based on empirical results tends to create corridors, which should serve as merging points to retrieve loads in tandem.

4. Design of UPBS with simultaneous loads, multiple escorts, and multiple I/O points

In practice, PBS store and retrieve multiple loads in parallel using multiple escorts and I/O points. Although the TEG model from Bukchin and Raviv (2023) may be modified to allow simultaneous retrievals, obtaining design insights using the model is limited given the complexity of the experiments.

Fortunately, Krühn *et al.* (2016) provide a decentralized control algorithm to simultaneously replenish and retrieve multiple loads in a PBS using multiple escorts and I/O points in a manner that avoids deadlocks even when some PBS cells conveyors fail (i.e. unreliable conveyors). This section will use their algorithm to develop managerial insights for the design of UPBS. Unfortunately, Furmans *et al.* (2013) does not provide enough information to validate a simulation using their algorithm. However, their approach is a straightforward extension of the decentralized algorithm used in Gue *et al.* (2014), which does provide empirical results that may be used to validate a simulation of their approach. Hence, the method employed is to replicate the decentralized algorithm in Gue *et al.* (2014) and validate it against their results. At that point, the decentralized algorithm can be easily extended following the description in Furmans *et al.* (2013).

In general, Gue *et al*. (2014) proposes a decentralized algorithm that requires at least one escort per row. The PBS input points are the row of cells at the top of the grid and the output points are the row of cells at the bottom of the grid. The algorithm allows multiple requests at the same time and assumes that exiting loads are immediately replenished with a new load that needs to move to the row of origin of the retrieved load. Furmans *et al*. (2013) extend this work by considering that some cells fail in a particular dimension for a random amount of time, until they are repaired. Their algorithm maintains the PBS running without deadlocks until repairs occur.

Our implementation of the decentralized algorithm in Gue *et al*. (2014) was done in Python using the Mesa package. The resulting code was validated visually and pushed to the limit in terms of required throughput to confirm that the algorithm avoids deadlocks as suggested by the authors. At that point, experiments were conducted with the following modeling assumptions:

(1) The PBS grid dimensions are $12 \times (12+ k)$ cells

(2) Once a load is requested, it must travel to any cell in the bottom row of the PBS (i.e. an output point), where it will immediately exit the PBS.

(3) The northernmost row of the PBS corresponds to the replenishment row. Once a load exits the system, a new replenishment load is immediately created and assigned to the input point with an available escort, and if none is available, the replenishment load is assigned to a random input point.

(4) Replenishment loads are to be stored in the row of origin of the corresponding retrieval request.

(5) When initiating a retrieval request, the system selects randomly from among loads not currently requested, which can already be in their assigned (home) row or still in transit to their home from the input point.

The Python code runs for 5,760 time steps (corresponding to an 8-h-shift and a 5 s per time step cycle time), considering a warm-up period of 200 steps and 30 replications of each run, as done in the experiments of Gue *et al*. (2014). Figure 7 presents the average throughput of the PBS considering 1, 2, and 3 escorts per row (labeled *k* to use their notation) under different number of active requested loads (i.e. average WIP per their notation).

Figure 7 closely resembles the results from Figure 8 in Gue *et al*. (2014) for the same experiments. There is a slight deviation between our results and the results from Gue *et al*. (2014) when the WIP exceeds 110 active loads, corresponding to 76.39% of the total loads

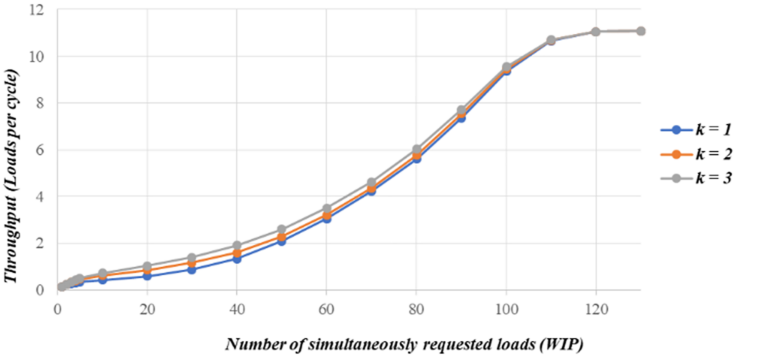

Figure 7. Average throughput for systems with different active load requests. **Source:** Authors' own creation

