Modelling Load Retrievals in Puzzle-based Storage Systems

Masoud Mirzaei[a], Nima Zaerpour[b], René de Koster[a]

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

[b]*Faculty of Economics and Business Administration, VU University Amsterdam*

1. Introduction

Warehouses are important nodes in the supply chain as they allow to better match supply with customer demand and to achieve economies of scale. Warehouses require labour and land. While labour is usually available in urban areas, land is expensive. As a solution, space efficient storage systems are being developed.

A conventional warehouse consists of racks and aisles. Aisles are used for transportation, moving to and from stored goods. They take up land and space which could alternatively be used efficiently for storing loads; the main purpose of a warehouse. As a solution, puzzle-based storage systems have recently been introduced which do not require aisles. Puzzle-based storage (PBS) systems are very compact storage systems which are fully automated. Unit-loads are stored dense, without aisles, 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 et al., 2014).

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 bring an escort next to the requested load. Thereby the overall retrieval process consists of series of shuttle moves that bring the escort to an effective position for the requested load.

In general, a PBS system with one escort is comparable to the well-known 15-tile puzzle game. The game consists of 15 numbered tiles which are randomly distributed in a larger 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 in an $N \times N$ PBS system. This results in very high space efficiency.

Figure 1(a) demonstrates the top view of a typical PBS system. The escort is initially located at the lower left corner, close to the I/O point. Figure 1(b) shows a PBS car parking system.

Figure 1- a) top view of a PBS system, b) A PBS car parking system

This paper aims to find the optimal retrieval methods for PBS systems. Gue and Kim (2007) appear to be the first researchers to study such 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 perform better than puzzle systems in terms of retrieval time, they have lower space efficiency. Rohit et al. (2010) analytically derive the single-load retrieval time expression when a single escort is 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 AGVs to transport unit loads in puzzle-based systems. They consider multiple I/O points, partition the storage area, and then assign AGVs to partitions based on expected work load. AGV moves are parallelized 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 et al (2014) study the optimal configuration of a multi-level PBS system (so-called live-cube systems) by assuming sufficient empty locations exist at each level to create virtual aisles. 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 equals the maximum of the rows and columns of the system.

Carousel systems shape another category of 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 rotating time and study number of steps before a turn. Hwang et al. (1999) study standard and double carousel systems and analytically measure the effect of double shuttle on throughput. Some papers study the rush hour problem, which deals with loads of different sizes in a PBS system, such as Flake and Baum (2002) and Hearn and Demaine (2005). Application Roush hour problem can be found in car parking.

While previous studies have focused on single load retrieval, in practice, information of multiple retrieval requests is usually available. Hence, multiple loads can 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 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 retrieval time model for two loads. Then, we extend this model for jointly retrieving three loads and generalize it to retrieving multiple loads using approximate analysis. *Table 1* summarizes the literature on PBS and highlights the contribution of this paper.

Table 1- comparing researches on PBS

paper	Optimal/ Heuristic	Number of escorts	Simultaneous load moves	Single / Multiple Load retrieval
Gue and Kim (2007)	Optimal	One, many	No	Single
Rohit et al. (2010)	Optimal	One, two (randomly placed)	No	Single
Alfieri et al. (2012)	Heuristic	Many	Yes	Single
Zaerpour et al. (2014)	Heuristic	Many	Yes	Single
Gue et al. (2013)	Heuristic	Many	Yes	Single
This paper	Optimal / Heuristic	One	No	Multiple

Section 2 describes the optimal retrieval model for two arbitrary loads in the system. Section 3 compares our model with individual retrieval.

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 two loads to a given distance of each other and then move 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, with an optimal rectilinear distance of one, i.e. where the loads are adjacent.

Our model is based on the following assumptions:

- The storage system has N rows and N columns.
- The I/O point is located at the lower left corner.
- There is only one escort, which is initially at located at position [1,1], which is near the I/O point
- One load moves at a time.

Definition1 (Joining location): for two loads in the PBS grid, the locations where the two requested loads become adjacent for the first time in their retrieval path, identify the joining location. The closest location of these two adjacent loads to the I/O point is defined as the joining location.

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

Lemma1: Dual-load retrieval always performs better than or equal to two single-load retrievals in terms of total number of moves.

Proof: setting the joining location at (1,1), immediately transforms the dual-load retrieval into two single load retrievals. 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 and then 5-moves when reaches a side of the system. Therefore, in order to have optimal method for retrieving two requested loads, we need to use a joining location, in such a way that every step of the method is optimal. The dual-load retrieval method satisfies these needs.

