Efficient Algorithms for Load Shuffling in Split-Platform AS/RS

Yahong Hu[1], Wen-Jing Hsu[1,2] and Xiang Xu[2]

[1]Singapore-MIT Alliance, E4-04-10, 4 Engineering Drive 3, Singapore 117576

[2]School of Computer Engineering, Nanyang Technological University, Singapore 639798

***Abstract*—We address the issue of shuffling loads in Automated Storage/Retrieval Systems (AS/RS) in this paper. The objective is to pre-sort the loads into any specified locations in order to minimize the response time of retrievals. 1D, 2D and 3D AS/RS racks have been designed in order to achieve the shuffling efficiently. The shuffling algorithms are described in detail. The response time of retrieval, the lower and upper bounds of energy consumption are also derived. Results of the analysis and numerical experiments show that the shuffling algorithms are quite efficient.**

***Index Terms*—Algorithm, AS/RS, Sorting**

I. Introduction

Automated Storage/Retrieval Systems (AS/RS) are computer-controlled storage systems that can automatically store and retrieve loads with high throughput. Conventional AS/RS typically use stacker cranes for reaching and accessing the storage cells. However, stacker cranes are only suitable for a certain range of task loads. To handle certain types of cargo (e.g. extra heavy loads [1]) at high speed, it is necessary to employ new Storage/Retrieval (S/R) mechanism in which vertical movement and horizontal movement of loads are carried out by separate devices, namely, the vertical platforms (VPs) and the horizontal platforms (HPs). For convenience, we shall refer to the new types of AS/RS as the *split-platform AS/RS*, or *SP-AS/RS* for short. Two AS/RS manufacturers have confirmed that this new design is both mechanically and economically feasible. One design of this kind of AS/RS is illustrated in Fig. 1. Detailed information about it can be found in [1].

One of the advantages of AS/RS is that it can offer high throughput. With the separate, independent vertical and horizontal platforms in the new design, more operations can be done concurrently. So this design provides us with the possibility to further improve the throughput. Our research shows that compared with stacker crane based AS/RS, the split-platform AS/RS can improve the performance quite a lot [2]. So the split-platform AS/RS not only can cope with very heavy loads that can't be handled by conventional stacker crane, but also it can offer better performance.

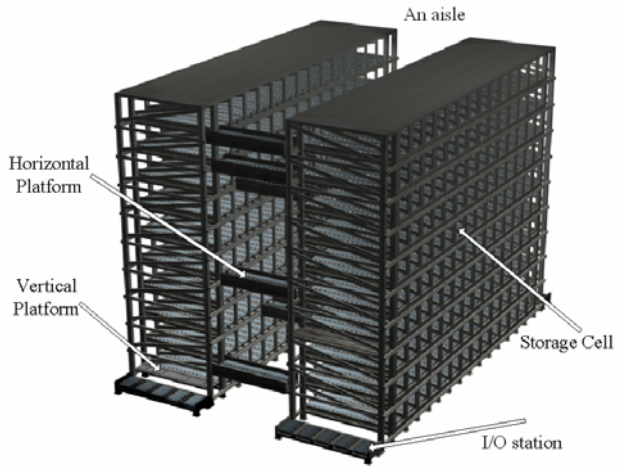

Fig. 1 Split-platform AS/RS

For AS/RS, it should store and retrieve loads in the shortest possible time period. Compared with storage, the quick response of retrievals is often more critical. This is because when a load is to be stored into an AS/RS rack, it can be put into any empty storage cell. While for retrieval, only the designated one is valid. How to retrieve loads as quickly as possible? A very natural solution is that since we generally know the retrieval sequence in advance, we can shuffle (pre-sort) the loads to specified locations to minimize the response time of retrieval. However, very little information about load shuffling can be found in the literature.

Our formulation of the shuffling problem is analogous to the sorting of data items where the loads in an AS/RS rack are analogous to the data items in an array. However, there is a fundamental difference between the two problems. In the conventional data sorting problem, two pieces of data items, disregarding their separation in the array, can be swapped in a constant amount of time. For the load shuffling, because the AS/RS platforms carry out the sorting operations, the time for swapping two loads depend on the actual distance between them. Therefore, while we

Y. H. Hu is with Singapore-MIT Alliance, National University of Singapore, 3 Science Drive 2, Singapore 117543 (phone:65-68744248; fax: 65-67794580; email: smahyh@nus.edu.sg)

W. J. Hsu is with the School of Computer Engineering, Nanyang Technological University, Singapore 639798 (email: smav52@nus.edu.sg)

X. Xu is with the School of Computer Engineering, Nanyang Technological University, Singapore 639798 (email: #xuxiang#@ntu.edu.sg)

use "sorting" as a synonym of "shuffling" to draw inspirations from the existing knowledge, bear in mind that the merits of AS/RS control algorithms are measured mainly by the time and the energy required for the platforms to carry out the moves.

