CONTROL POLICIES FOR A DYNAMIC STORAGE SYSTEM WITH MULTIPLE LIFTS AND SHUTTLES

Iris F.A. Vis
VU University Amsterdam, Faculty of Economics and Business Administration, De Boelelaan 1105, Room 3A-31, 1081 HV Amsterdam, The Netherlands

Hector J. Carlo
Paloma Diaz
Maria E. Laboy
Industrial Engineering Department, University of Puerto Rico -- Mayagüez, Call Box 9000, Mayagüez, PR 00681

Bruno van Wijgaarden
Vanderlande Industries Nederland BV Vanderlandelaan 2, 5466RB Veghel, The Netherlands

Abstract

New types of Automated Storage and Retrieval Systems, able to achieve high throughput levels, are continuously being developed and require new control polices to take full advantage of the developed system. In this paper we study a dynamic storage system as developed by Vanderlande Industries consisting of a conveyor, two lifts, multiple transfer shuttles, and a storage rack. One of the decision problems for this system is the scheduling problem of the two lifts. In other words, which lift is going to handle which request and in which order. In this paper, we derive an integrated look-ahead heuristic based on enumeration to simultaneously assign a set of pre-defined requests to the lifts and to schedule the lifts. As main performance measure we use the total time required to serve all requests.

1. Introduction

In warehouses and production environments automated storage and retrieval systems have been widely used and introduced since their introduction in the 1950s. As a fully automated system, an automated storage and retrieval system (AS/RS) is capable of handling pallets without interference of an operator. Cranes running through aisles in the system to store and retrieve pallets from racks. Implementing AS/RSs instead of non-automated systems results in savings in labor costs and floor space, increased reliability and reduced error rates. Apparent disadvantages are high investments costs, less flexibility and higher investments in control systems ([1]). The most basic version of an AS/RS has in each aisle one crane, which cannot leave its designated aisle (aisle-captive) and which can transport only one unit-load at a time (single shuttle). Product handling in this case is by unit-load (e.g., full pallet quantities) only; no people are involved to handle individual products. The racks in the basic version are stationary and single-deep, which means that every load is directly accessible by the crane. This AS/RS type is referred to as a single unit-load aisle-captive AS/RS. A large number of system options exist for AS/RSs. For an overview we refer to [2] and [3].

New designs of AS/RS are being introduced to the market to meet current demands in throughput and constraints with regard to delivery times in warehouses. In this paper, we study a dynamic storage system as developed by Vanderlande Industries consisting of a conveyor, two lifts, multiple transfer shuttles and a storage rack. The rack consists of multiple aisles where products can be stored. Each aisle comprises several levels of the storage rack, a transfer shuttle and two buffer positions located at the front of the rack: one position to offer storage requests to the shuttle and one position where the shuttle can offload its retrieval requests. The transfer shuttle moves requests between the rack locations and the buffer positions. The conveyor has two transfer points (I/O-points), one to deliver storage requests and one to pick up retrieval requests. The I/O points are preferably positioned at the same level as the buffer positions of one of the aisles in the storage rack. Two lifts share a mast while transporting requests from the aisle buffer positions to the conveyor I/O points and vice versa. The two lifts cannot pass each other, and as a result, the upper lift can only reach the I/O point if the lower lift is positioned at one of the aisles below the I/O point. Clearly, the upper lift can never reach the lowest aisle and the lower lift never can reach the uppermost aisle. Figure 1 shows the system.

In designing an AS/RS, many physical design and control issues have to be addressed in the right way to fully take advantage of all its pros. For an overview we refer to [2]. We consider two aspects to be important in the physical design namely the system choice and the system configuration (i.e., number of aisles or dimensions). We refer to [4] for a more elaborate overview of selection criteria for various AS/RS types. Control policies are methods which determine the actions performed by the AS/RS. We distinguish between storage assignment policies (i.e., which products are assigned to which locations), dwell-point policies (i.e., where to position an idle crane), sequencing rules (i.e., order and tour of requests) and batching policies (i.e., combining different orders in a single tour). One of the important decision problems for the system under study in this paper is the scheduling problem of the two lifts. In other words, which lift is going to handle which request and in which order. In literature various algorithms and heuristics are available to schedule storage and retrieval requests within a fixed period of time [3]. The main objectives in those approaches are to minimize total travel times or total travel distances. However, there is not much research for AS/RSs with two or more lifts sharing the same path. We notice a close resemblance with the problem of sequencing two cooperating automated stacking cranes (ASCs) in the storage area in a container terminal. Vis and Carlo [5] study a configuration of two ASCs that can pass each other during operation and propose a sequencing approach to handle both inbound and outbound storage and retrieval requests.