In the optimal dual-load retrieval method, the loads are brought together at an optimal joining location, and then moved toward the I/O point jointly by optimal dual-load moves. This procedure can be explained in four steps. Figure 2 illustrates the algorithm for this method. Optimality of steps 1 and 3 are ensured by enumerating all possibilities. Optimality of step 2 is ensured by the method of Gue and Kim (2007). Optimality of step 4 is shown in lemma 3 and theorem 1. Here, of two requested loads, the closer load 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.

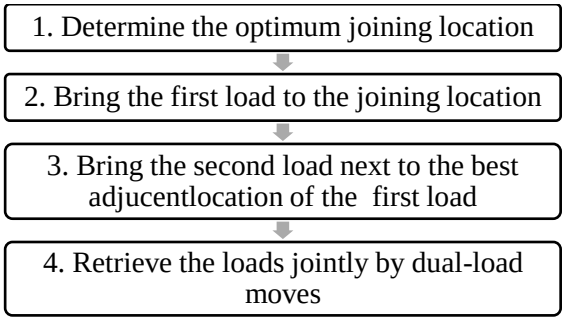


Figure 2- algorithm 1: optimal dual-load retrieval method

In the first step of the algorithm 1, the optimum joining location is determined such that the total number of moves in dual-load retrieval cannot be reduced by choosing a different joining location. To find such a location we enumerate all possible locations and compare the results. In a system of size $N \times N$ there are N^2 possible joining locations. We show in lemma 2 the number of locations that needs to be enumerated is actually less than N^2. Figure 3(a) shows two requested loads in the system. A joining location is marked by a cross sign. The dotted lines show the enumeration boundary.

Lemma 2: The Manhattan distance of an optimal joining location to the I/O point, is less than the maximum Manhattan distance of the requested loads to the I/O point.

Proof: if a location $X = (x, y)$ outside of this boundary is nominated, it can be outperformed by a joining location Y with the same x or y, obtained by projecting X on the boundary. Retrieving the two loads via joining location Y takes fewer moves than via joining location X, because: 1) Y is closer to the I/O point, and 2) it is closer to the loads.

After determining the joining location, the next step of algorithm is to bring the first load to the joining location via the shortest path. 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 third step brings the second load next to the first one. It selects the best locations adjacent to the joining location such that 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.

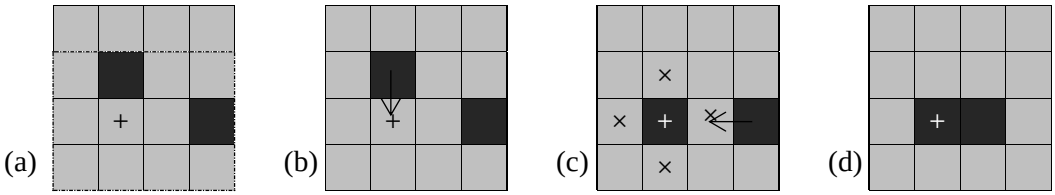


Figure 3: Joining procedure of two loads in dual-load retrieval: (a) joining location is selected, (b) first load moves there, (c) second load moves to its adjacent, (d) loads are ready to be retrieved together. Note: position of the escort is not shown.

The final phase is to move the loads jointly to the I/O point. In lemma 3 we prove the optimal way to move two adjacent loads is via dual-load moves. Regardless of 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 details. 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 get closer to the I/O point. Then it takes two more steps to move the 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 to move the loads. This is repeated until the loads are retrieved.