1315	3692	5611	7148	8295	8992	9240	9098	8514	7386	5828	3844	1358
1554	4334	6568	8369	9615	10395	10640	10480	9865	8598	6766	4431	1559
1472	4141	6317	8063	9385	10189	10448	10251	9596	8341	6575	4359	1550
1529	4237	6442	8190	9408	10150	10408	10240	9571	8353	6624	4353	1526
1546	4324	6612	8350	9606	10432	10713	10519	9818	8574	6752	4424	1573
1570	4410	6725	8514	9798	10594	10888	10685	9885	8567	6775	4460	1579
1586	4497	6831	8648	9979	10774	11058	10794	10039	8790	6952	4586	1618
1620	4604	6971	8783	10034	10769	11013	10791	10080	8801	6977	4603	1629
1642	4649	7043	8902	10215	10986	11226	10933	10200	8954	7075	4647	1656
1728	4819	7222	9066	10406	11207	11500	11229	10463	9168	7255	4757	1687
1725	4825	7264	9164	10525	11340	11608	11422	10709	9406	7437	4841	1685
1759	4931	7465	9434	10813	11671	11994	11725	10953	9571	7529	4908	1728

Bidimensional PBS with k = 1 and WIP = 1

(a)

1392	3990	6054	7655	8985	9810	10308	10477	10249	9401	7846	5600	2065
1460	4170	6372	8182	9642	10600	11041	11108	10714	9788	8135	5658	2091
1453	4173	6426	8295	9676	10610	11233	11396	10968	9884	8136	5646	2108
1491	4146	6322	8230	9802	10861	11434	11561	11125	9946	8045	5531	2061
1589	4372	6623	8564	10055	11128	11827	12024	11556	10402	8544	5946	2187
1663	4673	7065	9055	10653	11905	12598	12570	12019	10866	8991	6285	2346
1771	4965	7558	9646	11281	12552	13240	13207	12618	11410	9435	6669	2517
2037	5506	8269	10571	12304	13621	14221	14157	13551	12262	10184	7171	2698
2160	5926	8943	11442	13312	14631	15367	15385	14691	13201	10944	7662	2830
2386	6455	9681	12306	14335	15808	16507	16511	15815	14188	11745	8316	3110
2555	6968	10608	13384	15513	16966	17814	17909	17212	15499	12847	9047	3384
2791	7573	11256	14264	16638	18274	19138	19279	18538	16801	13951	9825	3670

Bidimensional PBS with k = 1 and WIP = 10

(b)

867	2433	3743	4784	5512	6000	6299	6271	5968	5463	4711	3512	1406
858	2360	3669	4767	5491	6064	6405	6483	6170	5596	4719	3418	1342
799	2298	3660	4758	5509	6063	6375	6362	6028	5486	4622	3329	1255
806	2409	3780	4926	5653	6104	6428	6535	6315	5722	4835	3490	1302
846	2499	3817	4988	6052	6719	6989	7042	6857	6204	5237	3702	1382
970	2784	4281	5535	6511	7230	7620	7729	7553	6841	5789	4181	1527
1069	3114	4822	6320	7380	8208	8700	8747	8500	7778	6545	4709	1792
1323	3709	5706	7262	8453	9310	9909	9989	9615	8761	7446	5503	2135
1547	4374	6652	8372	9754	10707	11246	11392	11051	10115	8651	6423	2432
1824	5141	7637	9661	11264	12457	13048	13110	12784	11774	10055	7352	2794
2126	6035	9086	11390	13211	14544	15258	15308	14971	13737	11583	8416	3203
2545	7112	10447	13117	15271	16797	17678	17852	17425	16007	13615	9779	3715

Bidimensional PBS with k = 1 and WIP = 20

(c)

623	1757	2581	3145	3558	3859	3967	3935	3936	3779	3315	2557	1001
632	1776	2716	3375	3738	3972	4127	4114	3951	3606	3099	2410	920
625	1756	2661	3271	3617	3806	3948	3897	3758	3450	3007	2223	862
616	1863	2717	3346	3699	3928	4091	4030	3844	3635	3150	2391	927
679	1990	2985	3684	4180	4422	4548	4449	4292	4022	3433	2527	994
819	2292	3422	4203	4712	5082	5151	5025	4878	4500	3871	2935	1148
922	2652	4022	4992	5566	5913	6122	5977	5782	5349	4667	3558	1354
1071	3168	4728	5789	6580	7122	7332	7209	6962	6444	5594	4265	1665
1377	3930	5761	7179	8065	8598	8710	8633	8406	7881	6845	5187	2056
1679	4790	7046	8786	9862	10520	10766	10712	10337	9774	8555	6438	2538
2119	5940	8745	10890	12243	13061	13453	13449	12886	12017	10491	7954	3093
2652	7233	10560	13033	14763	15905	16631	16807	16430	15470	13493	9987	3839

Bidimensional PBS with k = 1 and WIP = 30

(d)

Figure 8. Heatmaps for bidimensional PBS for different WIP levels. **Source:** Authors' own creation

being simultaneously requested. It is suspected that the discrepancy corresponds to an interpretation of how to assign input points to replenishment loads. Since the discrepancy favors our implementation, the matter was not investigated further. It is concluded that the Python implementation validates against the AnyLogic simulation results from Gue *et al*. (2014). The Python implementation was then extended to include the modifications described in Furmans *et al*. (2013). The only difference from the implementation from Furmans *et al*. (2013)

is that they assume the unreliable cells fail randomly, whereas in our model, the cells may be unidimensional by design. Henceforth, the Python simulation is referred to as the UPBS simulation.