The objective of this paper is to present an integrated look-ahead heuristic based on enumeration to simultaneously assign a set of pre-defined requests to the lifts and to schedule the lifts such that total times required to serve all requests are minimized. In Section 2 we present the problem in more detail. In Section 3 we introduce a conceptual model of the problem under study. Solution approaches will be introduced in Section 4. In Section 5 we present an illustrative example to demonstrate our method. Conclusions and further research issues are included in Section 6.

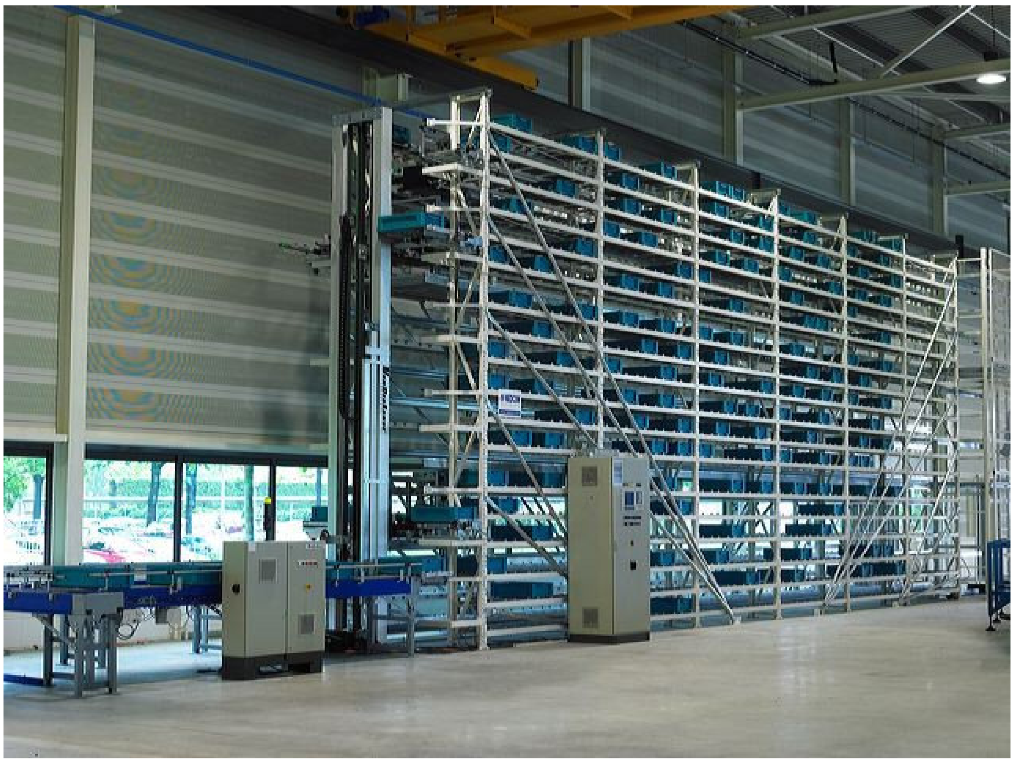
Figure 1: Dynamic storage system as developed by Vanderlande Industries (source: Vanderlande Industries)

2. Problem Description

Figure 2 gives a schematic representation of the dynamic storage system as designed by Vanderlande Industries. The main components of this type of AS/RS are:

- *a rack* consisting of *multiple levels* (*p*) to store products, where as the lower most level is level -1 and the highest level (p - 2)
- *a conveyor* to transfer loads between other departments in the facility and the storage system,
- *two lifts* (*L1* and *L2*) sharing a mast to transport the load along the rack to the appropriate level,
- *multiple transfer shuttles* N_x (x = -1, .., p-2) per level to actually store and retrieve the loads from the rack,
- *buffer areas* at the end of each level x (x = -1, .., p-2) where the lift can pick up or store a load, and
- *two I/O-points* at level 0 on the interface between the conveyor and the lifts, one to deliver and one to pick-up requests.

