Modelling Load Retrievals in Puzzle-Based Storage Systems

Masoud Mirzaei[a*], René B.M. De Koster[a], Nima Zaerpour[b]

[a] *Rotterdam School of Management, Erasmus University, Rotterdam, the Netherlands*

[b] *College of Business Administration, California State University San Marcos, San Marcos, CA 92096, USA*

Abstract

Puzzle-based storage systems are a new type of automated storage systems that allow storage of unit loads (e.g. cars, pallets, boxes) in a rack on a very small footprint with individual accessibility of all loads. They resemble the famous 15-sliding tile puzzle. Current models for such systems study retrieving loads one at a time. However, much time can be saved by considering multiple retrieval loads simultaneously. We develop an optimal method to do this for two loads and heuristics for three or more loads. Optimal retrieval paths are constructed for multiple load retrieval, which consist of moving multiple loads first to an intermediary "joining location". We find that, compared to individual retrieval, optimal dual load retrieval saves on average 17% move time, and savings from the heuristic is almost the same. For three loads, savings are 23% on average. A limitation of our method is that it is valid only for systems with a very high space utilization, i.e. only one empty location is available. Future research should investigate retrieving multiple loads for systems with multiple empty slots.

Keywords: puzzle-based storage; compact storage systems; automated warehouse; multiple load retrieval; parking lot

1. Introduction

Warehouses are important nodes in the supply chain as they allow to match supply with customer demand and to achieve economies of scale in transport. Warehouses are labour intensive and consume much space. Bartholdi and Hackman (2016) state that the fundamental idea of warehouse management lies in two resources: space and labour. While labour is usually available in urban areas, land is expensive. Space efficient storage systems offer a solution for this problem.

* Corresponding author. Email: mirzaei@rsm.nl

A conventional storage system consists of racks and aisles. Aisles are used for transporting goods to and from the storage racks. They take up space which could alternatively be used efficiently for storing loads. If space is not used efficiently, larger distances may have to be travelled to transport loads, requiring more resources. According to Tompkins et al. (1996), Roodbergen and De Koster (2001) and De Koster et al. (2007), non-value adding travel forms the majority of an order picker's time in such conventional storage systems. In the 60s, Automated Storage/ Retrieval (S/R) systems were introduced. These systems can store a large number of unit loads on a limited footprint. These systems have received much attention from researchers focusing on, e.g., travel time models and system size optimization (see e.g. Bozer and White, 1984 and Lee, 1997). In order to use the space even more efficiently, very high density storage (or puzzle-based) systems were introduced by storing the loads multi-deep (De Koster et al., 2008).

Recently, so called "puzzle-based" storage systems have been introduced. The term Puzzle-Based Storage (PBS) system comes from Gue and Kim (2007). PBS systems are very compact storage systems which are fully automated. Unit-loads are stored dense, without even a single aisle, yet each unit load can be retrieved independently. Applications of PBS systems can be found in warehouses and distribution centres (DCs), automated car parking systems, and container terminals (Zaerpour, Yu and De Koster 2015).

1.1. Description of the PBS system

The main components of a PBS system are: (1) shuttles that can move in horizontal x- and y- directions, carrying the unit loads, (2) a depot (I/O point), (3) a lift for vertical transportation in case of a multilevel system, and (4) one or more empty locations which provide sufficient manoeuvring space for shuttles to move. Such an open location is also called an "escort" because of its role of escorting the load to the destination. To bring a requested unit load to the I/O point, other shuttles have to move to first bring an escort next to the requested load. Then, the load is escorted to the I/O point by the escort.

A PBS system with one escort is comparable to the well-known 15-tile puzzle game. This game consists of 15 numbered tiles which are randomly distributed in a 4×4 square with one missing tile. The mission is to sort numbers by shuffling the tiles. In the

same fashion, N^2-1 unit loads can be stored at each level in an $N \times N$ PBS system. This results in very high space usage efficiency.

Figure 1(a) shows the top view of a typical PBS system. Cars in the picture represent loads stored in the system and the white cell represents the escort. The escort is initially located at the lower left corner, next to the I/O point. Figure 1(b) shows a PBS car parking system. Each unit load (a car) is stored on its own shuttle which can move in both horizontal directions. When an order for retrieving a load is released, the escort moves towards the requested load. This means all shuttles on the path have to move in opposite direction. Once the escort reaches a position next to the load (down or left), depending on the load's location, the shuttle which contains the load will move to the empty location. Then the escort will end up at the right or top of the requested load. It again needs to move to either down or left of the requested load to provide space for it to move. This repeats until the load arrives at the I/O point. In this way, any load can be accessed individually with no more than one empty space unit in the storage area. Although PBS systems are extremely space efficient solution, they are not fast. Therefore it is of the utmost importance to furnish this solution with faster methods. The multiple load retrieval method proposed in this paper addresses this drawback and makes PBS systems more responsive.

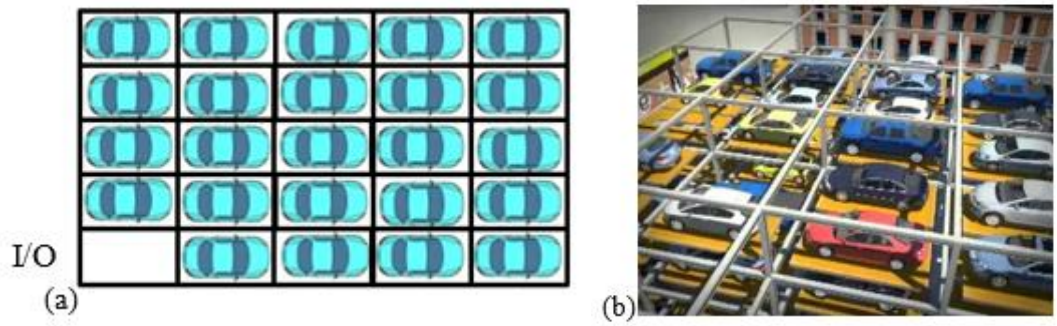

Figure 1. (a) A top view of a PBS, (b) 3D view of a PBS car parking system

1.2. Literature review

Literature on unit-load compact storage systems is not abundant. In practice, most compact storage still have at least one aisle. A crane, or S/R machine, operates in the aisle and retrieves unit loads using a satellite connected to the crane. Sari et al. (2005) study a flow rack compact storage system where the pallets are stored and retrieved at different rack sides by two S/R machines responsible for storage and retrieval respectively. De