4.1 Experimental results for the UPBS simulation

The UPBS simulation was run iteratively to evaluate different UPBS designs. The experiments consider a 12×13 UPBS grid with $k = 1$ escorts per row (i.e. 12 escorts in total). Higher level of escorts per row were not considered as PBS are designed for very high-density. Gue and Kim (2007) suggest that PBS become required for storage densities above 90%. The WIP levels (i.e. the number of active requested loads) considered were 1, 10, 20, and 30.

Given that the input points are on the top row and output points are on the bottom row, it was observed that eliminating north-south movements for any cell is highly disadvantageous compared to eliminating east-west movements. Hence, only eliminating east-west movement capability was considered (i.e. unidimensional cells only include north-south movements) in the experiments.

Initially, a traditional (bidimensional) PBS design was run with the UPBS simulation to establish a baseline in terms of the number of horizontal movements (i.e. horizontal movements of requested or non-requested loads) that move through each cell. The average number of times (based on 30 runs performed) a load moves horizontally through each cell is presented as a heatmap for different WIP levels in Figure 8. The heatmaps serve as a visualization tool to identify the cells with more horizontal movements (red) and those with less (green). Cells with heatmap gradient of green are considered good candidates for unidimensional cells.

For example, Figure 8b ($k = 1$ and WIP $= 10$) suggests that the cell with an average number of horizontal movements of 1,392 (cell location (1, 12)) is the best candidate to implement as a unidimensional cell. Notice that this implies that this cell is the least used. Once the best unidimensional cell was implemented, the experiment continued by exploring the implementation of a second unidimensional cell (assuming the UPBS now included the previously implemented unidimensional cell). Therefore, only 155 ($155 = 12 \times 13 - 1$) unidimensional cells are considered (the first unidimensional cell implemented is excluded). Once the second unidimensional cell is selected, a third unidimensional cell is identified, and so on.

Given the order in which unidimensional cells were incorporated, it was possible to identify a pattern. Once a unidimensional cell was incorporated, the remainder of the cells in that row became unidimensional. This result was corroborated empirically through multiple experiments. Hence, the experiment continues by implementing one unidimensional row at a time for each WIP level until the system throughput percent difference compared to the bidimensional PBS exceeded 10%. Table 2 summarizes the results of the system throughput as a function of the number of unidimensional rows incorporated for each WIP level. Figure 9 can be interpreted as, for WIP $= 1$, implementing the best unidimensional row, the throughput increases by 5.69% with respect to the original PBS. After implementing two unidimensional rows, the throughput increases by 7.77% over the PBS. With the addition of three and four unidimensional rows the throughput increases in the proportion corresponding to 9.82% over the original PBS.

Results indicating that implementing UPBS cells increases throughput is counterintuitive. The reason for the throughput increase is the gridlock prevention strategy from the decentralized algorithm from Furmans *et al*. (2013) as the algorithm "sacrifices" loads immediately south of the requested load under some circumstances to prevent gridlocks by creating an escort at the bottom of the column. In the simulation, the sacrificed load is not included in the throughput calculations and the load is immediately replenished to the system through the input point. However, the efficiency of a sacrifice load in generating escorts in the proper column (notice that east/west movements might be limited) results in an increase of

Table 2. Percentage difference of throughput as a function of number of unidimensional rows for different WIP levels

No. unidimensional rows	WIP 1	10	20	30
0	0%	0%	0%	0%
1	5.69%	4.54%	7.47%	*N/A*
2	7.77%	3.43%	*N/A*	*N/A*
3	9.42%	7.71%	*N/A*	*N/A*
4	9.82%	5.82%	*N/A*	*N/A*
5	*N/A*	9.30%	*N/A*	*N/A*

