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

Jean-Philippe Gagliardi[1,2], Jacques Renaud[1,2,*] and Angel Ruiz[1,2]

[1] *Faculté des sciences de l'administration, Laval University, Canada G1K 7P4*

[2]*Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT)*

[*]*Corresponding author at Faculté des sciences de l'administration, Laval University, Canada, G1V 0A6*

Tel.: +1 418 656 7029; Fax: +1 418 656 2624

E-mail adresses: Jean-Philippe.Gagliardi@cirrelt.ca (J.-P. Gagliardi), Jacques.Renaud@fsa.ulaval.ca (J. Renaud), Angel.Ruiz@fsa.ulaval.ca (A. Ruiz)

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 it shows that request sequencing (i.e., planning the order in which storage and retrieval requests are performed) is of paramount importance in AS/RS performance. This paper reviews and adapts the most popular storage and sequencing policies to dynamic contexts, and then it proposes a "sequencing mathematical model" (SMM) to simultaneously solve the sequencing and storage location problems. Extensive computational results based on a thorough simulation experiment plan confirm that performing the requests in the right sequence can have a positive impact on AS/RS performance. Our results show that the proposed sequencing mathematical model regularly outperforms other methods. When used in a dynamic context, the proposed SMM may yield up to a 25% reduction in average travel-time compared to the situation where a no-sequencing method is applied.

Keywords: Automated storage and retrieval systems; sequencing; storage assignment; random storage; turnover-based storage; discrete-event simulation

1. Introduction

Automated Storage and Retrieval Systems (AS/RS) are widely used in distribution centers to improve the efficiency of storage 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, implementing an AS/RS requires a serious analysis during the initial design phase (Sarker & Babu 1995; Roodbergen and Vis, 2009), 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, which constitute the object of this article, 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 several 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.

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 either storage or 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). Most of the studies consider crane speed as a constant which means that acceleration and deceleration are negligible and can be ignored. Moreover, since the goal of this work is to compare the performance of different request sequencing approaches, the way in which the crane speed is modeled becomes irrelevant provided that the same model is used when testing all the sequencing approaches. This being said, 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. (2012b) confirm these results through empirical simulations.

Gagliardi et al. (2012b) also demonstrate that these results are no longer valid as soon as 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 pure random storage assignment 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.

Let us assume that 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 (empty) locations evolve with each storage and retrieval task. Thus, the problem consists of sequencing 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.

To the best of our knowledge, there are only very limited results for request sequencing in AS/RS literature. 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 large or 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 defined as follows: (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 (NN, SL and TT) can be formalized 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 $f = h$, 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. The *generalized sequencing algorithm* (GSA) can easily handle a space allocation with multiple classes by a proper definition of SL_i.

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 (5), 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 SL and TT can be directly adapted to be used in a dynamic sequencing situation. As the NN heuristic minimizes only the interleaving time, it is not completely defined in the case where a SKU can be in multiple locations as the location where the storage will be done is not clearly specified. In our computational study, we will consider that the nearest neighbor heuristic is used with the closest-open-location storage assignment policy. In the following we will refer to GSA(NN), GSA(SL) and GSA(TT) as the generalized implementation of NN, SL and TT respectively.

4. An AS/RS sequencing mathematical model (AS/RS-SMM)

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 corresponds 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 SKU 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 (parameter) 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.

The AS/RS-SMM 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 selected 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 empty location is used for at most one storage request and insures that the stored product is compatible with the zone where this empty 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 help managing the zone configuration, 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 contribute reducing 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. When the sets are known, values of t_{ipjq} parameters are straightforward to compute. Let us consider the following small example to show how t_{ipjq} are evaluated. Let us assume that two products, A and B, need to be stored and that product A can be stored in locations 1 and 2, while product B can be stored in locations 3, 4 and 5. Two other products, C and D, need to be retrieved. C is currently available in locations 6 and 7 and product D is only in location 8. Thus we have S = {A, B}, SL_A = {1, 2}, SL_B = {3, 4, 5}, PS_1= {A}, PS_2= {A}, PS_3= {B}, PS_4= {B}, PS_5= {B}, R = {C, D}, RL_C = {6, 7}, RL_D = {8}, PR_6 = {C}, PR_7 = {C} and PR_8 = {D}.Traveling times parameters t_{ipjq} are : t_{A1C6}, t_{A1C7}, t_{A1D8}, t_{A2C6}, t_{A2C7}, t_{A2D8}, t_{B3C6}, t_{B3C7}, t_{B3D8}, t_{B4C6}, t_{B4C7}, t_{B4D8}, t_{B5C6}, t_{B5C7} and t_{B5D8}. Thus t_{A1C6} corresponds to the traveling time from the I/O to location 1, then to location 6 and returning to the I/O.

