On Sequencing Policies for Unit-Load Automated Storage and Retrieval Systems

Jean-Philippe Gagliardi
Jacques Renaud
Angel Ruiz

November 2012

CIRRELT-2012-68

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2012-017.

On Sequencing Policies for Unit-Load Automated Storage and Retrieval Systems

Jean-Philippe Gagliardi, Jacques Renaud*, Angel Ruiz

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, 2325, de la Terrasse, Université Laval, Québec, Canada G1V 0A6

Abstract. Automated Storage and Retrieval System (AS/RS) performance highly depends on the characteristics of the mechanical equipment. However, once the system has been physically implemented, achieving its maximum efficiency depends on the way the system is operated. This paper focuses on the tactical decisions for an AS/RS. In particular, it shows that request sequencing (i.e., planning the order in which storage and retrieval requests are performed) is of paramount importance in real-life AS/RS performance. This paper reviews and adapts the most popular storage and sequencing policies to dynamic contexts, and then it proposes a mathematical model to simultaneously solve the sequencing and storage location problems. Extensive computational results based on a thorough simulation experiment plan confirm that having the right sequencing can have a measurable impact (up to 23%) on AS/RS performance. When coupled with a storage assignment policy, an integrated approach encompassing request sequencing and storage location decisions can yield a reduction of up to 45% in the total travel time, which represents significant gains in performance.

Keywords. Automated storage and retrieval systems, sequencing, storage assignment, random storage, turnover-based storage, discrete-event simulation, warehousing.

Acknowledgements. This research was partially financed by grants [OPG 0293307 and OPG 0172633] from the Natural Sciences and Engineering Research Council of Canada (NSERC). This support is gratefully acknowledged.

---

* Corresponding author: Jacques.Renaud@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2012

1. Introduction

Automated Storage and Retrieval Systems (AS/RS) are widely used in distribution centers to improve the efficiency of warehousing and order-picking operations. These systems provide fast and efficient material handling and can operate 24 hours a day with minimal human supervision. As they involve a significant investment, AS/RS require a serious analysis during the initial design phase (Rouwenhorst et al., 2000; De Koster et al., 2007; Gu et al., 2010), since this phase will determine the system capacity and throughput. For example, during the design phase, the managers make decisions about rack configuration and capacity (e.g., single or double depth), the number of aisles and of cranes (e.g., aisle-captive cranes or not), as well as the location of the input/output point.

Once the AS/RS is implemented, a number of control decisions need to be made to obtain the maximal performance (Roodbergen and Vis, 2009). These control decisions include decisions about storage policies, batching, parking of idle cranes and sequencing, among others. Well-known, traditional storage policies for AS/RS include dedicated storage, random storage, closest-open-location storage, full-turnover storage and class-based storage (Hausman et al., 1976; Graves et al., 1977). Batching decisions refer to a situation in which a number of orders need to be retrieved in a person-on-board AS/RS. Dwell-point positioning indicates where to position an idle crane. Finally, sequencing decisions indicate how to best match storage and retrieval requests in order to optimize crane utilization. These control decisions can have a major impact on system performance, and they should be made carefully.

In this article, we will focus on control decisions. More specifically, we are interested in request sequencing in a unit-load AS/RS that has a single aisle between two single-depth racks. We studied sequencing heuristics using empirical simulations. These heuristics were reviewed and, for some of them, an updated definition has been provided in order to deal with systems with more than one location per item. We also introduce a new integer linear program to model AS/RS sequencing decisions. This model was proved to be very efficient at supporting sequencing decisions in complex, real-life AS/RS configurations. To the best of our knowledge, it is the first time that a mathematical model has been proposed for such realistic configurations.

The paper is organized as follows. Section 2 reviews the basic sequencing problem for AS/RS and the methods proposed in the literature to solve it. Section 3 presents the adapted heuristics, and section 4 proposes our new mathematical model. Sections 5 and 6 present, respectively, the simulation platform used to perform the experiments and the numerical results. Section 7 draws our conclusions and proposes several research avenues.

2. AS/RS sequencing problem and the traditional heuristics

The throughput of a single-aisle AS/RS is determined by two main factors: handling time and travel time. Handling time is the time required by the crane to extract or deposit a unit-load from or in a storage slot. This handling time depends on the crane's characteristics; it is constant for a given AS/RS, and thus it is not relevant to sequencing decisions. On the other hand, travel time represents the time required by the crane to move to the storage or retrieval locations. Therefore, minimizing the total crane travel time leads to maximizing the total throughput (Roodbergen and Vis, 2009; Graves et al., 1977). The travel time depends on two operational decisions: 1) the assignment of products to storage locations and 2) the sequencing decisions (i.e., the order in which the requests are executed).

Understanding why the choice of the locations where products are to be stored within the rack influences the AS/RS throughput is quite intuitive. For example, let us consider the case where the *Locations-To-Product Ratio* (LTPR) = 1, or in other words, the case where each product is assigned to only one storage location. For this case, extended analytical models (Graves et al., 1977) and empirical studies (Van Den Berg and Gademann, 2000) have been used in order to assess the performance of several location policies in terms of the total travel distance. These models and studies have shown that a random policy, which corresponds to using a single class system, leads to the lowest performance. Better results are obtained with a class-based system, using 3 to 5 classes. However, the best performance was achieved by a full-turnover storage policy, which locates products with the highest turnover in locations nearest to the I/O point (Hausman et al., 1976). Gagliardi et al. (forthcoming, b) confirm these results through empirical simulations.