Source(s): Authors' own creation

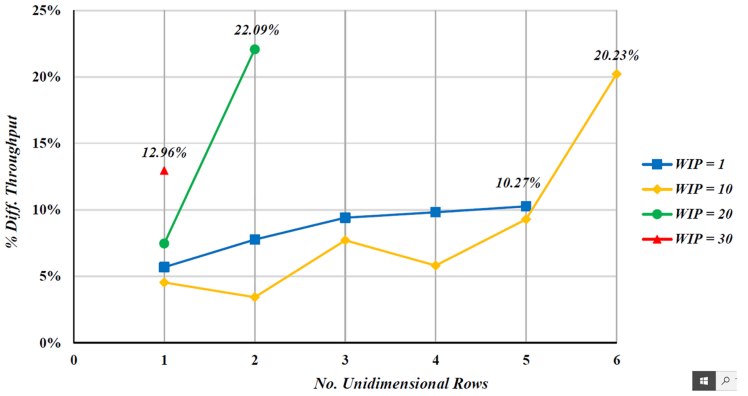

Figure 9. Percentage difference of throughput as a function of number of unidimensional rows for different WIP levels. **Source:** Authors' own creation

throughput. Technically, the simulation results suggest that it is very efficient in terms of throughput to remove all loads south of the requested load and send them back to the system as replenishment. Without the arbitrary 10% throughput increase threshold, the system is expected to continue incorporating unidimensional cells until the entire system is unidimensional, at which point the UPBS will resemble an automatic dispenser. In practice, this strategy would be limited by the conveyor load capacity going from the output to the input points as it would eventually block the system.

It can be observed from Table 2 and Figure 9 that for a WIP $= 10$ implementing the best unidimensional row out of the possible 156 possible cells increases the throughput by less than 5% when compared to the bidimensional PBS. That is, by implementing 8.33% of the potential unidimensional cells, the throughput increases, on average, by 4.54%. Considering two unidimensional rows increased throughput by less than 4%. With five unidimensional rows, the throughput increases by 9.5%.

Figure 10 depicts the heatmap for the final UPBS design with as many unidimensional rows as possible for $k = 1$ and different WIP levels. The unidimensional grids presented (white with the number 0) only allow north/south movements. The number included to the left of each unidimensional row indicates the stage in which they were incorporated (1 being the first unidimensional cell row). This number also corresponds to the abscissa in Figure 9. The maximum number of unidimensional cells depends on the WIP level. Note

1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0
	1669	4609	6875	8609	9769	10430	10736	10487	9771	8551	6850	4598	1653
	1914	5302	7909	9813	11152	11962	12246	12034	11265	9910	7992	5386	1910
	2026	5617	8404	10523	12007	12911	13325	13004	12152	10728	8597	5789	2085
	2197	6083	9111	11420	13133	14145	14560	14311	13413	11796	9459	6388	2300
	2304	6438	9707	12177	13934	15024	15453	15225	14298	12539	10048	6737	2410
	2277	6467	9750	12256	14070	15170	15555	15322	14373	12657	10084	6736	2426
	2322	6543	9828	12278	14042	15173	15583	15268	14310	12650	10134	6790	2434
	2210	6246	9481	11984	13816	14928	15280	15027	14083	12434	9958	6684	2394

Heatmap for final UPBS with WIP = 1

(a)

1	0	0	0	0	0	0	0	0	0	0	0	0	0
	1896	4904	7234	9026	10348	11436	12257	12736	12658	12010	10516	7772	2979
	2092	5456	8186	10499	12319	13742	14817	15564	15645	14766	12829	9264	3543
3	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0
	2467	6507	9567	11679	13250	14261	14888	15059	14718	13839	12054	8779	3313
4	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0
	1957	5379	7882	9647	10818	11527	11833	11781	11326	10596	9192	6890	2708
	2121	5863	8650	10754	12280	13332	13980	14016	13478	12375	10763	8006	3140
	2428	6597	9897	12400	14273	15478	16126	16226	15702	14590	12734	9323	3579
	2819	7811	11662	14645	16787	18296	19243	19479	19026	17581	15026	10987	4189

Heatmap for final UPBS with WIP = 10

(b)

2	0	0	0	0	0	0	0	0	0	0	0	0	0
	1418	3807	5722	7220	8351	9103	9560	9731	9321	8532	7278	5324	2047
1	0	0	0	0	0	0	0	0	0	0	0	0	0
	1306	3607	5298	6629	7659	8140	8269	8217	7844	7064	6035	4451	1770
	1184	3364	5069	6388	7321	7771	8016	8015	7621	6871	5732	4129	1616
	1200	3474	5318	6575	7508	8212	8473	8294	7914	7156	6119	4517	1734
	1392	3790	5738	7099	7984	8617	8905	8874	8506	7749	6636	4858	1852
	1490	4146	6109	7678	8718	9363	9712	9758	9410	8633	7448	5433	2086
	1808	4940	7282	8857	9923	10620	10944	11038	10602	9739	8404	6292	2468
	1998	5593	8150	10003	11259	12118	12627	12638	12245	11407	9955	7422	2874
	2426	6629	9545	11659	13119	14212	14828	15021	14690	13710	12013	8925	3454
	2871	7718	11160	13672	15643	17076	17883	18254	17793	16637	14553	10743	4127