Even if this mathematical model solves the sequencing model for a complete request block, it can be executed in a way that reproduces the generalized heuristics with parameters f and h, using then the same sets of storage and retrieval. Thus, this mathematical model will be evaluated against GSA(NN), GSA(SL) and GSA(TT).

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 horizon, it is not possible to execute a retrieval request in a location in which a product has been stored.

5. Simulation tool

In order to evaluate the performance of the various sequencing methods, we developed a three-phase discrete-event simulation tool, which was implemented in Visual Basic.Net. Simulation is a great approach for analyzing AS/RS because of its dynamic and stochastic nature. The simulator, 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.

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.

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. A detailed description of the simulation tool and its validation can be found in Gagliardi et al. (2013).

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 studies. 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 speed is constant (acceleration and deceleration are 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. In the first set, the number of products is equal to the number of locations. This first set of experiments allowed us to validate our model by comparing the results produced to the ones available in the literature. In the second set, each product may have many locations, which made the sequencing more important. In both sets, experiments aimed to evaluate the relative performance of various sequencing approaches in terms of crane travel time. We tested the three generalized sequencing algorithms presented in Section 3.1, GSA(NN), GSA(SL) and GSA(TT), and the sequencing mathematical model (SMM). SMM formulations were solved using Cplex 12.5 with its default parameters. All these approaches were executed with our simulation model on a Intel core i7-3612QM, 2.1 GHz, 8 Gb RAM computer. Simulations were run using the same parameters, which were set at the values reported in Table 1 after several preliminary tests.

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 preliminary tests, we chose the simulation parameters values of Table 1 according to our knowledge about the system. As will be show 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 (half-width under 3.1% of the average for all the cases). 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). Our preliminary results showed that the rack shape does not impact the relative performance of the heuristics.

6.1. Experimental factors

For the case in which LTPR=1 (each SKU has only one location), we set the number of products to 600 so each product has only one location in the system. To test the scenario in which LTPR >1, 150 products (25%) were generated (i.e., LTPR=4). This value was set according to the real situation that was suggested by one of our industrial partners.

In order to be able to compare our results with the ones in the literature, it was also necessary to consider the storage assignment policy to be used. In order not to overload the reader with too much information, we report only results produced for two extreme configurations (i.e., *N*-zone configuration, or full turnover based – FTB, and a 1-zone configuration) and only one rack shape, as our preliminary experiments showed that this parameter does not impact the relative performance of the sequencing heuristics. 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 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, two factors were tested: the sequencing horizon *h*, representing the number of retrieval requests to consider when taking a sequencing decision, and the frozen horizon *f*, representing the number of cycles to execute before re-sequencing. Using these two factors, we are able to reproduce a policy where no

sequencing is allowed ($h = 1, f = 1$), two block sequencing policies ($h = f = 5$ and $h = f = 10$) and three dynamic sequencing policies ($h = 5, f = 1$; $h = 10, f = 1$ and $h = 10, f = 5$). We evaluated four different sequencing methods for each combination of values of *f* and *h*. The nearest neighbor, shortest leg and total travel heuristics within the GSA scheme, respectively GSA(NN), GSA(SL) and GSA(TT), and the sequencing mathematical model (SMM).

The following section summarizes our results. These results represent the average total travel time in minutes required by the crane to perform 1200 double cycles. 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*).

6.2. Case with 600 products and 600 locations

Tables 2 and 3 focus on a system in which each product has a single location (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.