The lifts handle two types of requests. Storage requests that need to be stored in the system and retrieval requests that need to be picked as a response to customers' orders. For storage requests, we known the destination level k (k=-1,0,1,..p-2) and for each retrieval request we known the origin level k (k=-1,0,1,..p-2) for the lift. We denote storage requests at level k (k=-1,0,1,..p-2) with S_k and retrieval requests at level k (k=-1,0,1,..p-2) with R_k.

In this paper, we study the sequencing of the requests for both lifts operating in this dynamic storage system. Main constraint is that the two lifts cannot pass each other. Two important characteristics that restrict the lifts during operation result, namely:

1. The upper elevator can only reach the I/O point if the lower crane is positioned at one of the aisles below the I/O point.
2. The upper elevator never can reach the lowest aisle and the lower elevator never can reach the uppermost aisle.

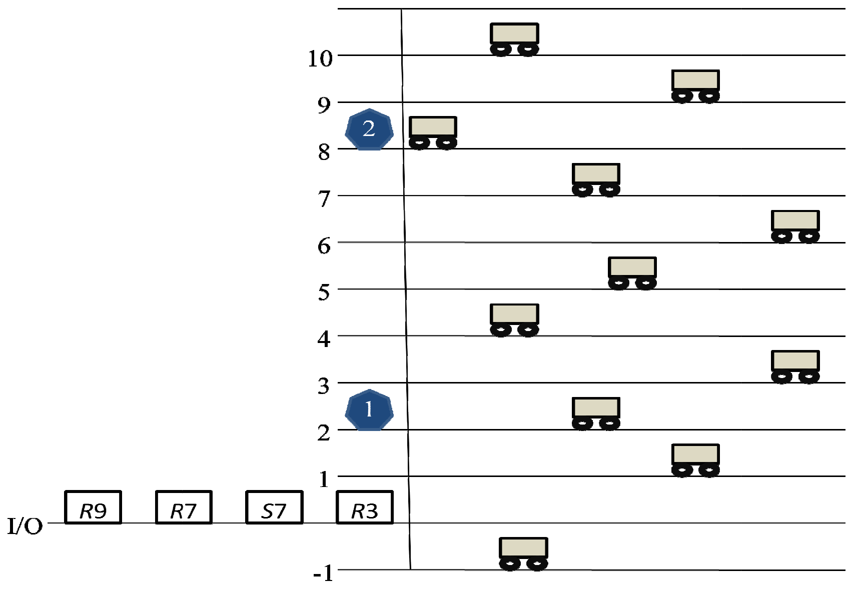

Figure 2: Schematic representation of the system under study.
In this example at level zero I/O points are positioned. Retrieval (R_k) and Storage (S_k) requests are waiting to be handled. Two elevators share a mast. The rack consists of 11 aisles with each a shuttle.

In sequencing both types of requests, we need to take into account some important characteristics of each type of requests. The sequence of storage requests depends on the order in which they arrive at the I/O point. The storage location for each request is selected beforehand and preferably in the same row where a retrieval request will be performed. As a result, double cycles can be performed similar to dual command scheduling for AS/RS systems [2]). Storage space is reserved for each order so that it is evenly distributed over the rack. The WMS makes the decision of where to store the request based on a short list with preferred storage locations. The sequence of the retrieval requests is known in advance. A lift is allowed to skip a request (i.e. give it to the other lift) as long as the requests are served in the order requested. Once a request is assigned to a lift, the assignment cannot be changed. We assume that a lift waits with a request if and only if the two lifts will collide when performing the desired move. If a lift is occupying part of the track, but they will not interfere, the lift will move as desired. In the next section, we introduce a conceptual model of the problem describing the movements of the lifts to perform requests in more detail and to introduce all relevant notation.

3. Conceptual Model

Basically, a lift performs four steps to handle a *storage request.* These steps can be described for a storage request S_k as follows (refer to Figure 2 for system characteristics):

1. move from current position to level 0
2. pick up item from I/O-point at level 0
3. move with load to level k
4. release item at buffer location at level k

Similar for a *retrieval request* R_k four steps need to be performed by a lift. Namely,

1. move from current position to level k
2. pick up item from buffer location at level k
3. move with load to level 0
4. release item at I/O-point at level 0