Gagliardi et al. (forthcoming, b) also demonstrate that these results are no longer valid in real-life situations, where LPTR > 1 (i.e., each product can have many locations, especially when some fast-moving products have more locations than the others). This is the case for most of the industrial settings, where the number of locations assigned to each product follows an ABC curve (e.g., when 20% of the products occupy 50% of the available space). In this case, a purely random storage policy outperforms the traditional full-turnover policy.

Although location policies have been extensively studied in the literature, the other factor impacting the total travel time – sequencing decisions – has been mentioned rarely until now. This article is one of the few explicitly devoted to the sequencing problem. The storage requests are represented by an ordered set S since it is generally assumed that storage requests are executed according to a first-come, first-served rule, due to the physical constraints of the conveyor system feeding the AS/RS (Roodbergen and Vis, 2009). However, the set R of retrieval requests can be sequenced in any order. The number of different products, or *stock keeping units* (SKU), is not restricted, but is lower than or equal to the number of slots, and each SKU has at least one slot. We can have any number of pallets of each SKU, and each SKU is assigned to a specific storage zone (in random storage, it is a single zone, while in full-turnover storage, the number of zones is equal to the total number of SKU). We consider a dynamic situation, where both R and S change over time, and the set of open locations evolve with each storage and retrieval task. Thus, the problem consists of sequencing, or scheduling, retrieval and storage requests while respecting product zoning, with the objective of minimizing the crane's total travel time.

Graves et al. (1977) showed that performing as many dual-command cycles as possible helps to minimize crane travel time. In addition, to deal with the dynamic behavior of AS/RS open locations, Han et al. (1987) suggested two alternatives: block sequencing and dynamic sequencing. In *block sequencing*, storage request and retrieval request sets are separated into blocks, or subsets. Then, a single block of storage requests and a single block of retrieval requests are sequenced. When the requests in these blocks have been scheduled, another pair of blocks is selected. In *dynamic sequencing*, the list is updated each time a new request is added. In this case, Han et al. (1987) suggested using due dates or priorities to insure that no requests are greatly delayed.

There only a few results for request sequencing in AS/RS. When the crane can visit many locations, such as *person-on-board* picking systems (Bozer et al. 1990), the sequencing problem corresponds to the well-known traveling salesman problem (Laporte, 1992). With an appropriate distance matrix transformation, this can also hold true for unit-load systems when dedicated storage is used (i.e., each product has only one predetermined position).

Lee and Schaefer (1996) studied the special case of a one-class system, in which each retrieval request is associated with a specific location and storage requests can be stored in any open position. They proposed solving an assignment problem first, followed by a tour-checking algorithm. If the number of open positions is not too large, this algorithm can find the optimal solution. However, when the number of openings is very large, the algorithm cannot find the optimal solution, so they proposed a ε-optimum algorithm. In another study by Lee and Schaefer (1997), the storage requests are considered according to a first-come, first-served rule, and these requests are assigned to predetermined storage locations. Retrievals can be sequenced, but their specific locations are known. In this case, the problem corresponds directly to an assignment problem. Clearly, these assumptions do not hold true in our context: each product may have many pallets in the system, and each product can be stored or retrieved from any location within a class.

In order to deal with more generic situations, three heuristics have been proposed in the literature. Each heuristic focuses on minimizing one or more components of the crane travel cycle: (1) T_s, the travel time between the input/output point (I/O point) and the storage location; (2) T_{sr}, the interleaving travel time between the storage location and the location of the retrieval; and (3) T_r, the return time from the retrieval location to the I/O point. If we define the distance metric as $D_{sr} = T_{sr}$, and we sequence the retrievals in order to minimize D_{sr}, we obtain the *nearest neighbor* (NN) heuristic (Han et al., 1987), which minimizes the interleaving travel time. If we set $D_{sr} = T_s + T_{sr}$, we obtain the *shortest-leg* (SL) heuristic, which minimize the sum of travel times from the I/O to the storage location and from the storage location to the retrieval location (Han et al., 1987). Finally, if we set $D_{sr} = T_s + T_{sr} + T_r$, we obtain the minimum *total travel time* (TT) (Lee and Schaefer, 1996).

According to these definitions, all three heuristics can be expressed as follows (Lee and Schaefer, 1996):

```
Let $R$ be the block of $n$ retrieval locations and $O$ be the set of $p$ initial available locations to store a SKU.

While $R \neq \emptyset$
    1. Select a pair $o \in O$ and $r \in R$ with minimum $D_{sr}$.
    2. Perform a Dual-command cycle, storing in $o$ and retrieving from $r$.
    3. $R \leftarrow R - \{r\}$.
    4. $O \leftarrow O - \{o\} + \{r\}$.
End.
```

In this version, the set S of storage requests is not used explicitly, since it is assumed that a storage request is readily available to form a dual-command cycle (Lee and Schaefer, 1996). In addition, since a storage request can be placed in any open location (remember that we are dealing with a one-class system with a random storage policy within the class), there is no need to individually identify each storage location. Each retrieval corresponds to a specific location, since it is assumed that the AS/RS holds only one pallet of each SKU (LTPR = 1). At each step, the set of open locations O is updated. The next section explains how to adapt these heuristics to be used on other AS/RS configurations that are closer to the ones found in industrial settings.