We first consider the simpler case of sorting loads in 1D racks. The result is then applied as a basis for shuffling on 2D racks and 3D racks. Our results are analyzed in terms of the number of steps involving the platforms, as well as the energy required. For this purpose, we present a simple model for the calculation of energy consumption. A lower bound is also derived for comparisons.

The contributions of this paper are three-fold: (1) The abstraction of the new problem; formulation and the approach to the problem. (2) The new rack configurations for load shuffling. (3) The provably efficient load shuffling algorithms.

The rest of the paper is organized as follows. Section II considers the design of 1D, 2D and 3D AS/RS racks for efficient shuffling. Section III is dedicated to the shuffling algorithms for 1D, 2D and 3D racks, respectively. The shuffling algorithms are evaluated in Section IV. In Section V, we analyze the response time of retrieval for a 2D rack. In Section VI, the lower and upper bounds of energy required for loads retrieval in a 2D AS/RS rack are calculated. Section VII gives a brief summary and directions for future research.

II. Structure And Operations Of AS/RS For Shuffling

First we will describe the structure and operations for a 1D rack, and this can act as the basis for later derivations.

A. 1D Rack

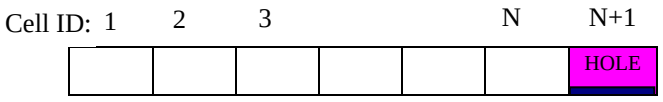

Fig. 2 Configuration of a 1D Rack (The dark block represents the platform)

Consider the configuration of a 1D rack shown in Fig. 2.
(1) There are in total $N+1$ cells numbered from 1 to $N+1$. The first N cells are for storing loads. The $(N+1)$-th, i.e. the rightmost cell is initially empty, and it is the initial location of the HOLE. The HOLE is used to temporarily store loads during the shuffling operation.
(2) There is one platform, which can move a load among the cells. We assume that the initial position of the platform is at the HOLE.

Definition 1(step) A *step* is defined to be the process of moving a load from its original cell to its destination cell.

Let Cell(x) denote the x-th cell, and Load(x) denote the load whose destination cell is Cell(x). Barring the trivial case where the origin cell coincides with the destination cell (i.e. the load is already in its target cell and no actual move is needed), one step can be further elaborated in terms of the following detailed moves:

1) If the destination cell of Load(d) is not presently empty, the platform first moves from its present dwell point to Cell(d); fetches the load in it and puts it into the HOLE.
2) The platform moves from the HOLE to Cell(x) that contains Load(d), and fetches it from Cell(x) then moves it into Cell(d).

B. 2D Rack

Normally the research of mesh sorting is mainly to sort data among different processors, and the processors in the mesh have three different connection models. One is the two-dimensional mesh-connected processor array. In this model, n×n processors are placed at the intersections of horizontal and vertical grids, and each processor is connected to its four neighbors. Another mesh-type model is the mesh of bus. In this model, no local links exist between the neighboring processors. Instead, n row and n column buses are provided to the n×n mesh. Each processor is connected to a couple of (row and column) buses [3]. The last model is the combination of the first and second one, i.e. extra n row and n column buses are added to the two-dimensional mesh-connected processor array [5].

Currently, it is more feasible to adopt the second option for AS/RS rack design. Therefore, our design will be based on this model. Fig. 3 gives the structure of the AS/RS for load shuffling.

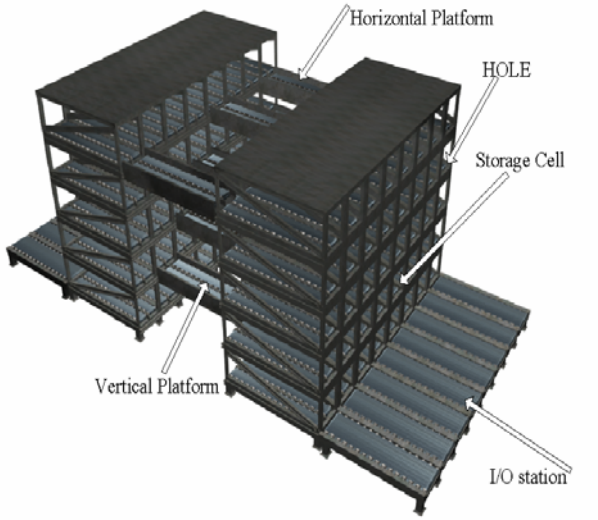

Fig.3 Structure of 2D AS/RS rack for shuffling

The difference between this new design and the one shown in Fig. 1 is that now each column has its own vertical platform instead of the shared VP in one rack. While the HPs act as the horizontal buses, the VPs serve as the column buses. At the same time, more I/O stations are provided to enable concurrent storage and retrieval of loads into the rack. The cells in the lowest row and the rightmost column are reserved as HOLEs. The side view of the 2D AS/RS rack is shown in Fig. 4.