In configuration 1 (600 products, 600 locations, 20% free space, s=0.4) space is allocated to product using the full-turnover-based approach. Table 2 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 $h = f = 1$, there are not sequencing decisions, and all the tested methods produce the same results as they consider only one storage and one retrieval at a time whose respective locations are predetermined. When $f < h$, GSA(NN) shows a good performance, producing the best results in three cases out of five. GSA(SL) never produced the best result, but almost always produced results extremely close to the best, as could be expected according to the results of Han et al. (1987). For this particular configuration, the results clearly demonstrate the benefits of using a dynamic sequencing approach. For example, the shortest average total travel time is obtained by a GSA(NN) policy with *f*=1 and *h*=10 (702.09). When compared to case where *f*=1 and *h*=1 (869.82), dynamic sequencing achieves nearly 20% decrease in travel time.

As can be seen in Table 2, the performance of the SMM model depends on the parameter *f*. In particular, SMM produces longer average travel times than GSA(NN) and GSA(SL) when *f* = 1

(except for $f = h = 1$) but improves them when $f = h = 5$ and $f = h = 10$. This behavior may be explained as follows. When $f = 1$, the model sequences the next h cycles but only the first one is performed. Then it could be profitable to perform the longest cycles first in order to obtain a better local optima over the h cycles. Nevertheless, the $h - f$ best moves could never be performed since a re-sequencing will be executed with new information. When $f = h$, the model gives the best results. This performance is accentuated by an increase in the length of the sequencing horizon. GSA(TT) is always dominated by the other heuristics.

Table 2: Average total travel time for configuration 1

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

Sequencing Horizon – h		**1**	**5**		**10**		
Frozen Horizon – f		**1**	**1**	**5**	**1**	**5**	**10**
GSA(NN)	Average	**869.82**	**741.82**	799.13	**702.09**	**719.77**	771.23
	HW	15.63	11.83	10.28	11.86	9.79	12.07
GSA(SL)	Average	**869.82**	742.6	799.81	703.59	721.32	770.23
	HW	15.63	12.58	10.74	9.77	11.32	9.67
GSA(TT)	Average	**869.82**	811.31	840.59	767.25	784.68	801.58
	HW	15.63	13.97	14.42	11.03	13.73	10.47
SMM	Average	**869.82**	757.11	**786.13**	727.22	735.39	**753.37**
	HW	15.63	11.33	10.29	10.53	11.50	10.22

Table 3 shows the numerical results for configuration 2 (600 products, 600 locations; 1 Zone, 20% free space, s=0.4). This configuration induces 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 choice not only to define the sequence of operations, but also to select the storage location for an item. For two out of six scenarios, the dominant method is the proposed sequencing mathematical model SMM. The GSA(SL) method still remains in the top two for this configuration.

Table 3: Average total travel time 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**
GSA(NN)	Average	902.69	801.63	841.57	775.81	797.05	809.28
	HW	17.27	14.25	9.23	14.59	8.34	17.29
GSA(SL)	Average	824.70	782.63	**782.12**	**752.64**	767.62	772.33
	HW	9.02	10.54	6.99	7.21	17.80	15.54
GSA(TT)	Average	824.71	793.78	791.79	769.77	**761.75**	**766.67**
	HW	9.02	8.82	16.86	9.69	10.97	15.40
SMM	Average	**810.01**	**774.32**	786.10	759.53	766.48	775.57
	HW	9.08	14.99	13.73	11.95	8.14	10.60

6.3. Case with 150 products and 600 locations

In configurations 3 and 4, we reduced 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 then much closer to real-life contexts. 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, the storage assignment policy becomes more important, as well as the sequencing rule. In Table 4 (150 products, 600 locations, FTB, 20% free space, s=0.8), both GSA(TT) and GSA(NN) yield the lowest total travel time for only one case, the best results are obtained by using the sequencing mathematical model SMM, which is the top approach in four of the six cases, mostly when pure block sequencing is employed.

Table 4: Average total travel time 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**
GSA(NN)	Average	1023.14	912.29	955.78	**875.09**	889.75	927.42
	HW	7.92	6.57	6.71	8.33	4.90	6.97
GSA(SL)	Average	1013.24	906.85	949.69	874.97	884.79	921.44
	HW	8.20	6.33	6.46	6.43	4.80	7.48
GSA(TT)	Average	**1012.97**	958.93	985.46	944.63	944.6	954.66
	HW	8.04	10.11	6.56	7.71	5.04	7.01
SMM	Average	1015.46	**906.29**	**929.71**	879.40	**884.02**	**896.33**
	HW	8.40	6.77	7.64	6.36	6.39	6.90