3. An adaptation of the traditional sequencing heuristics

In order to evaluate the performance of available sequencing methods for more generic, more realistic AS/RS configurations (i.e., class-based systems with more than one unit of each SKU), we had to extend the original definitions of all three heuristics (NN, SL, TT). In addition, we enhanced the heuristic flexibility with respect to the Han et al. (1987) alternatives. Therefore, the heuristics have been redefined in such a way that they can be executed with *block sequencing* (i.e., a complete block of requests is sequenced without taking into account the locations that become available during the executions of the retrievals in the block), with *dynamic sequencing* (i.e., a complete block of requests is sequenced, but only the first dual-cycle is executed, and the block is updated each time a new request arrives), and with *generalized sequencing* (i.e., a complete block of requests is sequenced, but only the *f* first command-cycles is executed), which is more flexible.

Executing the sequencing heuristics according to either block sequencing, dynamic sequencing, or generalized sequencing requires the use of two parameters. The first parameter h corresponds to the *sequencing horizon* (i.e., the length of the next block to sequence). Clearly, $h \leq n$, where n is the number of retrievals in R. Storage requests are performed according to a first-come, first-served (FCFS) rule, so it is useless to consider more storage requests than retrievals. The second parameter $f \leq h$ represents the *frozen horizon* (i.e., the number of command cycles to execute). In other words, it is the number of command cycles that will be performed by the crane before re-sequencing. It follows logically that block sequencing corresponds to the case where $= h$, and a dynamic sequencing policy corresponds to $f < h$.

It is worth mentioning that setting appropriately the sequencing horizon h is highly important in real-time environments, where a large number of requests may be waiting to be sequenced in the AS/RS. In this case, considering only one request at a time may lead to a myopic poor-quality solution. On the other hand, block sequencing may not be a better option, since it neglects incoming requests. The section devoted to numerical simulations focuses on identifying the values of f and h leading to the best results.

3.1. Generalized Sequencing Algorithm (GSA)

Without loss of generality, let us assume that all the requests considered in the sequencing horizon can be performed in terms of available quantities and open space. In addition, storage requests are treated according to the FCFS rule. We already defined h as the length of the sequencing horizon, f as the length of the frozen horizon, S as the ordered set of the next m storage requests, and R as the set of the next n retrieval requests. Finally, we define SL_i as the set of open locations available to receive a storage request i, and RL_j as the set of locations containing the item required by retrieval request j.

Thus, the Generalized Sequencing Algorithm – GSA is expressed as follows:

```
For $i$:=1 to $f$ do                                                              (0)
Select Storage Request with position $i$ ($S_i$) in queue                        (1)
   For each empty location $p$ in $SL_i$                                         (2)
     For each retrieval request $j$ remaining in $R$                              (3)
       For each candidate location $q$ in $RL_j$                                  (4)
         Select quadruplet $C$ :($S_i$, $p^* \in SL_i, j^*, q^* \in RL_j$) which minimizes $D_{sr}$   (5)
         Send cycle to the crane and update the data:
           $R = R \setminus \{j^*\}$                                              (6)
           $SL_i = SL_i \setminus \{p^*\} \; \forall \; i$                        (7)
           $SL_i = SL_i \cup \{q^*\} \; \forall \; i$ such that $i$ can be stored in $q^*$   (8)
Next $i$
```

In the execution of GSA, f dual-command cycles are formed (line 0) in order to minimize D_{sr} using a greedy approach. For each storage request that is considered in the ordered set S (1), the algorithm seeks the best cycle to form, using the remaining available and compatible locations (2), the remaining retrieval requests in the block (3), and their associated locations (4). Once a cycle is known to be the best move for a particular storage request i, the information is passed on to the crane and the sets are updated accordingly. The update removes retrieval request j^* from R (6), removes available location p^* from each set SL in which it appears, and adds the new available location associated to retrieval location q^* to the set of available locations SL_i, such that i can be stored in q^*. Using the GSA approach, the traditional heuristics NN, SL, and TT can be adapted to be used in a dynamic sequencing situation. This approach was implemented in the simulation engine presented in Section 5.

4 An AS/RS sequencing mathematical model (AS/RS-SM)

This section presents an integer linear program model that minimizes the travel time required to perform the storage and retrieval requests within a given block. In order to hold true, this formulation requires that $|S| = |R|$ which means that the number of retrievals is equal to the number of storage requests to be executed. When this requirement is not met, fictitious requests can be added to the smaller set.

Before presenting the model, we review the notations already introduced and define some new sets that are needed to reduce the number of variables:

S : Ordered set of storage requests, $S = \{s_1, s_2, \ldots, s_m\}$ where s_i can correspond to a precise SKU,

SL_i : SL_i is the set of all open locations available for storage request s_i. If a class-based system is used, SL_i correspond to the open location in the class where the SKU associated with s_i can be stored.

PS_p: The set of storage requests that can be stored to open location p (implicitly it is the set of request – products – that can be stored in the class of location p). If we use a 1-class system, $PS_p = S$ for all p. Note that PS_p may be empty.

$\bar{S}$: The set of all open locations (without repetition), $\bar{S} = \bigcup_{i\in S} SL_i$

R : Set of retrieval requests, $R = \{r_1, r_2, \ldots, r_n\}$ where r_j corresponds to a precise SKU,

RL_j : RL_j is the set of all locations having the same SKU as r_j.

PR_q: The set of requests that can be retrieved from location q (the set of requests which correspond to the SKU stored in location q).

$\bar{R}$: The set of all locations where a request can be picked (without repetition), $\bar{R} = \bigcup_{j\in R} RL_j$

t_{ipjq} : The travel-time required to store s_i in a location $p \in SL_i$ and retrieve r_j in a location $q \in RL_j$, while starting and ending at the I/O point. Note that $t_{ipjq} = M$, a sufficiently large value if either s_i or r_j are fictitious requests.