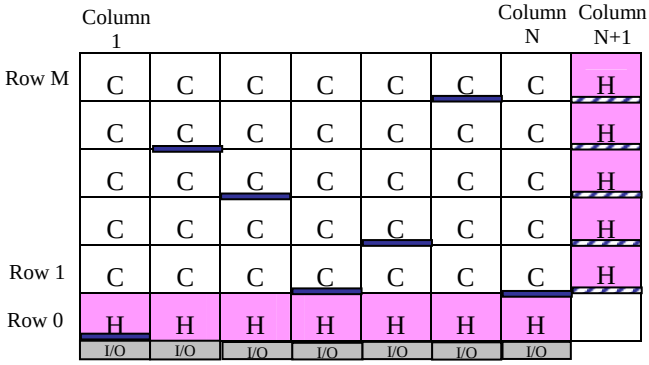

Fig. 4 The side view of a 2D rack

In Fig. 4, the letter 'C' denotes a cell, and 'H' denotes a HOLE. The little dark blocks represent the VPs, while the shaded blocks represent the HPs.

In our shuffling algorithm, the movement of the HPs and the VPs will not occur at the same time, so no platform conflict will happen.

Using the 2D AS/RS rack, the storage operation may be finished in the following steps.

1) After being put on the I/O station of the correct column, the load is transferred from the I/O station into the HOLE. At the same time, the VP of this column goes from its dwell point to the HOLE.
2) The load is transferred from the HOLE to the VP.
3) VP moves the load into the cell assigned to the load.

This description is for one column. Actually with so many I/O stations, the VPs can work concurrently to obtain high efficiency.

For the retrieval operation, the procedure is just the reversal. VP moves the load into the HOLE of the column, and then the load will be transferred to the I/O station to be carried away.

C. 3D Rack

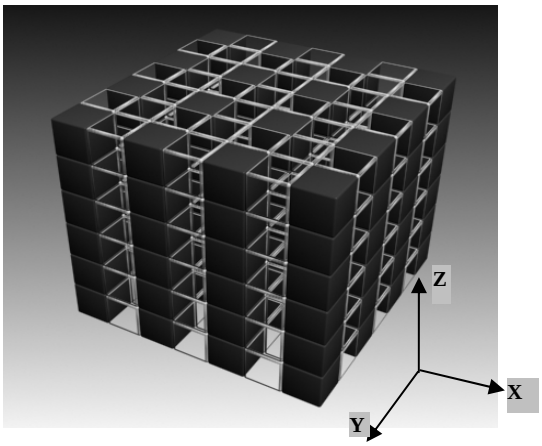

Fig. 5 Illustration of a 3D rack

There are quite a lot of ways to design a 3D racks, and one possibility is to have a similar structure to that of the 2D rack. In the 3D rack, each cell has three platforms to serve it. One is the so-called X-line platform, which is used to transfer a load along the X-line. The other two are the Y-line platform and the Z-line platform respectively. As in our shuffling algorithm, only the platforms of one direction run at a time, so no platform confliction will happen. Fig. 5 illustrates the basic idea of the structure of a 3D rack.

III. Shuffling Algorithms

For a comparative study, we first apply modified "Selection Sort" algorithms [4] to sort any permutations. The "Selection Sort" algorithm is chosen because it doesn't swap the loads. This is really useful in AS/RS designed for heavy loads, because swapping such loads can be time- and energy-consuming. The shuffling algorithms for 1D, 2D and 3D racks are described in each sub-section.

In order to simplify our discussion and analysis of the problem, we assume that all the loads in our AS/RS racks have unique destinations.

A. 1D Rack

The 1D shuffling algorithm is described as follows:

Algorithm 1: 1D shuffling algorithm

For i=1 to N /* *Step i, where N is the number of loads* */
IF (the ID of the destination of the load in Cell(i) is not equal to i)
Then
Move the load in Cell(i) to the current HOLE;
Move Load(i) to Cell(i);
Set the empty cell as the current HOLE;
End

Proposition 1 Using the 1D shuffling algorithm described in Algorithm 1, any initial permutation of N loads can be sorted in at most N steps.

Proof: Since in each step one load is moved to its final destination and is left there afterwards, and also some loads may already be in their correct locations before the shuffling, so it is clear that it takes at most N steps to sort all the N loads into the desired order. ■

B. 2D Rack

Refer to Fig. 4. The AS/RS rack has $(M+1)$ rows and $(N+1)$ columns. The total number of loads in the rack is $M \times N$, where N and M represent the number of loads in a row and in a column, respectively. Each load is labeled as Load(x, y) to represent the load's destination cell ID, where x is the load's destination row ID and y is its destination column ID. Let Cell(x, y) denote the storage cell located at Row x and Column y in the AS/RS rack. Supposing that we know the destination of each load before sorting, we can afford to pre-compute a solution off-line. We will present an off-line algorithm that allows all $M \times N$ loads to reach their destinations within $2N+M$ steps.

Algorithm 2: 2D shuffling algorithm

The 2D algorithm consists of the following three phases:

Phase 1. Permute loads in each row such that the loads at any given column have M different destination row IDs.

Phase 2. Permute the loads within each column to their destination rows.

Phase 3. Permute the loads within each row into order.

The correctness of Algorithm 2 is depend on whether the permutation described in Phase 1 can be found. Proposition 2 provides the guarantee.