In case of performing double cycles, the same steps can be considered. As mentioned in section 2, storage requests are preferably stored at the origin level of the retrieval. In that case, step 1 related to a retrieval request, basically starts at the same level k where the storage request was being made, and the retrieval is collected. In between two actions, a lift might need to wait at a specific position to allow the other lift to move out of the way. As a result, total handling times consist of pick-up/deposit times, travel times and waiting times. In sequencing lifts we need to take into account their positions in time to be sure that no collisions occur and that waiting times will be minimized in assigning requests to lifts. Therefore, we need to keep track of the lift position in time. We introduce the following notation:

i	index associated with lift 1 to represent the specific step of a request being performed; i = 1, .., 5
j	index associated with lift 2 to represent the specific step of a request being performed; j = 1, .., 5

The values 1-4 for i and j are related to the steps described above. Only when one lift needs to move away to allow the other one to pass the value 5 is used to indicate the additional movement of a lift. Clearly, the type of movement and the state (*i.e.*, full or empty) of the lift related to a step (*e.g.*, i=3) differs per type of request (see above for definition of steps for each type of requests). Next to that, we define a (continuous) function to represent the position of a lift.

$f(t)$	Function that represents the position of lift 1 in time t
$g(t)$	Function that represents the position of lift 2 in time t

$f_i(t)$ continuous i^{th} function *f(t)* that represents the position of lift 1 in time *t* for a specific order

$g_j(t)$ continuous j^{th} function *g(t)* that represents the position of lift 2 in time *t* for a specific order

In Figures 3-4 we show by means of an example the various functions for respectively lift 1 handling a retrieval request and lift 2 handling a retrieval request. In Figure 3, the current position of lift 1 is at level 0. The lift moves empty from level 0 to level 3 (step 1). It takes a certain amount of time to pick up the request from the buffer location at level 3 which relates to step 2. Thereafter, the lift moves with the load to level 0 where it arrives at time 4. In the fourth step, the item is being released at I/O-point at level 0

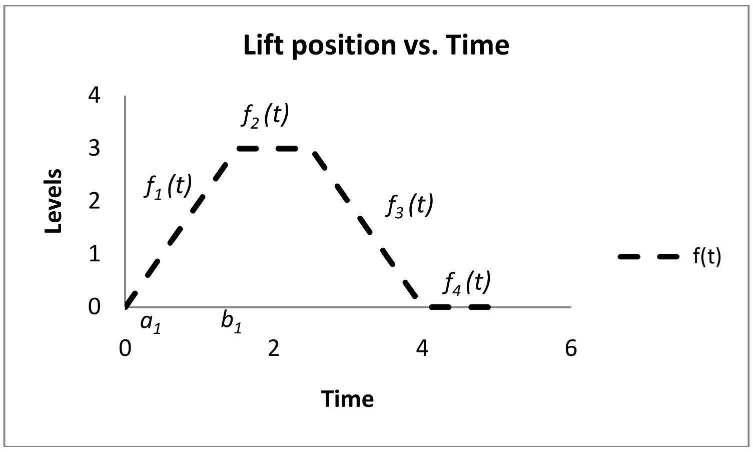

Figure 3: Example of a retrieval request handled by lift 1. The lift position in time is being represented.

In Figure 4, lift 2 starts at level 5 and travels empty to level 4 to pick up a retrieval request. In step 2 the retrieval request is derived from the buffer area. Thereafter, the lift moves in step 3 to the I/O point where it arrives a little bit after time equals three. Finally, in step 4, the load is transferred to the conveyor.

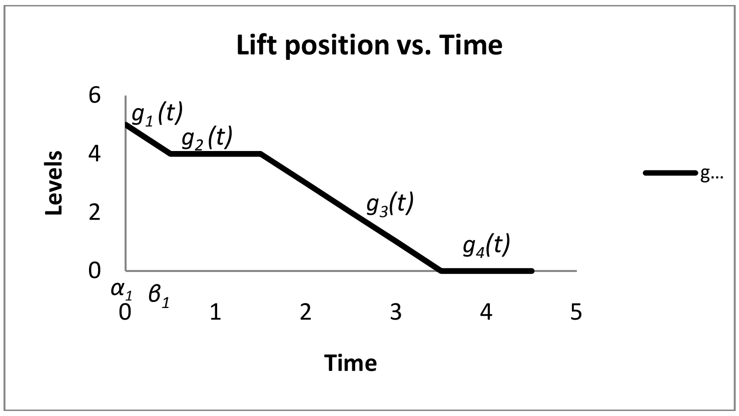

Figure 4: Example of a retrieval request handled by lift 2. The lift position in time is being represented.