The decision variables are:

x_{ipjq}: Binary variable equal to 1 if the crane performs a double cycle linking storage request s_i performed in location $p \in SL_i$ with retrieval request r_j in location $q \in RL_j$, 0 otherwise.

AS/RS SM model can be expressed as follows:

$$Min \sum_{i \in S} \sum_{p \in SL_i} \sum_{j \in R} \sum_{q \in RL_j} t_{ipjq}\, x_{ipjq} \qquad (1)$$

Subject to:

$$\sum_{p \in SL_i} \sum_{j \in R} \sum_{q \in RL_j} x_{ipjq} = 1, \forall\, i \in S \qquad (2)$$

$$\sum_{i \in S} \sum_{p \in SL_i} \sum_{q \in RL_j} x_{ipjq} = 1, \forall\, j \in R \qquad (3)$$

$$\sum_{i \in PS_p} \sum_{j \in R} \sum_{q \in RL_j} x_{ipjq} \leq 1, \forall\, p \in \bar{S} \qquad (4)$$

$$\sum_{i \in S} \sum_{p \in SL_i} \sum_{j \in PR_q} x_{ipjq} \leq 1, \forall\, q \in \bar{R} \qquad (5)$$

$$x_{ipjq} \in [0,1] \qquad (6)$$

The objective function (1) minimizes the total travel time required to perform all the dual-command cycles, which may include fictitious requests. Constraint (2) insures that each storage request is executed in the right storage zone and is paired with a single retrieval. Conversely, constraint (3) makes sure that each retrieval request is associated to a single storage request and is executed in a location with the corresponding SKU. Constraint (4) insures that each available location is used for at most one storage request and insures that the stored product is compatible with the zone where this available location is located. Constraint (5) guarantees that each location with a requested product can be visited one time, at most. Constraint (6) makes the decision variables binary.

In this formulation, we assume that the system can hold one or more zones, with each zone being able to receive pallets from a subset of the products. For this reason, we need to define SL_i and RL_j, which groups all candidate locations for storage or retrieval requests, respectively. By candidate location, we mean an available location in the appropriate zone for a storage request and a location containing a specific product to retrieve. In addition to helping with zone configuration management, these subsets also help to deal with storage constraints found in many AS/RS in which some product incompatibilities exist, for example. These sets also help to reduce the number of variables, keeping only those that are useful. We assume that each retrieval request given in the sequencing horizon is feasible, which means that the products are available in the quantities needed to execute the whole block.

The main limitation of the model is its static nature, implying that each location can be targeted for a single operation. For instance, in the same block, it is not possible to execute a retrieval request in a location in which a product has been stored. This limitation may be overcome by using a dynamic programming approach (Lee and Schaeffer, 1996).

5. Simulation model

In order to evaluate the performance of the various sequencing methods, we developed a three-phase discrete-event simulation model, which was implemented in Visual Basic.Net. A simulation model is a great tool for analyzing AS/RS because of its dynamic and stochastic nature (Gagliardi et al., forthcoming, a). The model, which was designed in a generic mindset, allowed us to integrate the sequencing methods under study and test their performance on many AS/RS configurations, varying from the theoretical ones described in the literature to more realistic configurations, seen in modern distribution centers. (Interested readers can find more details about the model in Gagliardi et al., forthcoming, c.)

The model was inspired by our observations at one of our partner's distribution centers, which uses an AS/RS to manage orders placed by large-volume clients. These observations were used to build the model, as well as validating it. Without loss of generality, the simulation model can be summarized by Figure 1. Figure 1 shows the crane's possible movements. This movement logic follows a hybrid-command approach (Eben-Chaime and Pliskin, 1996), meaning that when the crane is at the I/O point, it can execute a storage request or a retrieval request in *single-command* (SC) cycle or execute a *dual-command* (DC) cycle if the number of available requests allows it. A complete description of the event logic can be found in Gagliardi et al. (forthcoming, c).

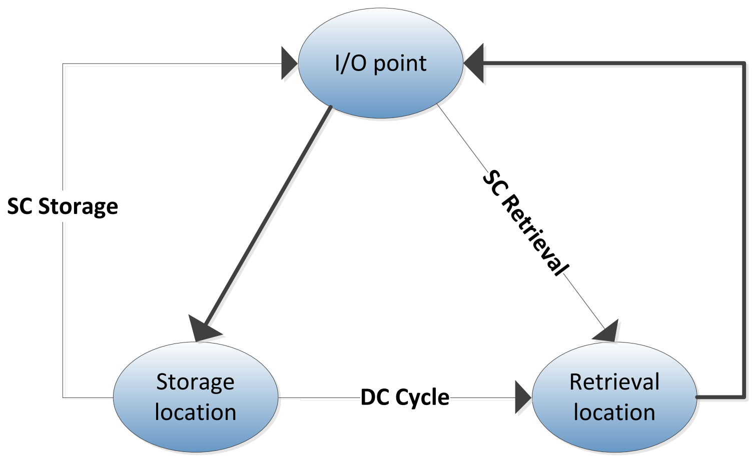

Figure 1: Possible movements for a single, aisle-captive crane

Our three-phase simulation algorithm works as follows. After an initialization phase, in which the events in the list are sorted in chronological order, the algorithm enters Phase A. In Phase A, the algorithm seeks the first event in the list, and the system clock is set to the execution time of the current event. In Phase B, the current event is executed: changes are applied to the system state and/or new events are added to the event list. In Phase C, the simulation engine verifies whether or not the system state allows conditional events to be executed. After executing each A-B-C loop, the algorithm verifies the stopping conditions. In our case, the stopping condition is based on the number of DC cycles performed after a certain warm-up period. Interested readers can consult Gagliardi et al. (forthcoming, c), where they will find an execution example.