Koster et al. (2008) and Yu and De Koster (2009, 2012) study a compact crane-based storage system with built-in multi-deep circular conveyors. The system is fully automated, and every pallet stored is accessible individually by rotating conveyors. Zaerpour et al. (2015) derive the optimal storage allocation for a crane-based compact storage system, operating in a cross-dock when all destinations of incoming loads are known.

Carousel systems constitute another category of compact automated storage retrieval systems in which accessing an item requires moving other items. These systems consist of a number of linked drawers carrying small and medium sized products that rotate in a closed loop. Litvak (2006) propose optimal picking of large orders based on the shortest rotation time and she studies the number of items collected before a turn. Hwang, Kim and Ko (1999) study standard and double carousel systems and analytically measure the effect of double shuttles on throughput.

Studies on PBS systems are new. Gue and Kim (2007) appear to be the first researchers to study PBS systems. They study a PBS system where unit loads are retrieved one by one using a single or multiple empty locations. They compare storage density and retrieval time of puzzle-based systems with traditional low density aisle-based warehouses. While traditional warehouses usually perform better than puzzle systems in terms of retrieval time, they have lower space efficiency. Kota, Taylor and Gue (2015) analytically derive the single-load retrieval time expression when multiple escorts are randomly placed within the system. They extend the expression to a system with two escorts and formulate an integer program for the general case with multiple escorts. Alfieri et al. (2012) propose heuristics for using a limited set of shared shuttles to transport unit loads in puzzle-based systems. They consider multiple I/O points, partition the storage area, and then assign shuttles to partitions based on expected work load. Shuttles move parallel where possible. Gue et al. (2013) propose a decentralized control for a deadlock-free puzzle system named GridStore. Loads arrive at one side of the system, can move individually within the system, and leave at the opposite side of the system. Each unit load communicates with neighbouring locations to decide its route. Zaerpour, Yu and De Koster (2015) study the optimal configuration of a multi-level PBS system (they call them live-cube systems) by assuming sufficient empty locations exist at each level to create virtual aisles and multiple loads can move simultaneously. When a virtual aisle has been created, determining the retrieval time is similar to a traditional, aisle-based, warehouse. The minimum number of empty locations to create a virtual aisle at a

given storage level equals the maximum of the rows and columns of the system. Yu et al. (2016) propose a method to find the optimal retrieval path for a requested load, with multiple open locations and with so-called block load movement. All these studies assume the loads all have the same size. Flake and Baum (2002) and Hearn and Demaine (2005) study the rush hour problem in a PBS car parking system, with the objective to store as many cars of different sizes in a compact storage.

While previous studies have focused on single load retrieval, in practice, information of multiple retrieval requests is usually available. Hence, multiple loads may be retrieved simultaneously, improving the performance of PBS systems significantly. In this paper, we study multiple load retrieval in a PBS system. We answer questions like how and in which sequence loads should be retrieved in order to minimize total retrieval time. This question has not yet been addressed in literature. We develop an optimal method for this problem based on joint load retrieval. The results show that by using joint retrieval, the total retrieval time can be reduced significantly compared to individual retrieval. We first present the optimal retrieval method for two loads. Then, we extend the optimal method for jointly retrieving three loads and afterwards, generalize it to retrieve multiple loads using approximate analysis. Table 1 summarizes the literature on PBS systems and highlights the contribution of this paper. The second column shows whether an optimal solution is provided or a heuristic. Column three defines the number of open locations assumed in the system. Column four shows whether there is a single move at a time or multiple loads can move simultaneously. The last column defines the number of loads that the system can retrieve together.

Table 1. Comparing the papers on PBS systems

Paper	**Optimal/ Heuristic**	**Number of escorts**	**Simultaneous load moves?**	**Single / Multiple Load retrieval**
Gue and Kim (2007)	Optimal	One, many	No	Single
Kota et al. (2015)	Optimal & Heuristic	Many (randomly placed)	No	Single
Alfieri et al. (2012)	Heuristic	Many	Yes	Single
Zaerpour et al. (2015)	Heuristic	Many	Yes	Single
Gue et al. (2014)	Heuristic	Many	Yes	Single
Yu. et al. (2016)	Optimal	One, many	Yes	Single
This paper	Optimal & Heuristic	One	No	Multiple

The remainder of the paper is organized as follow. Section 2 describes the optimal retrieval method for two arbitrary loads in the system. Section 3 extends the dual-load retrieval method to three or more loads. Section 4 compares the results with single- load retrieval. In the last section, conclusions are drawn.

2. An optimal dual-load retrieval method

In case two loads need retrieval, it is possible to reduce travel time, as compared with individual retrieval, by retrieving them jointly. We propose a dual-load retrieval method for this and demonstrate optimality by enumeration. Three methods are distinguished for retrieving two loads: (1) moving loads individually towards the I/O point using the algorithm of Gue and Kim (2007), (2) moving loads A and B by alternating between them, requiring the escort to move back and forth between the loads, and (3) bringing both loads to a given joint location and then moving them together. Obviously, the second method is not optimal, due to unnecessary extra moves of the escort traveling between the loads. We prove method (3) is optimal, for a joint position where the loads are adjacent.