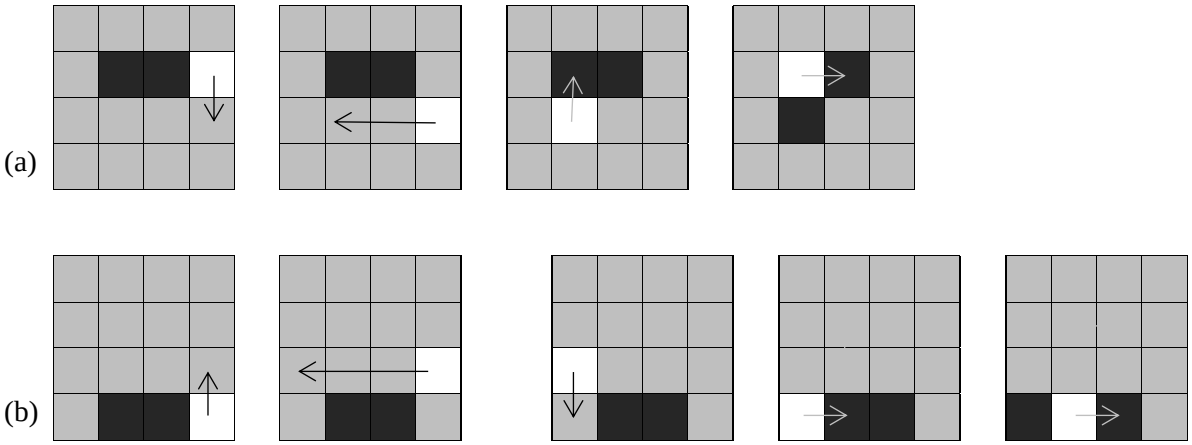


Figure 4: demonstration of moves in dual-load retrieval: (a) 5-step, (b) 7-step

Lemma 3: To move two adjacent loads to the I/O point, it is optimal to use dual-load moves and have no load between them.

Proof: We prove this lemma by contradiction. Suppose that there is one or more items between loads, we show number of steps can be reduced by eliminating them. In case of one item between loads, which means their rectilinear distance is two, as demonstrated in Figure 5(a), at least nine steps are required to move the loads one space unit. By simply eliminating the in-between item, as shown in Figure 5(b), number of steps reduces to seven. In fact, the existence of one item between loads, results

in two redundant escort steps merely to pass this item. The same argument holds for any other loads positions. Similarly, having k than one item between loads will result in $2k$ more escort steps in each repositioning of the loads. This essentially means having no item between loads is optimal.

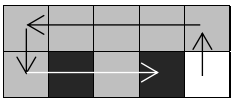
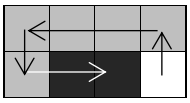
(a)  (b) 

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

Theorem 1 formulates number of moves 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.

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 adjacent loads in a puzzle system is:

$$\begin{array}{ll} 10i-5 & j=i \\ 7j+3i-9 & j>i \\ 7i+3j-5 & j<i \end{array}$$

where (i,j) is the location of the load closer to the I/O point and loads are located between the escort and the I/O point.

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 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 $5(2i) = 10i$ after which, the position of the first load should be (1, j-i-2). So j-i-2 number of 7-moves is required to retrieve the first load for 7(j-i-2). At the end an additional 5-step individual move is performed to retrieve the second load. In total $10i + 7(j\text{-}i\text{-}2) + 5 = 7j+3i\text{-}9$ escort moves are necessary. In case $j=i$, and i is odd number, it can reach I/O point with i-1 pairs of 5-moves. An additional 5-move is necessary to retrieve the second load for a total of $5*2(i\text{-}1)+5=10i\text{-}5$. If i is an even number, $2i$-3 5-moves is possible, and then extra 7-moves and 3-moves are needed to retrieve both loads. In total $5(2i\text{-}3)+7+3=10i\text{-}5$ which shows the same results for both even and odd i. Similarly we can formulate for $j<i$. Due to symmetric property stated in observation 2, the same approach is utilizable for the case of vertical alignment of the loads. The formulation is as follows:

$$\begin{array}{ll} 7j+3i-5 & j>i \\ 7i+3j-9 & j<i \\ 10i-5 & j=i \end{array}$$

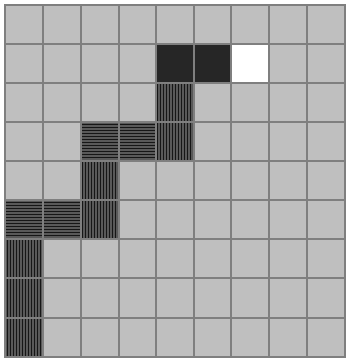


Figure 6: a typical dual retrieval route

In the dual-load retrieval method of algorithm 1, sometimes the optimal joining location is not unique. For instance when both loads are located at diagonal, see Figure 7, there are two joining locations. In this cases the total number of retrieval moves are the same for any of these optimal joining locations.

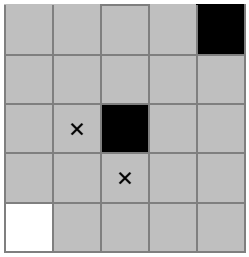


Figure 7- optimal joining location for two loads located at diagonal

3. Multiple-load retrieval method

In this section we first consider 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 combined with other loads. Thus, loads can be combined in for different ways. For example different combinations of the loads A, B and C will be (AB, C), (AC, B), (A, BC) and (ABC). Individual retrieval needs no joining location; while the (ABC) combination requires one joining locations and other combinations require two joining locations. One way to obtain the optimal solution is to solve the problem with all combinations and pick the one with the best result.