For experimentation purposes, the simulation engine allows the use of *common random numbers* (CRN) in order to reduce variance between alternatives. Applying CRN in our model involves using a dedicated stream of random variables for each stochastic process in order to replicate the same experimental conditions for all alternatives. Knowing that the differences in the results are mostly due to their underlying parameters, using this CRN approach allowed the model to produce results that can be compared.

6. Experimental design and computational results

In past researches, AS/RS performance proved to be very sensitive to both design and control decisions. Specifying the underlying assumptions is a very important thing in an AS/RS study. In some cases, it can be shown that the conclusions from past researches were not verified under realistic conditions. In order to proceed with the experimental phase of this study, we first define the main assumptions used in the simulation model:

1. The system is a single two-sided aisle unit-load AS/RS.
2. A single shuttle moves simultaneously on both axes (i.e., Chebyshev metric).
3. Crane acceleration and deceleration is not considered.

4. When a unit-load is retrieved from the system, a storage request is created and queued for storage after a deterministic period φ, which is short enough so that DC cycles can always be formed.
5. The system has a single input/output (I/O) point located at the bottom front of the rack.
6. The storage rack is initialized at full capacity. When a zone-based storage is selected, the fastest-moving product is stored in the nearest zone from the I/O point.
7. Storage space is divided among all products according to their demand.
8. All locations are identical and are able to store any SKU.
9. The time required by the shuttle to pick up or deposit a unit-load is constant.

Based on these assumptions, we ran two sets of experiments. The first set of experiments aimed to validate our model by comparing its results with the ones in the literature. The second set of experiments aimed to evaluate the relative performance of sequencing approaches with crane travel time and with throughput. Each sequencing method was executed with our simulation model, according to the same parameters. For the sake of simplicity, we divide these parameters in two categories: simulation parameters (Table 1) and system configuration.

Table 1: Simulation parameters

Parameter name	Value
Number of replications (runs)	5
Number of single-cycles to complete (warm-up)	120
Number of dual-cycles to complete (stopping criteria)	1 200

After running several exploratory tests, we chose the simulation parameters values of Table 1 according to our knowledge about the system. As seen later in the computational results, the selected values for the number of runs and stopping criteria yield tight confidence intervals for the mean response, which are satisfactory (less than 1.3 %). Thus, it would not be justified to execute more or longer runs. As for the warm-up, the value represents the number of retrieval requests to execute a SC cycle before starting only DC cycles and computing statistics. This warm-up serves two purposes: 1) filling the storage request queue, and 2) setting the stock level to 80% (i.e., 20% available locations for 600 locations) in order to obtain more flexibility for storage location selection.

The scenarios were also tested on the same system configuration. We defined the following configuration parameters. Aisles are 12 locations high and 25 locations long, thus the system contains (12 x 25 x 2 = 600) locations, 300 on each side of the aisle. Each location is one cubic meter in dimension. The crane travels at speeds of 1 meter per second horizontally and 0.4 meter per second vertically. Using these values, we obtained a system that is rectangular in time and where $b = Min(t_h, t_v)/Max(t_h, t_v)$ = 25 / 30 = 0.83 (Bozer and White, 1984).

6.1. Experimental factors

The case in which LTPR=1 is required in order to validate our results against the results in the literature. When testing the case in which LTPR=1, we set the number of products to 600 so each product has only one location in the system. Arbitrarily, 150 products (25%) were generated to test the scenario in which LTPR >1 (i.e., LTPR=4). This value was attributed according to the real situation of our industrial partner.

In order to validate our results with the literature, it was necessary to include storage assignment parameters in this study. In order not to overload the results, we limited our focus to two extreme configurations (i.e., N-zone configuration, or full turnover based – FTB, and a 1-zone configuration). When LTPR=1, all items receive only one space; when LTPR >1, space was given according to their turnover distribution, approximated by $G(i) = i^s$ (Hausman et al., 1976). In order to compare our results with the results in the literature, we arbitrarily chose to set $s = 0.4$ for the case in which LTPR=1 and $s = 0.8$ for the case in which LTPR >1 (e.g., when $s = 0.4$, 20% of the fastest-moving products account for 52.5% of total demand, and when $s = 0.8$, the same products account for 27.5% of total demand).

Since this study is focused on storage and retrieval requests sequencing, 2 factors are tested: the sequencing horizon h, representing the number of retrieval requests to consider in each instance and the frozen horizon f, representing the number of cycles to execute before re-sequencing. Using these 2 factors, we are able to reproduce a policy where no sequencing is allowed ($h = 1, f = 1$), two of block sequencing policies ($h = f = 5$ and $h = f = 10$) and three dynamic sequencing policies ($h = 5, f = 1$; $h = 10, f = 1$ and $h = 10, f = 5$).

Since we are studying situations in which LTPR >1, the sequencing rules, summarized in Table 2, cannot be separated from the storage assignment policies. The presence of FCFS and Random sequencing rules are required to obtain upper bounds on average travel time, as well as an indication whether or not sequencing is profitable for a given configuration. When more than one location is available for an item, the FCFS rule employs the nearest-neighbor policy.

The following section summarizes our results for the configurations under study. These results represent the average total travel time in minutes required by the crane to perform 1200 DC cycles, according to the specific sequencing rules/storage assignment policies. Each table gives the average total travel time in minutes over 5 runs (line *Average*). In addition, a 95% half-width confidence interval is provided (line *HW*).