We make the following assumptions for the system:

(1) All loads are stored on shuttles, which can move in both horizontal directions. This assumption is valid for particular types of PBS systems.
(2) The storage system has N rows and N columns. This can be extended to non-square systems.
(3) The I/O point is located at the lower left corner.
(4) There is only one escort, which is initially located at position (1, 1), next to the I/O point. Usually, escort will be found here after each retrieval.
(5) Only one load moves at a time, even when multiple loads need to move in the same direction.
(6) We distinguish only retrievals on a single storage level. For multiple levels, a lift fulfils the vertical transportations.

We first define several concepts to ease the exposition.

Definition 1 (Joining location for two loads in the PBS grid): A location where the two requested loads become adjacent for the first time in their retrieval path. The joining location is defined as the location of one of these two adjacent loads, namely the one which is the closest to the I/O point.

Definition 2 (Dual load move): Moving two loads consecutively on the same retrieval path with no other loads between them.

Definition 3 (Optimal joining location): A joining location for two loads, which leads to the minimum total number of retrieval moves.
Now we can formulate the following lemma.

Lemma 1: Dual-load retrieval from an optimal joining location, always performs better than or equal to two single-load retrievals, in terms of the total number of moves.
Proof: Setting the joining location at (1, 1), immediately transforms the dual-load retrieval problem into two single load retrievals. Indeed, a better joining location saves moves.

■

As Lemma 1 shows, retrieving two loads using an optimal joining location always results in a total number of moves less than or equal to the number of moves of two individual single load retrievals. Gue and Kim (2007) propose an optimal method for single load retrieval, where each load first moves by a number of so called 3-moves, followed by so-called 5-moves when the load reaches the side of the system.

Therefore, in the optimal dual-load retrieval method, the loads are first brought together at an optimal joining location, using optimal single load moves, and are then moved toward the I/O point jointly by optimal dual-load moves. Figure 2 gives a flow diagram of the dual-load retrieval method. This procedure can be explained in four steps. Algorithm 1 illustrates the steps in this method. Optimality of the joining location is ensured by enumerating all possibilities. Optimality of the adjacent location of the joining location, to which the second load will be brought, is ensured again by enumeration in step 2. Optimality of the single load moves is ensured by the method of Gue and Kim (2007). Moving the loads jointly in an optimal fashion in step 3 is explained in theorem 1. The two requested loads are (i_1, j_1) and (i_2, j_2). The load closest to the I/O point is labelled as the first load and the other load is labelled as the second load. In case of equal distances, they can be labelled randomly.

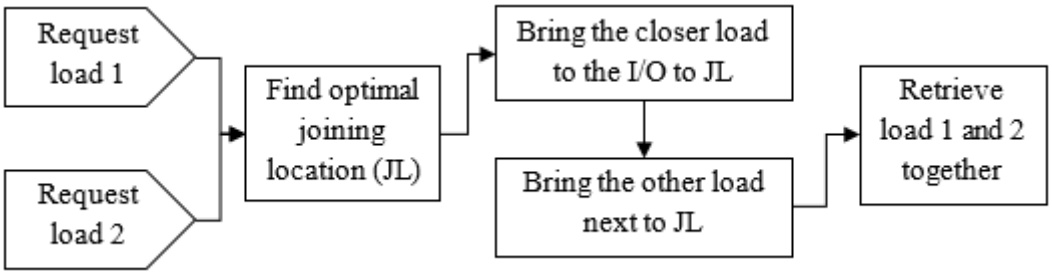

Figure 2. Flow diagram of dual-load retrieval

Algorithm 1: Optimal dual-load retrieval method

Step 1
1: **for** i=1: max $\{i_1, i_2\}$
2: **for** j=1: max $\{j_1, j_2\}$
3: let (i, j) be the joining location
4: calculate the minimum number of moves, $M_1(i, j)$, needed to bring the first load to (i, j)
Step 2
5: **for** k=1:4
6: calculate the minimum number of moves, $M_0(k)$, needed to bring the second load to one of the 4 adjacent locations of (i, j)
7: **end for**
8: pick the location k:= Argmin$\{M_0(k) | k=1,2,3,4\}$
9: $M_2(i, j):=M_0(k)$
Step 3
10: calculate the minimum number of moves, $M_3(i, j)$, needed to bring the loads jointly to the I/O point
11: $M(i, j):=M_1(i, j) + M_2(i, j) + M_3(i, j)$
12: **end for**
13: **end for**
Step 4
14: pick the solution with optimal joining location (i, j) which minimized $M(i, j)$

Algorithm 1 determines the optimum joining location, by enumerating all possible locations and comparing the results. In a system of size $N \times N$ there are N^2 possible joining locations. But, in practice, certain areas can be excluded from enumeration, depending on the position of the loads. We show in lemma 2 the number of locations that need to be enumerated is actually less than N^2. This accelerates the process of finding the joining location. Figure 3(a) shows two requested loads in the system. A joining location is marked by a 'plus' sign. The dotted lines show the boundary of locations to be enumerated.

Lemma 2: The Manhattan distance to the I/O point of an optimal joining location, is less than or equal to the Manhattan distance of the requested loads (i_1,j_1) and (i_2,j_2) to the I/O point.

Proof: a) Assume a location $L= (x, y)$ with a Manhattan distance larger than that of at least one of the loads, is nominated as the optimal joining location. This means $x + y > i_1 + j_1$ or $x + y > i_2 + j_2$. Without loss of generality we assume $x + y > i_1 + j_1$. We can obtain a new joining location *L'* by projecting *L* on a location with $x' + y' = i_1 + j_1$. *L'* outperforms *L* in the required number of moves to retrieve the loads, for two reasons. 1) *L'* is closer to the I/O point. 2) Even when *L'* is farther from one of the loads than *L*, the total number of moves will still reduce, as the other load does not have to travel to *L* first, and then reverse towards the I/O point, which would generate extra moves. Instead, the farther load can move towards *L'* and consequently the I/O point, which decreases the number of steps. As all this leads to a contradiction, *L* cannot be located farther than any of the two loads.