Proposition 2: Given $M \times N$ loads with distinct destinations, there exists a permutation that the loads at any given column have different destination row IDs.

We will first prove the following lemma.

Lemma 1: Given the $M \times N$ loads described in Proposition 2. If the first (i-1) loads in Row 1 to Row (i-1) of Column 1 have different destination row IDs, then there exists a permutation for loads in Row 1 to Row i which ensures that the loads in the first i rows in Column 1 have distinct destination row IDs.

Proof: Let R(x ,y) denote the destination row ID of the load stored in Cell(x, y) and $S = \{R(x,1) | 1 \le x \le i-1\}$. We consider two cases.

Case 1. $R(i,1) \notin S$. In this case, a new row ID is found and the claim is true.

Case 2. $R(i,1) \in S$ .In this case, we must show that a new row ID does exist.

According to our assumption, each load in the rack has distinct destination cell ID. So for any rows ID r, there exist exactly N loads whose destination row IDs are r. In these i rows, altogether there are $i \times N$ loads. If no new row ID can be found in these i rows, even if all the loads with the destination row ID belonging to S are in these i rows, there are $(i\text{-}1) \times N$ loads in the rows, because S contains only $(i\text{-}1)$ row IDs. This contradicts to the fact that the number of loads in these i rows is $i \times N$. As a result, a new row ID does exist in these i rows. ■

By using the following search algorithm, the new row ID can be found.

Search algorithm:

$S = \{R(x,1) | 1 \le x \le i-1\}$

If $R(i,1) \notin S$, then the new row ID is found.

Else

$ZJ = \{R(i,t) | 2 \le t \le N\}$

For each t, where $2 \le t \le N$

If $R(i,t) \notin S$,

Exchange the location of the load in Cell(i,1) and the one in Cell(i,t). A new row ID appears in these i rows.

If every element $a \in ZJ$ also belongs to S

For each $a \in ZJ$

If $R(a,t) \notin S \quad (2 \le t \le N)$

Exchange the load in Cell(a,1) and that in Cell(a,t), and a new row ID appears in *Row a*. In order to retain the original row ID that *Row a* provides, the load in Cell(i,1) should exchange its position with the load stored in Cell(i,w) where $R(i,w) = a$.

Else go on with the search.

Continue this procedure and the new row ID will appear in these i rows.

Armed with Lemma 1, we now prove Proposition 2.

Proof: First consider Column 1. Initially we assume $\{S = R(1,1)\}$. From Lemma 1, we can find a new row ID for Row 2 in Column 1. Repeat this operation, all loads in Column 1 will have different row IDs.

Since Column 1 has been in the desired status, we consider Column 2. Now we are dealing with $M \times (N\text{-}1)$ loads, by assumption, they all have distinct destination cell Ids. It is clear that Column 2 can be made to contain loads with distinct row IDs. This procedure may be repeated, until finally the loads in each column have different row IDs. ■

Claim: Given any initial configuration of $M \times N$ loads as stated above, each load can be routed to its destination in three phases as given in Algorithm 2. The total number of steps is bounded $2N+M$.

Proof: In Phase 1, the loads in the rows can be permuted such that they have different destination row IDs in each column. This can be achieved by using the M HPs simultaneously. Then, in Phase 2, based on their destination row IDs, the loads in each column are permuted to their destination rows. This can be achieved by using the N VPs concurrently. Then in Phase 3, we can sort the loads in each row simultaneously. Now every load is in its final destination already. By using Algorithm 1, it is clear that each of Phase 1 and Phase 3 can be accomplished in N steps. Phase 2 can be accomplished in M steps. Hence, the total number of steps is bounded by $2N+M$. ■

C. 3D Rack

In this section, we consider the shuffling algorithm for a 3D AS/RS rack as described in Section II.C. It can store N loads, M loads and K loads in each X-line, Y-line and Z-line respectively. So the total number of loads in the 3D rack is $N \times M \times K$. Each load is labeled as Load(x, y, z) to represent the load's destination cell ID, where x, y ,z are the load's destination IDs in X-direction, Y-direction and Z-direction respectively. Let Cell(x, y, z) denote the storage cell located at the intersection of X-line, Y-line and Z-line in the AS/RS rack. Like for the 2D shuffling algorithm, we suppose that the destination of each load is distinct and is known before sorting. We will present an off-line algorithm that allows all $N \times M \times K$ loads to reach their destinations within $4N+M+2K$ steps.

Algorithm3: 3D shuffling algorithm

The 3D algorithm consists of the following phases:

Phase 1: Permute the loads in each XZ-plane such that the loads in any given Y-line have M distinct XZ-plane IDs.

Phase 2. Permute the loads within each Y-line to their destination XZ-planes.

Phase 3. Permute the loads within each XZ-plane to their destination cells.

Again, we must prove that the permutation described in Phase 1 can be found and this is guaranteed by Proposition 3.

Proposition 3: Given $N \times M \times K$ loads with distinct destinations, there exits a permutation that the loads within any Y- line have *M* distinct destination XZ- plane IDs.
Before proving this Proposition, we need two lemmas.