Table 2: Summary of experimental factors and levels

Factor	Levels
Sequencing horizon	*1, 5, 10 requests*
Frozen horizon	*1, 5, 10 requests*

Sequencing rules	Storage assignment policies
First-Come, First-Served (FCFS)	Random (RAN)
First-Come, First-Served (FCFS)	Closest-Open-Location (COL)
Random (RAN)	Random (RAN)
Random (RAN)	Closest-Open-Location (COL)
Nearest-neighbor (NN)	Random (RAN)
Nearest-neighbor (NN)	Closest-Open-Location (COL)
Shortest-leg (SL)	
Total Travel Time (TT)	
Sequencing Model (SM)	

6.2. Case with 600 products and 600 locations

Tables 3 and 4 focus on a system in which each product has a single location. In this case, LTPR=1. This configuration is studied because, to the best of our knowledge, the literature provides comparable results only for this case. Consequently, this example serves as a validation as well.

Configuration 1 (600 products, 600 locations; FTB, 20% free space, s=0.4) in Table 3 is very interesting because it allows us to isolate the impact of sequencing rules on the throughput. Since each item has a single dedicated location, the only decision to make is to select one or more retrieval requests to assign to the next set of DC cycles to perform.

In the most restrictive scenario, when $h = f = 1$, the system follows a strict FCFS rule, and it is almost impossible to identify a dominant method. When $f < h$, NN and SL show promising results, giving better results than our mathematical model. SL yields results that are rarely the best but almost always in the top two. This observation follows logically from the results of Han et al. (1987). For this configuration, the results clearly indicate the benefits of using a sequencing approach. For example, the shortest average total travel time found is obtained by following a NN policy with h=10 and f=1. When compared to the Ran/Ran case (i.e., using no sequencing at all), an expected 23% decrease in travel time is obtained.

Under the configuration of Table 3, the AS/RS-SM model yields longer average travel times than NN and SL. For example, since the model sequences the next h cycles and only the first f are performed, it could be profitable to perform the longest cycles first in order to obtain a better local optima. Nevertheless, the $h - f$ best moves could never be performed since a re-sequencing will be executed with new information. When $h = f$, the model gives the best results. This performance is accentuated by an increase in the length of the sequencing horizon. TT is always dominated by the other heuristics, with average travel times approximately 9.94% higher than the top approach.

Table 3: Average total travel time (minutes) to perform 1200 DC cycles for configuration 1

(600 products, 600 locations (LTPR = 1); FTB, 20% free space, $s = 0.4$)

Sequencing Horizon		**1**	**5**		**10**		
Frozen Horizon		**1**	**1**	**5**	**1**	**5**	**10**
FCFS / RAN	Average	811.3	811.5	806.04	813	812.51	808.76
	HW	9.89	9.12	4.19	3.55	2.23	6.08
FCFS / COL	Average	807.31	819.1	813.76	814.48	817.3	812.54
	HW	4.37	11.56	11.44	6.12	7.83	5.46
RAN/ RAN	Average	811.45	806.57	820.23	825.82	820.04	809.99
	HW	9.17	8.27	8.43	1.42	4.79	4.69
Ran / COL	Average	809.6	811.42	819.54	808.86	804.34	813.45
	HW	10.23	4.38	4.44	2.3	9.38	8
NN / Ran	Average	818.29	685.36	742.34	644.05	667.03	710.44
	HW	6.71	4.5	3.16	3.17	5.23	6.19
NN / COL	Average	**807.05**	**680.13**	744.08	**639.22**	667.31	714.13
	HW	4.82	6.62	4.1	5.9	7.48	3.02
Shortest Leg	Average	817.47	684.11	741.35	641.27	**656.2**	719.02
	HW	8.49	5.39	6.47	4.86	6.92	3.68
Total Travel	Average	810.22	753.4	784.22	713.69	725.95	753.35
	HW	7.67	6.7	5.7	5.38	6.14	6.3
SM	Average	811.35	695.79	**729.79**	672.37	671.66	**689.81**
	HW	5.78	6.47	4.05	8.39	3.4	6.93

For configuration 2 (600 products, 600 locations; 1 Zone, 20% free space, s=0.4) in Table 4, we induce some slack in the system by using a single zone policy (random storage). In this case, a unit-load could be stored at any available location. Using this configuration, the system now has the latitude not only to define the sequence of operations, but also to select the storage location for an item.

According to these results, using 1 zone generates an increase in average travel time compared to the FTB policy when LTPR=1. This observation follows logically from Hausman et al. (1976) and Gagliardi et al. (forthcoming, b), implying that it is better to concentrate fast-moving products near the I/O point. For five of these six scenarios, the dominant method is the sequencing model. In this case, since the system is 80% full, it is better to concentrate available locations toward the back of the rack. As mentioned by Han et al. (1987), this is a characteristic of the *shortest-leg* heuristic, which explains why the method still remains in the top two for this configuration. This configuration also favors TT, which also concentrates operations near the I/O point. We found that NN does not perform well for this configuration.