In figure 3, a_1 and b_1 represent the first and last point, respectively, of the interval in which the $f_1(t)$ occurs. The parameters α_j and β_j represent the first and last point, respectively, of the interval in which the j^{th} function occurs, as shown in Figure 4 which shows a request being served by lift 2. In both figures, we assume that there is only one lift handling requests around the mast. Figure 5 depicts these functions where the two lifts handle their request simultaneously and share the mast. From this Figure it can be noticed that the constraint that the two lifts cannot pass each other is violated. So in sequencing requests, we need to carefully check that de situation depicted in Figure 5 is avoided. Lift 1 needs to travel to level -1 to allow lift 2 to reach the I/O point. Next to that, one of the lifts needs to move to avoid a collision at time 2.

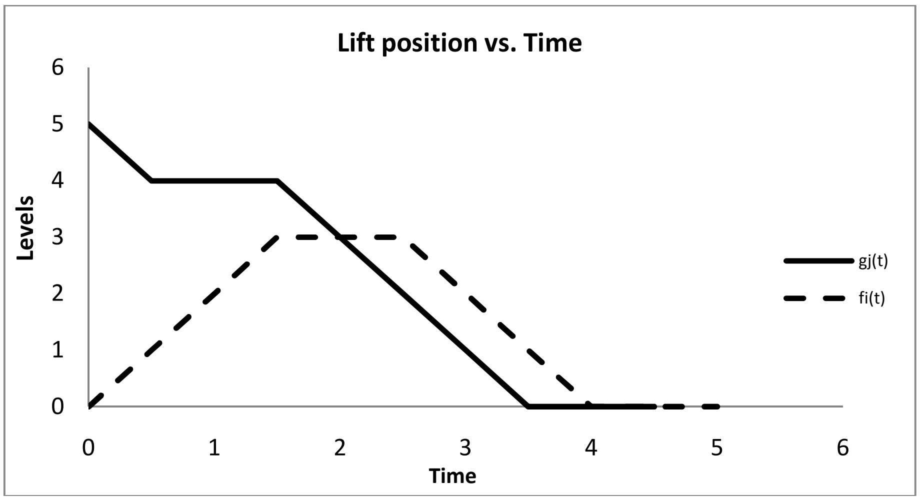

Figure 5: Example of lift 1 and lift 2 handling a retrieval request simultaneously (combining Figures 3 and 4). The lift positions in time of both lifts are being represented.

4. Heuristic

Our objective is to simultaneously assign requests to lifts and schedule the requests for each of the lift. As shown in Figure 5, the fact that the lifts share a mast makes that evaluating the fitness of a solution is not trivial. Next to that many solutions exist to sequence a set of requests and get a solution that is feasible. Our solution approach is based on exhaustive enumeration. We consider in Section 4.1 two options in defining a set of requests that has to be scheduled by proposing a look-ahead approach. For each set, several candidate solutions exist, that represent which lift performs what request and in what order. For each of these candidate solutions, we need to calculate the related total time to handle all requests by both lifts. In making these calculations we check if no collisions occur and if this is the case, we calculate the related delays and additional moves of lifts. In the final step, we select the candidate solution with the lowest total handling time. In Section 4.1 we show how we define a set of requests to be considered. In Section 4.2 we show how to derive total travel times for each candidate solution by effectively dealing with the constraints with regard to lifts sharing a mast as presented in Section 2.

4.1 Look-ahead strategy

Numerous options exist to divide all available storage and retrieval requests over both lifts. We deal with unit loads in the system under study and the capability of a lift to handle a single unit load at a time. Therefore, an important characteristic of handling a load is that the lift needs to visit the I/O point once per request or once per double cycle of a storage and retrieval request. Either at the start to pick-up a storage request or at the end to deliver a retrieval request. Next to that, due to the relation with other material handling systems in the facility there are restrictions on the order in which requests can be handled (see Section 2). As a result, we can consider a more dynamic approach and assign only a subset of the available requests to each lift. So we suggest, only considering the next available requests, resulting in a significant reduction of the number of possibilities to be considered.

We consider the following two options:

1. Only look at the next 2 requests
2. Only look at the next 3 requests

For each option we can perform an exhaustive enumeration procedure to solve the problem. Figures 6a and 6b show how the decisions would be made after considering all the possible solutions to serve the requests. In Section 4.2 we describe a method how to evaluate the performance of a candidate solution.