4. Comparison with single-load retrieval

To evaluate the performance of the systems presented in this paper, we compare total number of moves required to retrieve loads. The calculations are based on programs made in MATLAB. For the average saving, the program is run 100 times. In each run two loads are randomly selected. The results are calculated for different sizes of $N \times N$ storage systems.

Figure 8 illustrates savings in number of moves that are achieved for two loads, one location is given and the other is arbitrary. In Figure 8(a), one loads is located at (1,10) and the other can be anywhere in the system and in the same way, in Figure 8(b) the load is located at (5,5). Horizontal axes represent the place of the joining location in the system and vertical axis is percentage of saving for each location.

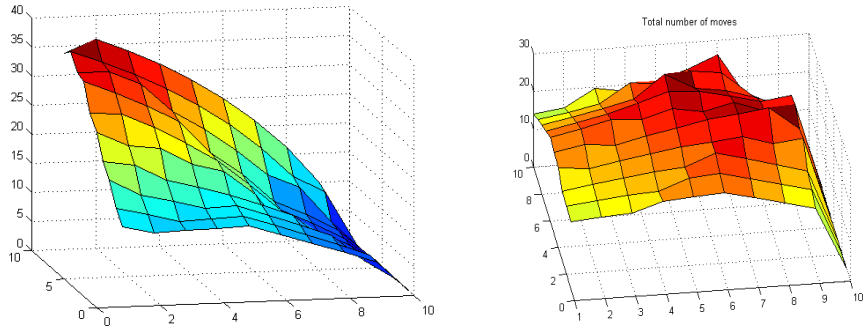


Figure 8- savings achieved for a given load located at (a) (1,10),and (b) (5,5)

Table 2 shows the savings that can be obtained by the dual-load retrieval method, as compared to individual retrieval. AvgSL and AvgDL are the averages of the total number of moves in single-load and dual-load retrieval, respectively. Maximum saving is calculated for loads at (1,*N*) and (1,*N*-1). The formula used for average saving is (AvgSL- AvgDL)/ AvgSL.

Table 2- saving on the number of moves for the dual-load retrieval

N	AvgSL	AvgDL	Maximum saving (%)	Average savings (%)
5	31	24	41	20
7	45	36	38	20
10	82	65	35	19
15	129	105	35	19
20	157	129	34	18
50	437	367	33	17

Results show that 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 method is not significant. On the other hand when both items are located at one of these locations, remarkable savings are achieved.

Conclusion

Puzzle-based storage systems help saving space and capacity. We proposed dual-load retrieval method, compared to single-load retrieval, saves in average 17 % in storage/retrieval time with no additional equipment. This saving can be increased even more by extending the method to multiple-load retrieval. Future researches can consider more than one escort, simultaneous load moves, etc. to further increase the efficiency in such systems.

References

Gue, K. R., & Kim, B. S. (2007). Puzzle-Based Storage Systems. *Naval Research Logistics*, 556–567.

A. Alfieri, M. Cantamessa, A. Monchiero, F. Montagna (2012). Heuristics for puzzle-based storage systems driven by a limited set of automated guided vehicles, Journal of Intelligent Manufacturing,

Zaerpour, N., Yu, Y. & de Koster, R. (2014). Small is Beautiful: A Framework for Evaluating and Optimizing Live-cube Compact Storage Systems. Transportation Science, Accepted.

Kota V. Rohit and G. Don Taylor, Kevin R. Gue, Retrieval Time Performance in Puzzle-Based Storage Systems, Proceedings of the 2010 Industrial Engineering Research Conference

Gue, K.R. ; Furmans, K. ; Seibold, Z. ; Uludag, O, (2013). GridStore: A Puzzle-Based Storage System With Decentralized Control, Automation Science and Engineering, IEEE Transactions on (Volume:PP , Issue: 99)

Litvak, Nelly, (2006). Optimal picking of large orders in carousel systems, Operations Research Letters 34, 219 – 227

Hwanga, Hark, Kimb , Chae-soo, Ko, Kyung-hee , (1999). Performance analysis of carousel systems with double shuttle, Computers & Industrial Engineering 36, 473-485

Flake, Gary William , Baum, Eric B. (2002). Rush Hour is PSPACE-complete, or "Why you should generously tip parking lot attendants", Theoretical Computer Science 270 , 895–911

Hearn, Robert A., Demaine, Erik D. (2005). PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation, Theoretical Computer Science 343, 72 – 96