Table 4: Average total travel time (minutes) to perform 1 200 DC cycles for configuration 2

(600 products, 600 locations; 1 Zone, 20% free space, $s = 0.4$)

Sequencing Horizon		**1**	**5**		**10**		
Frozen Horizon		**1**	**1**	**5**	**1**	**5**	**10**
FCFS / RAN	Average	966.96	974.76	972.49	976.22	969.81	968.79
	HW	9.62	6.51	10.73	5.21	5.26	6.21
FCFS / COL	Average	838.78	840.86	854.54	843.28	844.6	837.59
	HW	7.55	6.52	6.30	13.48	6.22	14.64
RAN/ RAN	Average	972.17	982.96	973.57	974.93	985.14	972.05
	HW	7.84	8.51	9.47	9.70	3.95	8.48
Ran / COL	Average	847.29	848.31	845.48	840.77	841.71	843.24
	HW	11.24	6.31	4.69	9.32	10.31	9.96
NN / Ran	Average	976.73	852.86	895.98	811.35	822.55	870.19
	HW	6.61	5.41	4.38	8.20	6.90	5.48
NN / COL	Average	836.82	742.46	766.67	707.66	732.1	738.59
	HW	14.67	4.61	7.42	4.89	6.32	4.85
Shortest Leg	Average	756.65	**701.18**	728.23	689.04	695.85	706.1
	HW	11.51	9.99	7.37	9.83	7.96	9.08
Total Travel	Average	757.69	735.48	727.73	706.15	710.36	708.33
	HW	6.13	12.86	11.03	4.94	11.34	11.14
SM	Average	**739.16**	712.94	**716.05**	**701.44**	**694.86**	**705.41**
	HW	9.64	8.17	5.34	6.74	7.67	5.90

6.3. Case with 150 products and 600 locations

For configurations 3 and 4, we reduce the number of items to 150 in order to obtain a system with a $LTPR = 4$. Using this value, each product can be found in more than one location. This situation is much closer to real-life AS/RS. In addition to defining the sequence of operations and selecting a storage location for an item within the appropriate zone, the system now has to choose, for each retrieval request, the most appropriate location to execute that request. For this reason, storage assignment policy takes more importance, combined with the sequencing rule. In Table 5 (150 products, 600 locations; FTB, 20% free space, s=0.8), TT yields the lowest total travel time in one of the six cases, while SL is the top method two times out of six. The best results are obtained by using the sequencing model, which is the top approach in 50% of the cases, mostly when pure block sequencing is employed.

Table 5: Average total travel time (minutes) to perform 1 200 DC cycles for configuration 3
(150 products, 600 locations; FTB, 20% free space, $s = 0.8$)

Sequencing Horizon		1	5		10		
Frozen Horizon		1	1	5	1	5	10
FCFS / RAN	Average	929.3	932.67	939.45	941.4	930.32	939.94
	HW	5.06	3.06	6.66	4.91	7.31	5.27
FCFS / COL	Average	933.73	937.45	931.81	943.26	935.37	935.62
	HW	6.04	8.65	4.12	6.13	4.77	3.72
RAN/ RAN	Average	982.47	977.58	977.8	981.94	982.48	979.02
	HW	6.01	4.02	5.67	8.55	4.62	5.11
Ran / COL	Average	984.64	988.87	984.24	990.78	987.23	979.41
	HW	4.22	5.80	3.38	4.74	4.10	7.25
NN / Ran	Average	932.55	832.05	868.31	801.25	809.57	844.06
	HW	5.79	4.12	6.23	2.73	4.42	3.36
NN / COL	Average	937.18	832.75	875.02	800.59	802.2	841.16
	HW	4.87	7.02	4.91	5.46	1.76	4.28
Shortest Leg	Average	919.67	**821.18**	861.52	791.24	**797.59**	836.27
	HW	6.54	5.66	5.16	4.30	4.61	8.48
Total Travel	Average	**917.43**	881.95	891.98	870.33	868.37	865.87
	HW	5.67	6.97	3.75	5.95	5.43	4.25
SM	Average	921.66	821.32	**836.67**	**789.88**	800.38	**812.54**
	HW	1.86	9.49	6.88	7.07	4.45	8.69

Table 6: Average total travel time (minutes) to perform 1 200 DC cycles for configuration 4
(150 products, 600 locations; 1 Zone, 20% free space, $s = 0.8$)

Sequencing Horizon		1	5		10		
Frozen Horizon		1	1	5	1	5	10
FCFS / RAN	Average	840.63	842.55	839.5	851.25	841.62	847.26
	HW	10.98	5.61	5.98	2.53	6.50	5.64
FCFS / COL	Average	716.79	713.19	720.03	718.86	726.34	715.59
	HW	5.26	6.90	7.50	12.75	12.41	7.41
RAN/ RAN	Average	970.7	970.01	968.33	973.12	967.51	969.09
	HW	7.20	4.98	6.46	6.81	4.49	5.89
Ran / COL	Average	849.49	848.23	845.77	849.63	851.6	852.95
	HW	5.76	5.40	6.52	6.97	9.69	3.32
NN / Ran	Average	860.19	782.49	809.6	747.59	764.74	790.26
	HW	6.84	8.96	9.21	5.45	4.49	7.83
NN / COL	Average	719.7	667.14	697.06	628.52	636.79	670.18
	HW	6.72	8.29	8.22	15.84	11.76	4.71
Shortest Leg	Average	685.66	625.41	659.03	595.47	586.5	632.85
	HW	6.33	8.34	8.52	13.43	5.28	6.66
Total Travel	Average	584.93	553.44	558.75	**527.11**	527.02	540.54
	HW	4.86	5.11	6.89	5.67	6.14	6.06
SM	Average	**563.68**	**540.4**	**534.08**	548.51	**526.56**	**529.22**
	HW	5.25	4.71	9.25	7.84	8.36	3.86

The configuration of Table 6 (150 products, 600 locations; 1 Zone, 20% free space, s=0.8) reproduces the most realistic condition The results of Table 6 show that two methods now dominate the others. The sequencing model proves very efficient in five of the six cases for travel time reduction, followed by TT with an average gap of 2.8%. In addition, NN and SL, which proved efficient in some configurations, are dominated in 100% of the cases, implying that they may not be suited for more realistic conditions.