■

In a number of situations, the joining location can be predefined and no enumeration is needed. Table 2 provides a list of such conditions. In condition 1, when both requested loads are located at the left side of the system (i.e. i_1, i_2=1), the escort goes directly to the farther load and brings the closer load to (1, j_1-1) on its path. This is the joining location for the loads. The same procedure can be applied for condition 2. In condition 3, where both loads are on the diagonal, again the optimal path for the escort is to directly go to the farther load and bring it next to the closer one. This involves moving the closer load to either (i_1, j_1-1) or (i_1-1, j_1). In the worst case scenario of condition 4, one load is located anywhere at the left side of the system and the other one is anywhere at the bottom side. Then the joining location is (1,1) which basically means two individual retrievals.

Table 2. Conditions that lead to a predefined optimal joining location

Nr.	Condition	Predefined optimal joining location
1	i_1, i_2=1 and j_1< j_2	(1, j_1-1)
2	j_1, j_2=1 and i_1< i_2	(i_1-1,1)
3	i_1= j_1 and i_2= j_2	(i_1, j_1-1) and (i_1-1, j_1)
4	(1,j_1) and (i_2,1)	(1,1)

In the first step of algorithm 1, we need to know the number of steps to bring the first load to the joining location via the shortest path for each possible joining location. This can be done by the single-load retrieval method of Gue and Kim (2007); the only difference is the I/O point as the destination has been replaced by the joining location. Figure 3(b) shows this transfer.

The second step brings the second load next to the first one. It selects the best locations adjacent to the joining location such that the total number of moves is minimized. It is determined by enumeration and comparing the results for each adjacent location. Figure 3(c) shows how the second load joins the first one. The enumerated locations are marked by '×'. At the end of this phase, as shown in Figure 3(d), the loads are adjacent, in a horizontal position. However, depending on the position of the loads, vertical optimal joining configurations are possible.

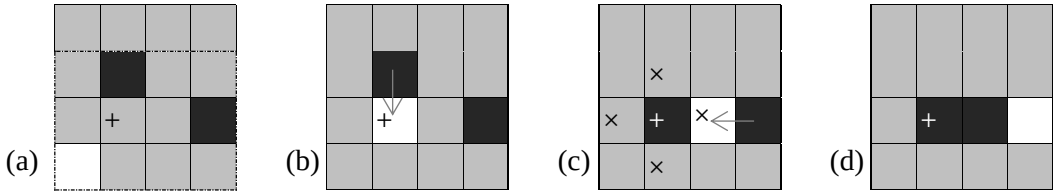

Figure 3. Joining procedure of two loads in dual-load retrieval: (a) joining location is selected, (b) first load moves there, (c) second load moves to its adjacent, (d) loads are ready to be retrieved together.

The third step calculates the number of moves needed to bring the loads jointly to the I/O point. In lemma 3 we prove the optimal way to move two adjacent loads is via so called "dual-load" moves. Regardless of a horizontal or vertical position of the loads at the joining location, two types of dual-load moves are available: 5-moves and 7-moves. In the following we explain them in detail. The smallest series of steps that is needed to perform a joint move is via 5-moves. As shown in the Figure 4(a), the escort takes three steps to reach the proper position that makes space for the loads to move closer to the I/O point. Then it takes two more steps to move both loads ahead. A series of 5-moves are performed until no more move of this type is possible, i.e. the loads reach one side of the grid. After that, 7-moves are performed as shown in Figure 4(b). Here, the escort takes 5 steps to reach the proper position, and then two more steps are required to move the loads. This is repeated until the loads are retrieved.

In the fourth step of the algorithm, the solution is picked with the total minimum number of moves.

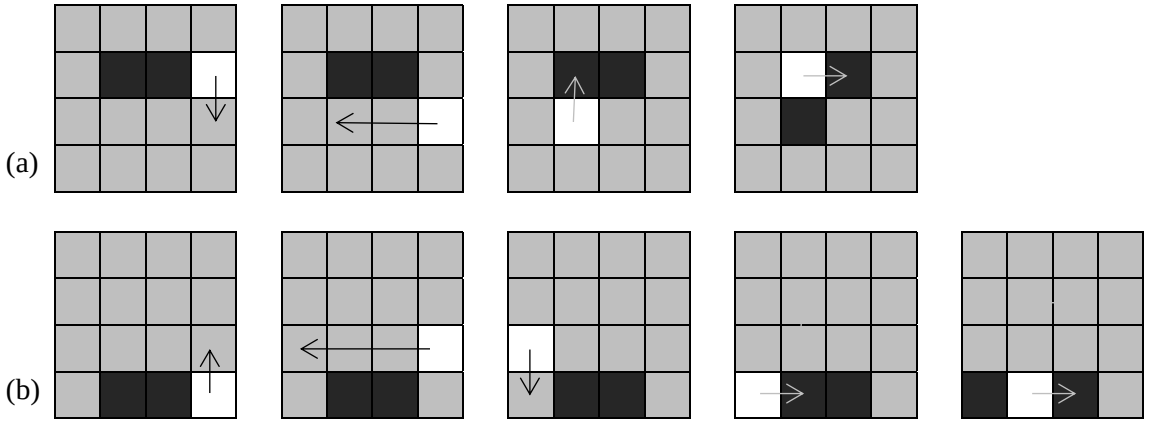

Figure 4. Demonstration of moves in dual-load retrieval: (a) 5-step, (b) 7-step

Lemma 3: To move two adjacent loads to the I/O point, it is optimal to have no load between them during the retrieval steps.