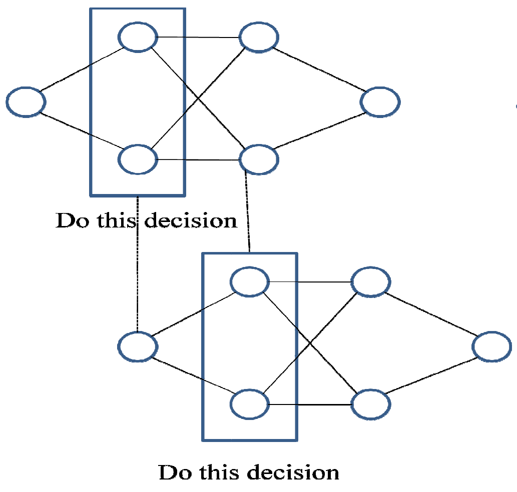

Figure 6a: looking at the next 2 requests

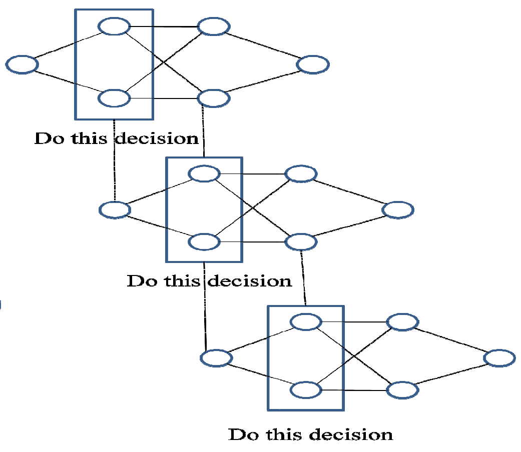

Figure 6b: looking at the next 3 requests

4.2 Evaluation performance of a solution

As introduced in Section 3, we consider four steps for each request to be performed by a lift. Related to these steps we introduced the functions $f_i(t)$ and $g_j(t)$ to respectively represent the positions of lift 1 and lift 2 in time t. Only if the lifts do not move in the same area, the four steps can be completed without interruptions. If one of the lifts has to be moved to allow the other one to pass, an additional step has to be performed. In that case either the value of i or j might change to 5 to indicate this.

We introduce the following solution approach to determine which lift performs what request and in which order by dealing with all relevant constraints. First we check if the two continuous functions representing the positions of both lifts in time are parallel. Basically, parallel can be defined as the two lines having the same slope. If the two lines are not parallel (see for example $f_1(t)$ and $g_1(t)$ in Figure 5), we check if the two segments of the line representing the actual movement of the lifts intersect. If this is not the case, we consider them as being parallel. If this is the case than there is a conflict that has to be solved. If they intersect, we need to move the appropriate lift. Our initial assumption is that we always give priority to lift 1 (see step 2b - ii(2b)). If lift 2 has been waiting for lift 1 to visit the I/O-point, lift 1 needs to move to level -1 to allow lift 2 to access level 0. Secondly, we assume that the lift already moves to its destination and waits at the neighboring level instead of waiting until the entire track is empty (see step 2b - ii(1)). Both options could easily be released in the heuristic proposed by making some small modifications. Simulation studies have to be performed to check which strategies are preferred for both decisions. A graphical representation of the heuristic is depicted in Figure 7.

```
0)  i := 1; j:= 1
1)  Determine $f_i(t)$ and $g_j(t)$
2)  Verify if they are parallel functions. If they are parallel,
    go to step 2a, otherwise go to step 2b.
    a)  Compare $b_i$ and $\beta_i$. If $b_i > \beta_j$, then $j = j+1$, if $b_i < \beta_j$
        then $i = i+1$. If $b_i = \beta_j$ $i = i+1$ and $j = j+1$
    b)  Verify if $f_i(t) = g_j(t)$ in $\{\max(a_i, \alpha_j), \min(b_i, \beta_j)\}$
        i)  if $f_i(t) \neq g_j(t)$, go to step 2a and consider them as
            being parallel.
        ii) else, verify $f_i(a_i)=f_i(b_i)$
            (1) If yes, then $g_j(\beta_j) = f_i(b_i) + 1$ and $i = i+1$
            (2) If no, verify $g_j(\alpha_i) = g_j(\beta_j)$
                (a) If yes, $f_i(b_i) = g_j(\beta_j) - 1$ and $j = j+1$
                (b) If no, $g_j(\beta_j) = f_i(b_i) + 1$ and $i = i+1$
Repeat Step 1
```