Heatmap for final UPBS with WIP = 20

(c)

2	0	0	0	0	0	0	0	0	0	0	0	0	0
	1520	3886	5489	6443	7119	7313	7348	7236	6940	6452	5787	4570	1838
	1280	3457	5024	5894	6491	6832	6858	6767	6423	5907	5117	3825	1459
1	0	0	0	0	0	0	0	0	0	0	0	0	0
	1295	3274	4474	5284	5805	6068	5964	5789	5457	5113	4504	3409	1325
	1182	3146	4390	5257	5839	6001	5947	5786	5469	5080	4424	3358	1313
	1182	3274	4724	5698	6211	6516	6507	6286	5880	5448	4898	3755	1436
	1486	3888	5511	6554	7074	7312	7209	6985	6644	6279	5689	4428	1718
	1680	4622	6519	7666	8266	8472	8487	8327	8015	7534	6708	5274	2125
	2038	5504	7820	9252	10080	10482	10493	10212	9866	9156	8161	6518	2635
	2671	6963	9745	11499	12573	13029	13053	12766	12301	11630	10558	8460	3407
	3202	8556	11762	13947	15503	16366	16551	16465	16157	15395	14050	11146	4464

Heatmap for final UPBS with WIP = 30

(d)

Figure 10. Heatmaps for final UPBS layout for different WIP levels. **Source:** Authors' own creation

that in Figure 9, the stage in which the percentage difference in throughput increased by 10% or more was included for each WIP level, so in the final heatmap layouts (Figure 10) this iteration is not considered.

The pattern for selecting unidimensional cells in Figure 10a is evident – i.e. sequential from the top and full rows become unidimensional. The UPBS layouts in Figure 10b seem to have created horizontal corridors between unidimensional rows to facilitate north-south movements. For WIP = 20 (in Figure 10c) the only unidimensional rows that may be

incorporated and still remain under the threshold value of 10% is the third and then the first from the top. Interesting, this leaves a horizontal corridor in the second row from the top to favor replenishments. Clearly, the quantity of replenishments is directly proportional to the WIP levels. Hence, the horizontal corridor between unidimensional rows intuitively makes sense to position inbound loads. Lastly, for WIP = 30 two rows were left as corridors between unidimensional rows. The logical argument for WIP = 20 also intuitively justifies the result. We can safely state that for higher WIP levels, the first unidimensional row would be among the first 4 rows from the top. Clearly, the number of unidimensional rows depends on the trade-off between cost and throughput.

5. Conclusions and future work

This study considers PBS where some cells are limited to horizontal or vertical movements (i.e. unidimensional cells). PBS with some unidimensional cells is referred to as UPBS. This study presents a LP formulation to find the optimal load retrieval path using a single escort in a UPBS. The LP is solved recursively to determine the expected optimal retrieval distance for a layout considering a random load and escort location for a 4×4 PBS. The unidimensional cell with the least impact on the expected optimal retrieval distance is incorporated into the design, and the process is repeated to progressively determine the next unidimensional cells to incorporate. The maximum number of unidimensional cells that could be implemented was 8 cells. At that point, no more unidimensional cells may be added as some cells became unreachable by the escort. It is concluded that a layout including 3 unidimensional cells (25% of the candidate unidimensional cells) affected the expected optimal retrieval distance by less than 10%, compared to the bidimensional PBS. Adding four unidimensional cells (33% of the candidate unidimensional cells) affected the expected optimal retrieval distance by less than 15%. Considering six unidimensional cells further increased the expected optimal retrieval distance by less than 20%. As an example of the cost-to-throughput tradeoff, a PBS with 3 unidimensional cells would have a cost reduction, compared to the bidimensional PBS, of 1.8 times the capital cost of a bidimensional cell, at the expense of an 8.34% average reduction in throughput. It is argued that multi-escort UPBS designs should be similar to the ones empirically produced for single-escort UPBS.

A modification from the TEG-based formulation in Bukchin and Raviv (2023) and a new MIP formulation that assumes the path of the load is known were presented for the multi escort problem to retrieve a single load. It is argued that the MIP formulation and the single-escort LP may be used to determine a lower bound for the critical T parameter in the TEG formulation from Bukchin and Raviv (2023). Unfortunately, both models were deemed too complex to provide UPBS design insights. Instead, the decentralized PBS algorithm from Furmans *et al*. (2013) was modified to design UPBS considering simultaneous retrievals, multiple escorts, and multiple I/O points. Upon proper validation of the algorithm, a simulation was used to evaluate UPBS by evaluating the average number of horizontal movements in each cell under different WIP levels for a 12×13 UPBS. The unidimensional cell with the least impact on the throughput was incorporated into the design, and the process is repeated to progressively determine the next unidimensional cell to incorporate. It was observed that if one cell becomes unidimensional, the remaining cells of that row became unidimensional before cells in other rows. The maximum number of unidimensional cells depends on the WIP level. It is concluded that for a low WIP level of 1, it is possible to design UPBS with 33.33% unidimensional cells, while increasing the throughput by 9.82%. The counterintuitive result that throughput increased by incorporating unidimensional cells is attributed to the decentralized algorithm which is designed to "sacrifice" loads under the requested loads as part of its deadlock prevention protocol. For a medium WIP level of 10, a layout including one row of unidimensional cells increased the throughput by less than 5% when compared to the bidimensional PBS. Considering two unidimensional rows increased the throughput by less than 4%. With five