Proof: We prove this lemma by contradiction. Suppose that one or more items are located between the two loads, we then show the number of steps can be reduced if there is no intermediary item. Assume there is one item between the loads, this means their rectilinear distance is two. As demonstrated in Figure 5(a), at least nine steps are required to move both loads one space unit. By simply eliminating the in-between item, as shown in Figure 5(b), the number of steps reduces to seven. This single item between loads, results in two extra escort steps merely to bypass this item. The same argument holds for the case where the loads are vertically aligned. In a similar fashion, having k >1 items between the loads will result in 2k extra escort steps to reposition the loads one space unit. This means having no item between the loads leads to the minimum number of steps.

■

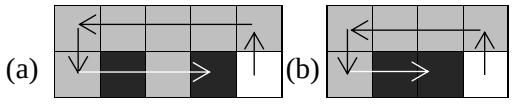

Figure 5. Moving loads with different amount of interspace: (a) one interspace, (b) no interspace

Theorem 1 formulates the number of moves needed in an optimal method of retrieving two adjacent loads. Before that we make the following observations.

Observation 1: In order to determine the number of steps required to retrieve loads from given positions, it is sufficient to track the number of moves made by the escort. This is because each load move corresponds to an escort move.

Observation 2: Due to symmetry of the system, the number of moves required to retrieve a load at location (*i*, *j*) equals the number of moves for a load at location (*j*, *i*).

Theorem 1: The minimum number of moves to retrieve two horizontally aligned adjacent loads in a puzzle system is:

$$\begin{array}{ll} 7i + 3j - 9 & i > j \\ 10i - 9 & j = i \\ 7j + 3i - 13 & i < j \end{array}$$

where (*i*, *j*) is the location of the load closer to the I/O point and the escort is behind them.
Proof: According to Lemma 3, two adjacent loads should travel by dual-load moves. In this strategy we keep track of the route of the first load and the second load follows it. The first load can move either leftward or downward, using 5-moves and 7-moves. Figure 6 shows an example of a typical route in this approach. An observed property of the dual-load moves is that the route changes direction every time after two 5-moves. In case $j > i$ there are i pairs of 5-moves necessary for a total of $5(2i) = 10i$ moves, after which the position of the first load should be (1, *j*-*i*-2). Next, *j*-*i*-2 7-moves are required to retrieve the first load for 7(*j*-*i*-2) moves. At the end, an additional one move is performed to retrieve the second load, thanks to extra empty space obtained by retrieving the first load. Therefore, in total $10i + 7(j-i-2) + 1 = 7j+3i-13$ escort moves are necessary. In case $j= i$, and i is an odd number, the first load can reach the I/O point with i-1 pairs of 5-moves. An additional move is necessary to retrieve the second load for a total of $5*2(i - 1) + 1= 10i - 9$ moves. If i is an even number, $2i$-3 5-moves is needed, and then an extra 7-move and an additional single move are needed to retrieve both loads. In total $5(2i - 3) + 7 + 3= 10i - 9$ moves are needed which shows the results are the same for both even and odd i. the case for $j < i$ follows in a similar fashion.

■

As a corollary to this theorem, according to the symmetry property stated in Observation 2, the same approach can be used for the case of vertical alignment of the loads. The formulation is as follows:

$$\begin{array}{ll} 7j+3i-9 & j>i \\ 7i+3j-13 & j<i \\ 10i-9 & j=i \end{array}$$

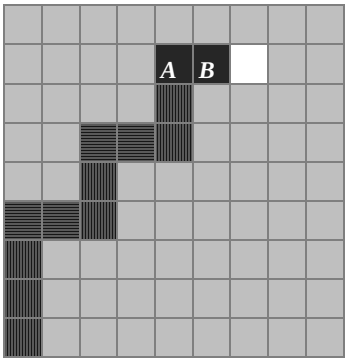

Figure 6. A typical dual retrieval route for two loads *A* and *B*

In the dual-load retrieval method of algorithm 1, the optimal joining location is not unique. For instance when the adjacent loads are located at the diagonal, two optimal joining locations exist: one space unit to the left and one space unit to the south.

Algorithm 1 helps to find the optimal solution. However, since the number of joining locations that are evaluated is $O(N^2)$, and the number of steps to jointly retrieve the loads is $O(N)$, the algorithm is $O(N^3)$, where N is the size of the system. Therefore, in addition to this optimal method, we propose a heuristic, that yields a near optimal solution in a considerably shorter time. This heuristic, can be easily adapted to retrieve more than 2 loads as will be explained in section 3. Algorithm 2 shows the steps required for two loads (i_1, j_1) and (i_2, j_2). The location of the load closest to the I/O point is denoted by (i_c, j_c) and the location of the farther load is denoted by (i_f, j_f). The subroutine introduced in this algorithm finds J and L as the joining location and the location of the adjacent load respectively, based on location of the loads as the input.