Lemma 2: Consider the $N \times M \times K$ *loads* described in Proposition 3. If the first *(m-1)* loads in the first Y-line inside the first XY-plane have different XZ-plane IDs, then there exists a permutation that the first *m* loads in this line also have distinct XZ-plane IDs.
Proof: Let R(x,y,z) denote the destination XZ-plane ID of load stored in Cell(x,y,z) and $S = \{R(1, y,1) | 1 \le y \le m-1\}$. We consider two cases.

Case 1: $R(1, m,1) \notin S$. In this case, these *m* loads already have distinct XZ-plane IDs.

Case 2: $R(1, m,1) \in S$. In this case, we need show that a new XZ-plane ID can be found.

According to our assumption, each load in the rack has distinct destination cell ID. So for the first *m* XZ-planes, there should exist $N \times m \times K$ loads. If no new XZ-plane ID can be found, it means that all the loads in these *m* planes have the destination XZ-plane ID belonging to *S*, hence there are altogether $N \times (m-1) \times K$ loads. This leads to a contradiction. ■

Lemma 3: Consider the $N \times M \times K$ *loads* described in Proposition 3. If the loads in the first *(n-1)* Y-lines residing in the first XY-plane have different destination XZ-plane IDs, then there exist a permutation which ensures that loads in the first *n* Y-lines in the first XY-plane have distinct XZ-plane IDs.
Proof: Assume that one particular XZ-plane ID y' cannot appear in the *y-th* Y-line ($1 \le y \le k$) of this plane for all the permutations. It means that the total number of loads destined for XZ-plane y' is at most $(n-1) \times K$. This contradicts the fact that the total number of such loads should be $N \times K$. As a result, all the distinct IDs for XZ-plane will appear in the *y-th* Y-line of the first XY-plane. Then all the loads in these first *(n-1)* Y-lines have distinct destination XZ-plane IDs. ■

With Lemmas 2 and 3, we now prove Proposition 3.
Proof: First consider the first Y-line in the first XY-plane. Initially, let $S = \{R(1,1,1)\}$. From Lemma 2, we can ensure that loads in Cell(1,1,1) and Cell(1,2,1) have distinct XZ-plane IDs. Repeating this operation, then all the loads in this first Y-line can have different XZ-plane IDs.

Then from Lemma 3, since the loads in the first Y-line in the first XY-plane have distinct XZ-plane IDs, it is clear that the loads inside the second Y-line in this plane also can obtain different XZ-plane IDs. Repeating this operation, loads in all the Y-lines inside the first XY-plane can have distinct XZ-plane IDs.

For the second XY-plane, now we need to solve the problem with $N \times M \times (K-1)$ loads with distinct destination cell IDs. It should be clear that all the loads within the second XY-plane can have distinct XZ-plane IDs. By repeating this operation, all the XY-planes can obtain loads with different XZ-plane IDs. ■

Search algorithm similar to that of the 2D algorithm can be used to find the permutation.

Claim: Given the initial configuration of $N \times M \times K$ loads as stated above, each load can be routed to its destination in three phases given in the 3D shuffling algorithm. The total number of steps is *4N+M+2K*.
Proof: In Phase 1, the loads are permuted in each XZ-plane such that the loads in any given Y-line have distinct XZ-plane IDs. This can be achieved by using the 2D algorithm and the loads within each XZ-plane can be shuffled simultaneously. Here, *2N+K* steps are needed. In Phase 2, all the loads are shuffled to their destination XZ-plane. All the Y-direction platforms can operate concurrently using the 1D shuffling algorithm. To accomplish this phase, *M* steps are required. Then in Phase 3, we can sort the loads in each XZ-plane simultaneously. Every load will be in its final destination within *2N+K* steps using the 2D algorithm again. Hence, the total number of steps is bounded by *4N+M+2K*. ■

IV. Evaluation Of The Shuffling Algorithms

In this section, we calculate upper bound and lower bound of the distances traversed by a platform to finish shuffling *N* loads in a 1D rack. It is the basis for the corresponding calculations for 2D and 3D racks.

A. Upper Bound of the Distance Traversed by Platforms

Proposition 4 The total distance *s* traversed by the platform using Algorithm 1 satisfies the following inequality:

$$s \le \begin{cases} (N^2 + 3N)L & \text{if } N \text{ is even} \\ (N^2 + 3N - 2)L & \text{if } N \text{ is odd} \end{cases}$$

Proof: Because of space limitation, the reader is referred to [6] for the detailed proof. ■

B. Lower Bound of the Distance Traversed by Platforms

Proposition 5 To finish shuffling of *N* loads, the overall distance *s* traversed by the platform in the worst case is lower-bounded by:

$$\begin{cases} \dfrac{N^2 L}{2} & \text{if } N \text{ is even} \\ \dfrac{(N^2 - 1)L}{2} & \text{if } N \text{ is odd} \end{cases}$$

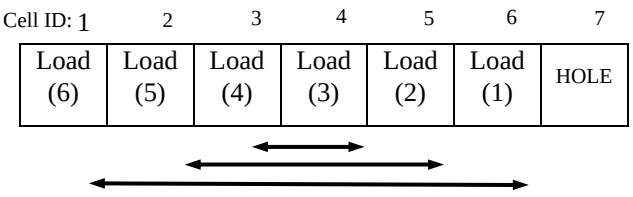

Fig. 6 Load distribution in the worst case

Proof: One load distribution in the worst case is that the loads located in the symmetric cells in a 1D rack need to exchange their positions, i.e., one load's current cell location is just the destination of the other load. Fig. 6 gives an example.