In summary, our empirical results show that applying one of the four best sequencing methods has a positive impact on crane efficiency. For the four cases under study, we observed that sequencing may yield between 7% and 44% decrease in average travel-time compared to the situation where no sequencing is applied. These results are interesting because they can be achieved without expensive changes to the system's design. Another observation is that the performance of the sequencing policies varies greatly from one configuration to the other. The overall ranking based on the average decrease in travel-time is the following: nearest neighbor (20.1%), total-travel (21.6%), shortest-leg (22.4%) and sequencing model (24.9%). These results show that the sequencing model performs very well. The latter reaching its top performance when more realistic configurations are considered.

7. Conclusions

In this paper, we focus on sequencing policies for unit-load AS/RS. We first introduce traditional heuristics presented in the literature and propose an adaptation in order to allow the case in which LTPR >1 is considered. We also introduce a mathematical formulation of the sequencing problem, the first one that considers the case in which LTPR >1. Since each item can be present in more than one pallet in the rack, the methods have to deal with storage assignment policies as well, making the sequencing problem richer. We provide an empirical study built to compare the sequencing methods. The study is based on a discrete-event simulation engine and allowed us to evaluate the performance of each method according to different configurations, from theoretical ones to more realistic ones based on a real-life AS/RS.

Based on these results, we conclude that sequencing has a measurable impact on AS/RS performance (up to 23%) and that this impact is function of the system configuration. When coupled with the storage assignment policy, an integrated approach can yield a reduction of up to 45% in travel time, which represent a significant gain in performance. We also note that increasing the sequencing horizon reduces expected travel time. Our results allows us to conclude that, even though the sequencing model does not represent the top approach for all configurations, it remains a robust model that takes advantage of certain characteristics of real-life configurations of the unit-load AS/RS. In complex configurations where each product has many locations and where random storage is used, the proposed model almost always outperforms other methods.

Acknowledgements

This research was partially financed by grants [OPG 0293307 and OPG 0172633] from the *Canadian Natural Sciences and Engineering Research Council* (NSERC). This support is gratefully acknowledged.

References

Bozer, Y. A., Schorn E. C. & Sharp G. P. (1990), Geometric approaches to solve the Chebychev traveling salesman problem. *IIE Transaction*, 22, 238-254.

Bozer, Y. A. and White, J. A. (1984). Travel-Time Models for Automated Storage/Retrieval Systems. *IIE Transactions.* Vol. 16, No. 4, pp. 329-338.

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

Fukurani, M. and Malmborg, C.J. (2008) A heuristic travel-time model for random storage systems using closest open location load dispatching. *International Journal of Production Research*, Vol. 46, No. 8, pp.2215-2228.

Gagliardi, J.-P., Renaud, J. & Ruiz, A. (a) Models for Automated Storage and Retrieval Systems : A Literature Review. *International Journal of Production Research*. In-Press. DOI: 10.1080/00207543.2011.633234.

Gagliardi J.-P., Renaud J. & Ruiz A. (b). On storage assignment policies for unit-load automated storage and retrieval systems. *International Journal of Production Research*. In-Press. DOI: 10.1080/00207543.2010.543939.

Gagliardi, J.-P., Renaud, J. & Ruiz, A. (c) A Simulation Modeling Framework For Multiple-Aisle Automated Storage and Retrieval Systems. *Journal of Intelligent Manufacturing*. In-Press, DOI: 10.1007/s10845-012-0686-x.

Graves S.C., Hausman W.H., & Schwarz L.B. (1977). Storage-retrieval interleaving in automatic warehousing systems. *Management Science*, 23, pp. 935-945.

Gu J., Goetschalckx M. & McGinnis L. F. (2010). Research on warehouse design and performance evaluation: A comprehensive review. *European Journal of Operational Research* , 203, pp. 539-549.

Han, M.-H.; McGinnis, L. F.; Shieh, J. S. and White, J. (1987). On sequencing retrievals in an automated storage / retrieval system. *IIE Transactions*, 19, 56-66.

Hausman W.H., Schwarz L.B., & Graves S.C. (1976). Optimal storage assignment in automatic warehousing systems. *Management Science*, 22, pp. 629-638.

Laporte G. (1992), The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59, 231-247.

Lee, H.F. & Schaefer, S.K. (1996). Retrieval sequencing for unit-load automated storage and retrieval systems with multiple openings. *International Journal of Production Research*, 34, 2943-2962.

Lee H. F. & Schaefer S. K. (1997), Sequencing methods for automated storage and retrieval systems with dedicated storage. *Computers and Industrial Engineering*, 32, 351-362.

Roodbergen K. J. & Vis I. F.A. (2009). A survey of literature on automated storage and retrieval systems. *European Journal of Operational Research*, 194, pp. 343-362.

Rouwenhorst B., Reuter B., Stockrahm V., Van Houtum G. J., Mantel R. J. & Zijm W. H. M. (2000). Warehouse design and control: Framework and literature review. *European Journal of Operational Research*, 122, pp. 515-533.

Van Den Berg, J. P. & Gademann, A. J. R. M. (2000). Simulation study of an automated storage/retrieval system. *International Journal of Production Research*. 38, 1339-1356.