Algorithm 2: Heuristic method for two loads

1: let $(i, j) = (\min\{i_1, i_2\}, \min\{j_1, j_2\})$ be the joining location
2: **Subroutine JL (J, L, (i_1, j_1), (i_2, j_2))**
3: **if** $(i, j) \neq (i_c, j_c)$ **then**
4: $J := (i, j)$
5: $L := (i + \frac{i_f - i}{(i_f - i)+(j_f - j)}, j + \frac{j_f - j}{(i_f - i)+(j_f - j)})$
6: **else** $L := (i_c, j_c)$
7: **if** $i > j$ **then** $J := (i-1, j)$
8: **else** $J := (i, j-1)$
9: **end if**
10: **end if**
11: **End subroutine JL**
12: move the load located at (i_c, j_c) to J

```
13: bring the other load to L
14: move the loads jointly to the I/O point
```

To compare the performance of the heuristic with the optimal method, numerical results for 20 random instances of dual-load requests are presented in Table 3. To generate random requests, for any given *N*, two unique locations with random coordinates (*i* , *j*) are picked. Then, the loads stored at these locations are retrieved using both optimal and heuristic methods to compare the results. This is repeated 20 times for each system size. The optimal and approximate number of moves for dual-load retrieval are presented in column Avg. Opt. and Avg. Aprx., for systems of different sizes, averaged over 20 instances. The average gap between the number of optimal and heuristic methods is presented in column Avg. Gap, together with the minimum and maximum gap. The last column shows the average computation time for the optimal method. The computation time for the heuristics is negligible. The heuristic method appears to perform near optimal. In fact it appears the heuristics performs optimal in more than half of the instances.

Table 3. Comparison between optimal dual-load retrieval and the heuristic method

Size of the system(N)	Avrg. Opt.	Avrg. Aprx.	Avrg. Gap	Avrg. Compt. time (sec)
5	21.4	22.10	0.7, (0, 2)	0.18
7	38	38.4	0.4, (0, 2)	0.61
10	63.4	63.8	0.4, (0, 2)	1.64
20	133.8	135.9	2.1, (0, 10)	18.62
50	346.2	358.6	12.4, (0, 46)	273.10

3. Multiple load retrieval method

In this section we first consider retrieving three loads in the system, and then generalize it to more than three loads. In case of three requested loads, each load can be retrieved individually, or together with one or two other loads. Lemma 4 proves that joining loads is required to obtain the minimum number of moves, similar to lemma 1.

Lemma 4: Retrieving three loads jointly from an optimal 3-load joining location, performs better or equal to retrieving one or all of them individually.

Proof: Setting the joining location at (1, 1), immediately transforms the joint retrieval of three loads into three single-load retrievals, or a single-load retrieval and a dual-load retrieval. A better choice of joining locations saves moves.

■

To join and retrieve three loads, different combinations and sequences for the loads A, B and C exist. For example (AB, ABC) means the loads A and B join first at an intermediate joining location, and then they join C at a final joining location. Similarly, the other alternatives are (AC, ACB) and (BC, BCA). These are the main alternatives for joining the loads, but there are other alternatives that are sub-cases of these main alternatives. For instance, individual retrievals is a case when the joining location is set at (1, 1), or joining all loads at one location is a case when the intermediate and the final joining locations are at the same point. Therefore, we only evaluate the three main combinations in the algorithm 3.

One way to obtain the optimal solution to the problem is enumerating all move sequences to all possible joining location. Therefore the number of moves would be $O(m!\,N^m)$where m is the number of loads to be retrieved. As this number grows very rapidly with m and N we propose a heuristic method for three loads or more. Suppose that the third load (i_3, j_3) is requested in addition to the other two loads. Algorithm 3 shows the steps required to retrieve them together. (i_a, j_a) and (i_b, j_b) are the locations of the first two loads in the combination k. (i_l, j_l) is the location of the last load in the combination k.

Algorithm 3: Heuristic method for three loads

1: let $(r, q)=(\min\{i_1, i_2, i_3\}, \min\{j_1, j_2, j_3\})$ be the joining location
2: **for** k= 1:3
3: **Subroutine JL (J, L, (i_a, j_a), (i_b, j_b))**
4: calculate the number of moves needed to bring the first two loads in the combination to J and L for $M_1(k)$ moves
5: calculate the number of moves needed to bring these two loads jointly to the (r, q) for $M_2(k)$
6: $L_2 \coloneqq (r + \frac{i_l - r}{(i_l - r)+(j_l - q)}\,,\; j + \frac{j_l - q}{(i_l - r)+(j_l - q)})$
7: calculate the number of moves needed to bring the third load (i_l, j_l), to L_2 for $M_3(k)$
8: $M(k) := M_1(k) + M_2(k) + M_3(k)$
9: **end for**
10: pick the combination k:=Argmin$\{M(k) \mid k=1,2,3\}$
11: move the loads jointly from (r,q) to the I/O point

In the heuristic method for three loads, usually an intermediate joining location is established for combining two loads, in order to minimize the individual moves and maximize dual-load moves. Theorem 2 shows the number of moves required to retrieve the loads after they became adjacent. For more loads, the algorithm can be extended using the same approach.

Theorem 2: The minimum number of moves to retrieve three adjacent loads in a puzzle system is:

$9i + 5j - 11$	$i > j$
$14i - 13$	$i = j, j = 3,6,9, \ldots$
$14i - 6$	$i = j - 1, j \neq 4,7,10, \ldots$
$14i - 1$	$i = j - 2, j = 4,7,10, \ldots$
$14i + 8$	$i = j - 3, j = 4,7,10, \ldots$
$9j + 5i - 17$	O.W.

where (i, j) is the location of the load closest to the I/O point and loads are horizontally aligned, having the escort behind them.

Proof: The proof is similar to theorem 1. Again, the moves of the first load are tracked, and the other two loads follow it. The only difference is that the loads move using 7-moves and 9-moves due to an extra load. ■

According to the symmetric property stated in Observation 2, the same approach applies for the case of the vertically aligned loads. The formulation is as follows:

$9j + 5i - 11$	$j > i$
$14j - 13$	$j = i, i = 3,6,9, \ldots$
$14j - 6$	$j = i - 1, i \neq 4,7,10, \ldots$
$14j - 1$	$j = i - 2, i = 4,7,10, \ldots$
$14j + 8$	$j = i - 3, i = 4,7,10, \ldots$
$9i + 5j - 17$	O.W.

4. Numerical results

To evaluate the performance of the multiple load retrieval method presented in this paper, we here compare the total number of moves required to retrieve the loads with single load retrieval method. All calculations are done in MATLAB.