Whatever shuffling algorithm is used, for Load(6) and Load(1), they must traverse the longest distance to arrive at the destinations. For Load(5) and Load(2), the distances they have to cover are the second longest and so on. Analogously, the total distance covered by the platform is the maximum under this kind of load distribution.

For Load(1) and Load(N), the distance they have to go is *(N-1)L*. For Load(2) and Load(N-1), the distance is *(N-3)L* and so on. So the total distance is:

When N is even,

$$2L[(N-1)+(N-3)+\ldots\ldots+(N-(N-3))+(N-(N-1))] = \frac{N^2 L}{2}$$

When N is odd, the load in the middle cell doesn't need to move, so the formula is:

$$2L[(N-1)+(N-3)+\ldots\ldots+(N-(N-3))+(N-(N-2))] = \frac{(N^2-1)L}{2}$$

■

V. RETRIEVAL TIME COMPARISON

Since presently most AS/RS racks are essentially 2D racks, in this section, we calculate the operation time to retrieve a batch of $M \times N$ loads when they have been pre-sorted.

A. Response Time for Retrieval Using the 2D AS/RS

The notations used in this section are:

N: the number of VPs

M: the number of HPs

T_{load}: the time for transferring a load from a cell onto a platform or vice versa, i.e. transferring a load from a platform into a cell

L: the length of an AS/RS cell

H: the height of an AS/RS cell

Assume that the HPs travel at an average speed of V_h while the VP travels at an average speed of V_v. The initial locations of the platforms are their corresponding HOLEs.

Since all loads have been sorted, we can retrieve the loads in a more efficient fashion. The *N* VPs can operate concurrently, so the time taken to retrieve all the $M \times N$ loads is equal to that for retrieving the *M* loads located in the same column. The time can be obtained through the following steps.

1) VP travels from the HOLE to Row 1. $T_1 = \frac{H}{V_v}$

2) Fetches the load and returns back to HOLE, puts it into HOLE. $T_2 = 2T_{load} + \frac{H}{V_v}$

3) The load is moved to the I/O station.
Simultaneously, VP goes to Row 2, fetches a load, comes back at HOLE, and moves the load into HOLE.

$$T_3 = 2T_{load} + 2\frac{2H}{V_v}$$

4) Continuing the process with each row, until all the loads in the column have been retrieved.

So the total time of retrieval is,

$$T_{shuffling} = \sum_{i=1}^{M}(2T_{load} + 2i \times \frac{H}{V_v}) = 2MT_{load} + \frac{HM(M+1)}{V_v} \qquad (1)$$

B. Response Time for Retrieval without Shuffling

In this section, we will give the retrieval time to fetch all the loads in a rack without shuffling. Two kinds of AS/RS rack configuration are considered. One rack configuration is the one illustrated in Fig. 1, and henceforth we call it **Configuration 1**. The other is the AS/RS for shuffling and it is called **Configuration 2**.

(1) Configuration 1

Because in this configuration only one VP is provided, loads in the rack have to be retrieved one by one. The total retrieval time is upper-bounded by:

$$T_{retrieval1} = 2\sum_{j=1}^{M}\sum_{i=1}^{N}\left(\frac{(j-1)H}{V_v} + \frac{iL}{V_h}\right) + 3MNT_{load} = \frac{M(M-1)NH}{V_v} + \frac{N(N+1)ML}{V_h} + 3MNT_{load}$$

For each load, it will be fetched from the cell, and then transferred from HP to VP, after that it will be moved from VP to the I/O station. So altogether, the time spent on this kind of transfer is $3T_{load}$. The other two parts of time in the above equation are for the movements of the VP and HPs.

(2) Configuration 2

By using the 2D AS/RS shuffling rack, all the VPs are capable of operating concurrently. The retrieval time is entirely dependent on the loads' initial location distribution in the rack and their retrieval sequence. The case that shuffling can gain the most benefits is that all the loads are stored sequentially in the columns according to their retrieval sequence. Fig. 7 gives such an example. The

load's number in each storage cell is its serial number in the retrieval sequence.

4	8	12	16	20
3	7	11	15	19
2	6	10	14	18
1	5	9	13	17

Fig. 7 An example of the sequential locations of the loads in a 2D AS/RS rack

It is clear that in this case, the VPs cannot work concurrently in order to guarantee the correct retrieval sequence. So shuffling the loads before the actual retrieval process can help to minimize the retrieval time.