Table 5 reports the results for configuration 4 (150 products, 600 locations; 1 Zone, 20% free space, s=0.8) which reproduces the most realistic setting. The results of Table 5 show that two methods now dominate the others. The proposed sequencing mathematical model SMM proves to be very efficient, producing the best results in four out of six cases, followed by GSA(TT) which produced two best results. In addition, GSA(NN) and GSA(SL), which proved a good performance in some previous configurations, are now dominated in 100% of the cases, implying that they may not be suited for these more realistic conditions. If we compare the reference case (GSA(NN) with $f = 1$ and $h = 1$) to the results produced by the mathematical model SMM, we observe that a total distance reduction of up to 25.3% (SMM with $f = 5$, $h = 10$) may be achieved.

Table 5: Average total travel time 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**
GSA(NN)	Average	795.13	727.30	771.47	687.25	703.33	749.95
	HW	7.98	9.01	6.41	21.37	10.75	5.79
GSA(SL)	Average	761.48	706.85	717.95	674.06	684.07	702.82
	HW	4.51	17.37	7.55	3.99	17.45	10.31
GSA(TT)	Average	661.44	617.72	629.24	**589.53**	**589.30**	610.35
	HW	6.27	8.26	8.50	9.49	11.48	7.02
SMM	Average	**637.59**	**611.75**	**603.41**	609.27	594.88	**599.10**
	HW	7.33	6.23	7.37	10.90	8.29	1.72

We did not present the required computing times because they were generally negligible. For configuration 4, which was the one requiring the longer computational times, GSA(NN) required in average 0.00048, 0.002 and 0.01 second per cycle for sequencing horizons equal to 1, 5 and 10 respectively. GSA(SL) and GSA(TT) took almost the same computing time with 0.01, 0.05 and 0.09 second per cycle for sequencing horizons equal to 1, 5 and 10 respectively. Finally, SMM required the longest computational times with 0.03, 0.34, and 1.54 second per cycle for sequencing horizons equal to 1, 5 and 10 respectively. These times correspond to the time needed by Cplex to solve each SMM problem to optimality. Clearly, in all cases, computing time is negligible compared to the time needed by a crane to perform a cycle and thus all these approaches can be applied in industrial settings.

In summary, our empirical results show that applying a sequencing method has a positive impact on crane efficiency. For the four cases under study, we observed that sequencing may yield up to

a 25% reduction in average travel-time compared to the situation where a basic sequencing method is applied – GSA(NN). These results are interesting because they can be achieved without expensive changes to the system's design. These results also show that the sequencing mathematical model performs very well, reaching its best 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. To the best of our knowledge, this is the first formulation that considers the case in which LTPR >1 as well as any zone configuration. 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 an important impact on AS/RS performance and that this impact is function of the system configuration. Our experiments on real-life inspired instances demonstrate that, when coupled with the appropriated storage assignment policy, a sequencing approach can yield a reduction of up to 25.2% with respect to the total travel time produced when sequencing is neglected, 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 mathematical model does not represent the best approach for all the 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 mathematical model outperforms other methods almost always.

Acknowledgements – This research was partially supported by Grants OPG 0293307 and OPG 0172633 from the Canadian Natural Sciences and Engineering Research Council (NSERC). This support is gratefully acknowledged. We also express our gratitude to the referees for their valuable comments and suggestions.

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,* 16, 4, 329-338.

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*, 46, 8, 2215-2228.

Gagliardi, J.-P., Renaud, J. & Ruiz, A. (2012a). Models for Automated Storage and Retrieval Systems : A Literature Review. *International Journal of Production Research*. 50, 7110-7125.

Gagliardi J.-P., Renaud J. & Ruiz A. (2012b). On storage assignment policies for unit-load automated storage and retrieval systems. *International Journal of Production Research*, 50, 879-892.

Gagliardi J.-P., Renaud J. & Ruiz A. (2013). A simulation modeling framework for multiple-aisle automated storage and retrieval systems. To appear in *Journal of Intelligent Manufacturing*. 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, 935-945.

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, 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.

Sarker B. R. & Babu P. S. (1995). Travel time models in automated storage/retrieval systems: A critical review. *International Journal of Production Economics*, 40, 173-184.

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.