For any given system size, two unique locations are randomly generated. These random locations represent requested loads. For the case of three-load retrieval, three unique locations are randomly generated. The numbers of steps required for retrieval of these loads by different methods are calculated. This is repeated for 100 instances.

Table 4 compares the dual-load retrieval method to individual retrieval and shows savings the dual-load retrieval method can obtain. AvgSL and AvgDL are the averages of the total number of moves in single-load and dual-load retrieval, respectively. The

maximum savings are obtained when the loads are positioned at (1, *N*) and (1, *N*-1). The average saving is calculated as (AvgSL- AvgDL)/ AvgSL × 100%.

Table 4. Savings for the optimal dual-load retrieval compared to individual retrieval

N	AvgSL	AvgDL	Maximum savings (%)	Average savings (%)
5	31.3	24.5	41	20
7	45.6	36.4	38	20
10	82.5	65.8	35	19
15	129.2	105.9	35	19
20	157.4	129.5	34	18
50	437.1	367.3	33	17
100	911.5	755.6	33	17

According to Table 4, for large values of *N*, the maximum savings are about 33% and the average savings are about 17% of the number of moves needed for individual retrieval. Note that the savings for small systems are higher than for large systems. This is caused by the effect of the second empty spot that appears next to the I/O point, after the first load has been retrieved, and which makes the distance of the second load to the I/O point one unit shorter. This effect disappears for systems larger than 10.

Figure 7 illustrates savings in number of moves that are achieved for two loads, when one location is given and the other is arbitrary. In Figure 7(a), one load is located at (1, 10) and the other one is at location (*i*, *j*). The vertical axis represents the percentage of savings for each location. In Figure 7(b) the first load is located at (5, 5).

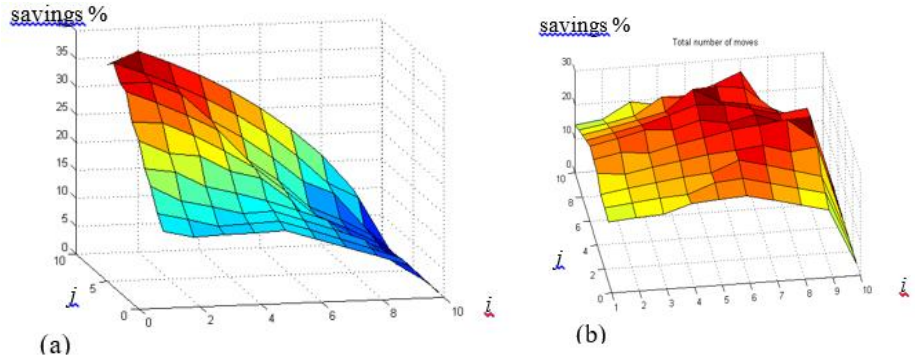

Figure 7. Savings achieved for a given first load located at (a) (1,10), and (b) (5,5), and the second load is at (*i*, *j*).

Based on the experiment and Figure 7, we can make the following observation.

Observation 3: when one load is located at (1,*j*) for *j*=1..*N* and the other load at (*i*,1) for *i*=1..*N*, savings in dual-load retrieval are not significant (see Figure 7(a)). Apparently, too many single-load moves are required before the loads become adjacent. On the other

hand, when the loads are located at (1,*j*) and (1,*j*-1) for *j*=1..*N* or (*i*,1) and (*i*-1,1) for *i*=1..*N*, savings are substantial. This is due to the fact that they are already adjacent, and they are located at the very end of the grid.

Table 5 shows the saving that can be obtained by the three-load retrieval method, as compared to individual retrieval. AvgTL is the averages of the total number of moves in three-load retrieval. The maximum saving is calculated when the loads are located at (1, *N*) and (1, *N*-1). The average savings are shown in the last column.

Table 5. Savings for three-load retrieval heuristics

N	**AvgSL**	**AvgTL**	**Max savings (%)**	**Average savings (%)**
5	41.6	29.1	59	30
7	75.3	54.2	52	28
10	113.4	83.5	49	26
15	179.5	134.6	47	25
20	254.8	193.9	46	24
50	886.9	682.3	45	23
100	1683.1	1296.4	45	23

Savings achieved by three-load retrieval shows significantly better performance than single-load. Based on the results, the performance is on average 6% higher than the dual-load retrieval. The values converge as the size of the system grows.

4.1. A case study

In the numerical study, we considered random load requests. However, in a real system, where there is a pool of retrieval requests, there is an opportunity to group the loads that are close together and gain higher savings. To demonstrate such benefits and to illustrate the application in practice, we apply the multiple load retrieval method to a medium-sized car parking system. We consider a 10×10 single-level puzzle-based car parking system (total capacity of 99 cars)where the size of each shuttle is 2.5×4.8 m (see multi story car parks, 2016). The shuttle speed is 52 m/min in *x*-direction and is 100 m/min in *y*-direction. Given these specifications of the system, each move in both directions takes 2.88 seconds. Therefore, although the system's shape is rectangular (25×48 m), it is square in terms of travel time. The system follows the dual-load retrieval method when possible. We assume always 3 people are waiting to retrieve their car with a flexible first-come first-served policy, which means for every retrieval, we take the first car request and pair it with the closer one of the other two car requests. The parking lot operates 24/7 and we perform a Monte Carlo simulation for 1000 random car requests.

Table 6 shows the time it takes to retrieve cars individually and in pairs. The total retrieval time for 1000 cars using dual-load retrieval method is 23.59 hours. On the other hand, the total retrieval time using individual retrieval method is 30.80 hours. Thus, we can obtain 23 % improvement in total retrieval time using the dual load retrieval method. This saving is even higher than the average saving for the random case in Table 4, as we first examine the location of the three retrieval request and then pair the closer cars.