unidimensional rows, the throughput increased by 9.3%. For a WIP level of 20, incorporating one unidimensional row increases throughput by 7.47%.

By incorporating one unidimensional cell at a time, the paper uses an iterative greedy approach to finding a good (yet not necessarily optimal) solution. However, experimental results suggest that the UPBS designs obtained have a reasonable tradeoff between capital investment cost and throughput. Hence, UPBS may help develop a business case for the PBS technology.

Future work on single-load retrieval may explore the effect of alternative I/O locations on the UPBS design. For instance, placing the I/O in the middle of an edge, as suggested in Bukchin and Raviv (2022), may improve performance and yield interesting symmetric UPBS designs. Alternatively, studying UPBS designs with input and output points on opposite sides of the grid, as in staging applications, could provide interesting design insights and applications.

For multiple-load retrievals, future work should consider developing a decentralized control algorithm specifically for UPBS to reduce reliance on sacrifice moves as a deadlock prevention strategy. In general, sacrifice moves should be explicitly modeled to better understand their impact on the system input points when they reenter the UPBS. Lastly, future research may incorporate UPBS designs for multidimensional PBS (i.e. live-cube storage) or incorporate additional performance metrics into the objective function.

References

Alahmad, R. and Ishii, K. (2021), "A puzzle-based sequencing system for logistics items", *Logistics*, Vol. 5 No. 4, p. 76, doi: 10.3390/logistics5040076.

Alfieri, A., Cantamessa, M., Monchiero, A. and Montagna, F. (2012), "Heuristics for puzzle-based storage systems driven by a limited set of automated guided vehicles", *Journal of Intelligent Manufacturing*, Vol. 23 No. 5, pp. 1695-1705, doi: 10.1007/s10845-010-0471-7.

Blanco, A. (2023), "Design and optimization of puzzle-based storage systems with unidimensional movements", Master's Thesis, University of Puerto Rico – Mayagüez.

Bukchin, Y. and Raviv, T. (2022), "A comprehensive toolbox for load retrieval in puzzle-based storage systems with simultaneous movements", *Transportation Research Part B: Methodological*, Vol. 166, pp. 348-373, doi: 10.1016/j.trb.2022.11.002.

Bukchin, Y. and Raviv, T. (2023), "An efficient MILP formulation for the parallel load retrieval in puzzle based storage systems with simultaneous load movements", in *Progress in Material Handling Research*, [Online], available at: https://digitalcommons.georgiasouthern.edu/pmhr_2023/10

Carlo, H. and Blanco, A. (2023), "Designs of puzzle based storage systems with unidimensional cells", *16th Proceedings (Dresden, Germany- 2023). 2*, Progress in Material Handling Research, [Online], available at: https://digitalcommons.georgiasouthern.edu/pmhr_2023/2

Furmans, K. and Gue, K. (2018), "A framework for modeling material handling with decentralized control", in *Progress in Material Handling Research*, [Online], available at: https://digitalcommons.georgiasouthern.edu/pmhr_2018/23

Furmans, K., Schonung, F. and Gue, K. (2010), "Plug-and-work material handling systems", in *Progress in Material Handling Research: 2010*, [Online], available at: https://digitalcommons.georgiasouthern.edu/pmhr_2010/1

Furmans, K., Gue, K.R. and Seibold, Z. (2013), "Optimization of failure behavior of a decentralized high-density 2D storage system", in Kreowski, H.-J., Scholz-Reiter, B. and Thoben, K.-D. (Eds), *Dynamics in Logistics*, Springer, Berlin, Heidelberg, pp. 415-425, doi: 10.1007/978-3-642-35966-8_35.

Gue, K.R. (2006), "Very high density storage systems", *IIE Transactions*, Vol. 38 No. 1, pp. 79-90, doi: 10.1080/07408170500247352.