The total time to fetch all such loads consists of two parts.

1) The time to retrieve all the loads in Column 1. It is the same as that provided by Equation (1).

$$T_1 = 2MT_{load} + \frac{HM(M+1)}{V_v}$$

2) Time needed to retrieve the loads from any of the other columns.

$$T_2 = 2MT_{load} + \frac{HM(M+1)}{V_v} - \left[\frac{2H}{V_v} + 2T_{load} + \frac{2H}{V_v}\right]$$

Because the VP in other columns can pre-fetch the load in Row 1 and then go to Row 2 to wait for the time to fetch the load there during the period that the loads in Column 1 are being retrieved, so the time needed to retrieve loads in columns except Column 1 doesn't need to contain these parts. The total retrieval time is:

$$T_{retrieval2} = T_1 + T_2(N-1)$$

$$= 2(MN - N + 1)T_{load} + \frac{HMN(M+1) - 4H(N-1)}{V_v}$$

The possible time reduction by load shuffling should be between 0 and $T_2(N-1)$.

C. Numerical Calculation

In this section, we compare the time to retrieve a batch of loads with shuffling or without shuffling under different rack configurations. In the experiments, the number of loads to be retrieved is the same as the number of storage cells in the rack and we calculated three cases. In the first case, loads are stored in the 2D rack of AS/RS for shuffling and have been sorted before retrieval. The second case deals with the loads under Configuration 1. Here, we didn't use Equation (2) to calculate $T_{retrieval1}$, because the concurrent movements of HP and VP for one job are not taken into account in this case. We directly calculate the time needed to retrieval a load from a cell and the concurrency between the HP and VP are considered. The last case is to retrieve the un-shuffled loads in the rack under Configuration 2.

All the specifications of the AS/RS used in the calculation are:

a) The height of each cell is 4.5m, and the width is 4.5m.
b) The vertical platform travels at 1m /sec and the horizontal platforms travel at 2 m/sec.
c) The time to transfer a load between VP and HP or between HP and a cell is 15 seconds.

The number of rows and columns listed in Table 1 exclude the row and column for the HOLEs.

TABLE 1
Comparison of the Retrieval Time

Rack Configuration		Retrieval time with shuffling (sec.)	Configuration 1		Configuration 2	
Row × Column			Retrieval time (sec.)	Improvement (%)	Retrieval time (sec.)	Improvement (%)
10	10	795	9135	1049.1	7518	845.7
10	20	795	22635	2747.2	14988	1785.3
10	30	795	40635	5011.3	22458	2724.9
20	10	2490	25245	913.9	24468	882.7
30	10	5085	50295	889.1	50418	891.5

From Table 1, it is clear that compared with Configuration 1, on average the shuffling scheme can have 2122.1% improvement. Compared with Configuration 2, the average improvement is about 1426.0%. The improvement scales up with the increased number of columns. This is because the shuffling scheme can provide more parallel operations for the VPs. It should be quite clear that by pre-sorting the loads, the retrieval time for a batch of loads could be drastically cut down.

VI. Analysis Of The Energy Consumption For Shuffling

Nowadays, energy has become more and more important issues in many applications. As we intend to handle extra heavy loads using the new type of AS/RS, we will calculate the upper bound of energy consumption for shuffling N loads in a 1D rack. Then based on the results, we give an upper bound and a lower bound energy consumption for operations in a 2D rack.

The notations used in this section are:

W_{load} : energy required for picking up or depositing a load

m_{hp} : the mass of a HP

m_{vp} : the mass of a VP

m_{load} : the mass of a load

We assume that the energy needed for picking up and that for depositing a load are the same, and all the loads

have the same mass m_{load}. Without loss of generality, we assume that M and N are even.

A. Upper Bound of Energy Consumption for Load Shuffling in a 1D rack

An upper bound of energy consumption to shuffle N loads of a 1D AS/RS rack into order is:

$$W = \begin{cases} f_{total}(N^2+3N)L & \textit{horizontal movement} \\ m_{total}g(N^2+3N)L & \textit{vertical movement} \end{cases} \quad (3)$$

where f_{total} denotes the friction force exerted to the platform, and m_{total} is the total weight of the platform and the load on it. The detailed deduction can be found in [6].

B. Upper Bound of Energy Consumption for Load Shuffling in a 2D rack

Here we derive an upper bound of the energy required by all platforms using the shuffling scheme. The energy required in the sorting process is:

$$W_{shuffling} = W_{hp} + W_{vp} + W_l$$

W_{hp} is the energy consumed by the movements of all HPs, W_{vp} is the energy for VPs' movement and W_l represents the energy used to transfer the loads between the platforms and cells. According to the 2D algorithm, each load has to be picked up or deposited 12 times in the worst case. Referring to Equation (3), the energy required by all platforms using the 2D algorithm is upper-bounded by:

$$W_{shuffling} = 2f_{total}\left(N^2+3N\right)ML + \left(m_{vp}+m_{load}\right)g(M^2+3M)NH + 12MNW_{load}$$

Referring to the retrieving policy in Section V.A, it is easy to give the energy required to complete all the retrievals after sorting. The retrieval process is divided into two parts.