Table 6. Savings for a real car parking system

Number of Bays	**Total time individual retrieval (h)**	**Total time dual-load retrieval (h)**	**Average saving (%)**
99	30.80	23.59	23

5. Conclusions

Puzzle-based storage systems are fully automated, unit-load, high-density storage systems that pair a small footprint with high efficiency in retrieval. In this paper, we first proposed an optimal dual-load retrieval method that, compared to single-load retrieval, saves on average 17 % in retrieval time by bringing the loads first together to an optimal joining location. In addition, a heuristic method is proposed for retrieving two loads that finds a near-optimum solution much faster. This heuristic is then extended to retrieve three and more loads. For retrieving three loads, the results show that on average a 23% saving can be achieved compared to single-load retrieval. Puzzle-based storage systems are still quite rare in practice. However, as the technology becomes less expensive, space becomes more scarce, and as we move into a 24/7 economy, these systems provide a great opportunity to provide high fulfilment performance. Our algorithms and insights can help to realize such high performance, by properly grouping requests and retrieving them jointly.

The paper makes some assumptions which may be relaxed in future research. First, we study retrieval of loads on a single level. In multi-level systems, our results will apply per level. Second, we assume a single escort. Extension of exact results to systems with multiple escorts is not straightforward, but heuristics results may be possible. Third, the proposed algorithms can be embedded in a simulator to obtain the cycle time savings. Last, we assume loads move only one step at a time and only one load can move at a time. This assumption is valid, depending on the type of mechanical retrieval system, but it may be possible to extend results to systems with simultaneous load movements.

References

Alfieri, A., M. Cantamessa, A. Monchiero, and F. Montagna. 2012. “Heuristics for puzzle-based storage systems driven by a limited set of automated guided vehicles.” Journal of Intelligent Manufacturing 23 (5): 1695-1705.

Bartholdi, J. J., and S. T. Hackman. 2016. “Warehouse & Distribution Science.” v. 0.97, Georgia Institute of Technology, available online at: www.warehouse-science.com (accessed on: 10 Nov. 2016).

Bozer, Y. A., and J. A. White. 1984. “Travel-time models for automated storage / retrieval systems.” IIE Transactions 16: 329-338.

De Koster, M.B.M., T. Le-Duc, and Y. Yu. 2008. “Optimal storage rack design for a 3-dimensional compact AS/RS.” International Journal of Production Research 46(6): 1495–1514.

De Koster, R., T. Le-Duc, and K. J. Roodbergen. 2007. “Design and control of warehouse order picking: A literature review.” European Journal of Operational Research 182(2): 481-501

Flake, G.W., and E. B. Baum. 2002. “Rush Hour is PSPACE-complete, or why you should generously tip parking lot attendants.” Theoretical Computer Science 270 (1–2): 895–911.

Gue, K.R., K. Furmans, Z. Seibold, and O.Uludag. 2014. “Gridstore: a puzzle-based storage system with decentralized control.” Automation Science and Engineering, IEEE Transactions on Automation Science and Engineering 11(2): 429 – 438.

Gue, K. R., and B. S. Kim. 2007. “Puzzle-based storage systems.” Naval Research Logistics 54 (5): 556–567.

Kota, R. V., D. Taylor, and K. R. Gue. 2015. "Retrieval time performance in puzzle-based storage systems." Journal of Manufacturing Technology Management 26 (4): 582 – 602.

Lee, H. F. 1997. “Performance analysis for automated storage and retrieval systems.” IIE Transactions 29: 15-28.

Litvak, N. 2006. “Optimal picking of large orders in carousel systems.” Operations Research Letters 34 (2): 219 – 227

Hearn, R. A., and E. D. Demaine. 2005. “PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation.” Theoretical Computer Science 343 (1-2): 72 – 96.

Hwanga, H., C. Kim, and K. Ko. 1999. “Performance analysis of carousel systems with double shuttle.” Computers & Industrial Engineering 36 (2): 473-485.

Roodbergen, K. J., and R. De Koster. 2001. “Routing methods for warehouses with multiple cross aisles.” International Journal of Production Research 39(9): 1865-1883.

Tompkins, J. A., J. A. White, Y. A. Bozer, E. H. Frazelle, J. M. A.Tanchoco , and J. Trevino. 1996. “Facilities Planning.” New York: John Wiley & Sons, Inc., 2nd edition.

Sari, Z., C. Saygin, N. Ghouali. 2005. “Travel-time models for flow-rack automated storage and retrieval systems.” International Journal of Advanced Manufacturing Technology 25: 979-987.

Yang, J. H., and K. H. Kim. 2006. “A grouped storage method for minimizing relocations in block stacking systems.” Journal of Intelligent Manufacturing 17: 453–463.

Yu, Y., H. Yu, R. De Koster, and N. Zaerpour. 2016. “Optimal algorithm for minimizing retrieval time in puzzle based storage system with multiple simultaneously movable empty cells.” Working paper Erasmus University Rotterdam.

Yu, Y., R. De Koster. 2012. “Sequencing heuristics for storing and retrieving unit loads in 3D compact automated warehousing systems.” IIE Transactions 44(2): 69-87.

Yu, Y., R. De Koster. 2009. “Designing an optimal turnover-based storage rack for a 3D compact automated storage and retrieval system.” International Journal of Production Research 47(6): 1551-1571.

Zaerpour, N., Y. Yu, and R. de Koster. 2015. “Small is beautiful: A framework for evaluating and optimizing live-cube compact storage systems.” Transportation Science. DOI 10.1287/trsc.2015.0586

Zaerpour, N., Y. Yu, and R. de Koster. 2015. “Storing fresh produce for fast retrieval in an automated compact cross-dock system.” Production and Operations Management 24(8): 1266-1284.