Gue, K. (2016), "A high-density, puzzle-based system for rail-rail container transfers", in *Progress in Material Handling Research: 2016*, [Online], available at: https://digitalcommons.georgiasouthern.edu/pmhr_2016/14

Gue, K.R. and Kim, B.S. (2007), "Puzzle-based storage systems", *Naval Research Logistics (NRL)*, Vol. 54 No. 5, pp. 556-567, doi: 10.1002/nav.20230.

Gue, K. and Taylor, G. (2014), "Comparison of alternative configurations for dense warehousing systems", in *Progress in Material Handling Research: 2014*, [Online], available at: https://digitalcommons.georgiasouthern.edu/pmhr_2014/27

Gue, K. and Uludag, O. (2012), "A high-density, puzzle-based order picking system", in *Progress in Material Handling Research: 2012*, [Online], available at: https://digitalcommons.georgiasouthern.edu/pmhr_2012/19

Gue, K.R., Furmans, K., Seibold, Z. and Uludag, O. (2014), "GridStore: a puzzle-based storage system with decentralized control", *IEEE Transactions on Automation Science and Engineering*, Vol. 11 No. 2, pp. 429-438, doi: 10.1109/TASE.2013.2278252.

Hao, J., Yu, Y. and Zhang, L.L. (2015), "Optimal design of a 3D compact storage system with the I/O port at the lower mid-point of the storage rack", *International Journal of Production Research*, Vol. 53 No. 17, pp. 5153-5173, doi: 10.1080/00207543.2015.1005767.

He, J., Liu, X., Duan, Q., (Victor) Chan, W.K. and Qi, M. (2023), "Reinforcement learning for multi-item retrieval in the puzzle-based storage system", *European Journal of Operational Research*, Vol. 305 No. 2, pp. 820-837, doi: 10.1016/j.ejor.2022.03.042.

Kota, V., Taylor, D. and Gue, K. (2015), "Retrieval time performance in puzzle-based storage systems", *Journal of Manufacturing Technology Management*, Vol. 26 No. 4, pp. 582-602, doi: 10.1108/JMTM-08-2013-0109.

Krühn, T., Falkenberg, S. and Overmeyer, L. (2010), "Decentralized control for small-scaled conveyor modules with cellular automata", *2010 IEEE International Conference on Automation and Logistics*, pp. 237-242, doi: 10.1109/ICAL.2010.5585288.

Krühn, T., Sohrt, S. and Overmeyer, L. (2016), "Mechanical feasibility and decentralized control algorithms of small-scale, multi-directional transport modules", *Logistics Research*, Vol. 9 No. 1, p. 16, doi: 10.1007/s12159-016-0143-x.

Ma, Y., Chen, H. and Yu, Y. (2022), "An efficient heuristic for minimizing the number of moves for the retrieval of a single item in a puzzle-based storage system with multiple escorts", *European Journal of Operational Research*, Vol. 301 No. 1, pp. 51-66, doi: 10.1016/j.ejor.2021.09.032.

Mayer, S. and Furmans, K. (2010), "Deadlock prevention in a completely decentralized controlled materials flow systems", *Logistics Research*, Vol. 2 No. 3, pp. 147-158, doi: 10.1007/s12159-010-0035-4.

Mirzaei, M., De Koster, R.B.M. and Zaerpour, N. (2017), "Modelling load retrievals in puzzle-based storage systems", *International Journal of Production Research*, Vol. 55 No. 21, pp. 6423-6435, doi: 10.1080/00207543.2017.1304660.

Next Intralogistics (n.d.), "FlexConveyor", available at: https://next-intralogistics.com/catalog/product/view/id/130/s/357-01-hubumsetzer-mitparallelhub-24v-uber-eine-bahn-motorisch-einseitig/category/10/ (accessed 5 May 2025).

Raviv, T., Bukchin, Y. and de Koster, R. (2023), "Optimal retrieval in puzzle-based storage systems using automated mobile robots", *Transportation Science*, Vol. 57 No. 2, pp. 424-443, doi: 10.1287/trsc.2022.1169.

Schwab, M. (n.d.), *A Decentralized Control Strategy for High Density Material Flow Systems with Automated Guided Vehicles*, [Online], available at: https://publikationen.bibliothek.kit.edu/1000047227 (accessed 20 April 2025).

Seibold, Z., Stoll, T. and Furmans, K. (2013), "Layout-optimized sorting of goods with decentralized controlled conveying modules", *2013 IEEE International Systems Conference (SysCon)*, pp. 628-633, doi: 10.1109/SysCon.2013.6549948.

Seibold, Z., Furmans, K. and Gue, K.R. (2022), "Using logical time to ensure liveness in material handling systems with decentralized control", *IEEE Transactions on Automation Science and Engineering*, Vol. 19 No. 1, pp. 545-552, doi: 10.1109/TASE.2020.3029199.