1) VPs travel from the HOLE to Row i without loads, and then go back to the HOLE with loads.

$$W_{up} = \sum_{i=1}^{M} i m_{vp} gHN$$

$$W_{down} = \sum_{i=1}^{M} i(m_{vp}+m_{load})gHN$$

2) Total amount of energy used to transfer loads from the cells to the I/O stations.

$$W_1 = 3MNW_{load}$$

Then the energy used to do the retrieval is formulated as:

$$W_{retrieval} = W_{up} + W_{down} + W_1$$

$$= 3MNW_{load} + M(M+1)NHm_{vp}g + \frac{M(M+1)NH}{2}m_{load}g$$

Therefore, an upper bound of energy required for retrieval using the algorithm is:

$$W_{upper} = W_{shuffling} + W_{retrieval}$$

$$= 2f_{total}(N^2+3N)ML + 15MNW_{load} + (2M^2+4M)m_{vp}gNH + \frac{3M^2+7M}{2}NHm_{load}g$$

C. Lower Bound of Energy Consumption for Load Shuffling in a 2D rack

Any algorithm for retrieving $M \times N$ loads in a rack must pick up each load and move it to an I/O station. Therefore a trivial lower bound is given by:

$$W_{lower} = \sum_{j=1}^{M-1}(m_{load}+m_{vp})gjNH + 2MNW_{load}$$

$$= \frac{NM(M-1)(m_{load}+m_{vp})gH}{2} + 2MNW_{load}$$

Comparing the lower bound and upper bound, we can see that reducing the mass of HPs and VPs is a simple yet effective way to improve the energy efficiency.

VII. Discussions And Conclusions

We have presented the design of 1D, 2D and 3D AS/RS racks for shuffling and also have described the algorithms for shuffling unit loads stored in these AS/RS racks. The algorithms have been analyzed in terms of number of steps and energy requirements. For 2D rack, we have also conducted numeric experiments to compare the performance of AS/RS with or without shuffling scheme. The results suggest that our algorithms are rather efficient for shuffling loads with our proposed rack design.

The AS/RS racks presented here have been specially optimized for the shuffling operations. It can be seen that the large number of moveable platforms also entails a high cost. Moreover, the model we presented for energy calculation is still rather crude, although it serves our present purpose well. Therefore, the following issues deserve future research:

(1) Fault tolerance is very important in AS/RS operations. More flexible shuffling algorithms are needed to handle the cases when some of the platforms are out of order.

(2) It will also be useful to derive more realistic models for energy consumption.

(3) To optimize the search algorithm, and to determine the optimal move sequence since we know the original and destined loads distribution in the AS/RS rack.

Acknowledgement

We gratefully acknowledge the support by the Agency for Science, Technology and Research, Singapore, the Maritime and Port Authority of Singapore, Nanyang Technological University, and the Singapore-MIT Alliance, Singapore.

References

[1] C.Y. Chen, W.J. Hsu, V.Y. Vee, P. Lu, S.Y. Huang and M.K. Lai. "Automated storage/retrieval system for container operation,"

Technical report (TR-HCTS-002). Nanyang Technological University, Singapore. 2001.
[2] Y. H. Hu, S. Y. Huang, C. Y. Chen, W. J. Hsu, A. C. Toh, C. K. Loh, T. C. Song. "Travel time analysis of a new Automated storage and retrieval system," to appear in "Computers & Operations Research".
[3] K. Iwama and E. Miyano. "Recent Developments in Mesh Routing Algorithms," IEICE Trans. INF. & SYST., Vol. E83-D, No. 3, 530-540, March 2000.
[4] D. E. Knuth. *The Art of Computer Programming, Volume III: Sorting and Searching*, Addison Wesley, Reading, USA. 1973.
[5] J. Y-T. Leung. "Packet Routing on Square Meshes with Row and Column Buses," Proc. 3rd IEEE Symposium on Parallel and Distributed Proceeding, Dallas Texas. 834-837, 1991.
[6] Y. H. Hu, X. Xu, and W. J. Hsu. "Efficient Algorithms for Load Shuffling in Automated Storage/Retrieval Systems," Technical report. Nanyang Technological University, Singapore. 2003.

Ya-Hong Hu is a Research Fellow in Singapore-MIT Alliance, National University of Singapore, Singapore. She received her Ph.D, M.S and B.S from Xi'an Jiaotong University, China. Her research interests include modeling and simulation, automation, distributed resources sharing and remote collaborative design environment.

Wen-Jing Hsu is an Associate Professor in the School of Computer Engineering, Nanyang Technological University, Singapore. He received his BS, MS and Ph.D from National Chiao Tung University, Taiwan. His research interests include algorithms, computer architectures, parallel and distributed systems and networks.

Xiang Xu is a Ph. D candidate in the School of Computer, Nanyang Technological University. His research interests include automation algorithms and P2P systems.