Sgarbossa, F., Halseide, M. and Timenes, A. (2023), "Warehouse robotization with Wheel.me genius: a puzzle-based movable racks system", in *Progress in Material Handling Research*, [Online], available at: https://digitalcommons.georgiasouthern.edu/pmhr_2023/11

Shekari Ashgzari, M. and Gue, K.R. (2021), "A puzzle-based material handling system for order picking", *International Transactions in Operational Research*, Vol. 28 No. 4, pp. 1821-1846, doi: 10.1111/itor.12886.

Shirazi, E. and Zolghadr, M. (2021), "An item retrieval algorithm in flexible high-density puzzle storage systems", *Applied System Innovation*, Vol. 4 No. 2, p. 38, doi: 10.3390/asi4020038.

Siddique, P.J., Gue, K.R. and Usher, J.S. (2021), "Puzzle-based parking", *Transportation Research Part C: Emerging Technologies*, Vol. 127, 103112, doi: 10.1016/j.trc.2021.103112.

Sohrt, S. and Overmeyer, L. (2020), "Decentralized routing algorithm with physical time windows for modular conveyors", *Logistics Research*, Vol. 13, p. 8, doi: 10.31224/osf.io/f4eq6.

Sohrt, S., Seibold, Z., Krühn, T., Prössdorf, L., Overmeyer, L. and Furmans, K. (2014), "Buffering algorithms for modular, decentralized controlled material handling systems", *1st Symposium on Automated Systems and Technologies (AST), Berichte aus dem ITA Band 4/2014, S. 29-36. Garbsen: TEWISS-Technik und Wissen GmbH*.

Taha, H.A. (2006), *Operations Research: an Introduction*, 8th ed., Prentice-Hall, USA.

Taylor, G.D. and Gue, K. (2008), "The effects of empty storage locations in puzzle-based storage systems", *IIE Conference and Expo Proceedings (Vancouver, Canada - 2008)*, pp. 519-523.

Trenkle, A., Seibold, Z. and Stoll, T. (2013), "Safety requirements and safety functions for decentralized controlled autonomous systems", *2013 XXIV International Conference on Information, Communication and Automation Technologies (ICAT)*, pp. 1-6, doi: 10.1109/ICAT.2013.6684063.

Wu, G., Xu, X., (Yale) Gong, Y., Koster, R.De and Zou, B. (2019), "Optimal design and planning for compact automated parking systems", *European Journal of Operational Research*, Vol. 273 No. 3, pp. 948-967, doi: 10.1016/j.ejor.2018.09.014.

Yalcin, A., Koberstein, A. and Schocke, K.-O. (2019), "An optimal and a heuristic algorithm for the single-item retrieval problem in puzzle-based storage systems with multiple escorts", *International Journal of Production Research*, Vol. 57 No. 1, pp. 143-165, doi: 10.1080/00207543.2018.1461952.

Yu, Y. and Koster, R. (2012), "Sequencing heuristics for storing and retrieving unit loads in 3D compact automated warehousing systems", *IIE Transactions*, Vol. 44 No. 2, pp. 69-87, doi: 10.1080/0740817X.2011.575441.

Zaerpour, N., Yu, D.Y. and De Koster, R. (2010), "Optimal configuration in a puzzle-based compact storage system", *11th Trail Congress*, pp. 1-5.

Zaerpour, N., Yu, Y. and Koster, R. (2012), "Toward sustainability, high density and short response time by live-cube storage systems", in *Progress in Material Handling Research*, [Online], available at: https://digitalcommons.georgiasouthern.edu/pmhr_2012/37

Zaerpour, N., Yu, Y. and de Koster, R. (2017a), "Small is beautiful: a framework for evaluating and optimizing live-cube compact storage systems", *Transportation Science*, Vol. 51 No. 1, pp. 34-51, doi: 10.1287/trsc.2015.0586.

Zaerpour, N., Yu, Y. and de Koster, R.B.M. (2017b), "Response time analysis of a live-cube compact storage system with two storage classes", *IISE Transactions*, Vol. 49 No. 5, pp. 461-480, doi: 10.1080/24725854.2016.1273563.

Zaerpour, N., Yu, Y. and de Koster, R.B.M. (2017c), "Optimal two-class-based storage in a live-cube compact storage system", *IISE Transactions*, Vol. 49 No. 7, pp. 653-668, doi: 10.1080/24725854.2016.1273564.

Zhang, L.L., Yu, Y. and Zhang, L. (2013), "Determining optimal zone boundaries for three-class-based puzzle-based compact storage systems", *2013 IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 361-366, doi: 10.1109/IEEM.2013.6962434.

Supplementary material
The supplementary material for this article can be found online.

Corresponding author
Héctor J. Carlo can be contacted at: hector.carlo@upr.edu