The lower part of Figure 7 studies if the lifts intersect at some point ($f_i(t) = g_j(t)$ in $\{\max(a_i, \alpha_j), \min(b_i, \beta_j)\}$). There are only three possible ways in which the lifts can intersect. Namely,

- lift 2 is going down and lift 1 is retrieving or releasing an item in a specific level that lift 2 need to pass.
- lift 1 is moving up and lift 2 is retrieving or releasing an item in a specific level that lift 1 need to pass.
- lift 2 is moving down and lift 1 is moving up.

Note that requests at the highest level always have to be assigned to lift 2 and at the lowest level to lift 1 (see Sections 2 and 3). If the *f(t)* and *g(t)*functions intercept each other then one of the lifts must move to allow the other lift to complete its request. The lift to be moved depends on the location and destination of each lift.

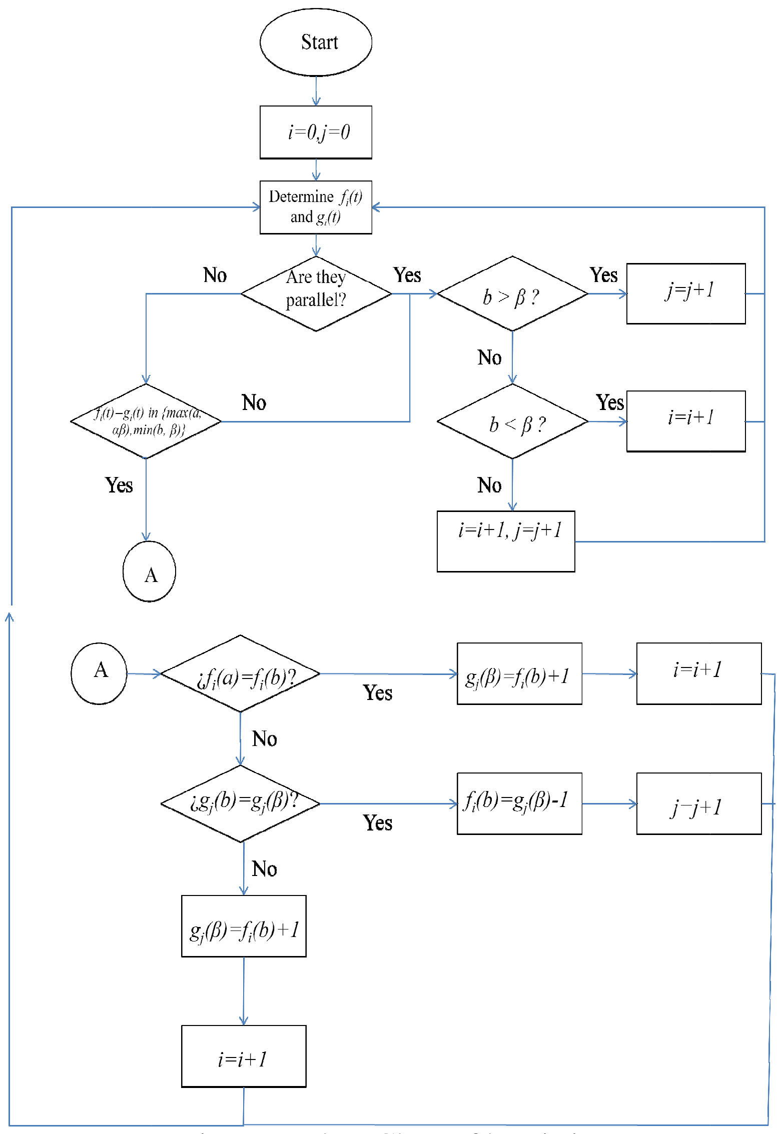

Figure 7: Flow Chart of heuristic

5. Illustrative Example

In this section we show how we can apply the method presented in Section 4.2 to evaluate the performance of a candidate solution. We use the situation represented in Figure 5 and show how the heuristic depicted in Figure 7 can be applied.

We start with $i := 1$ and $j := 1$. $f_1(t)$ and g_1(t) are depicted in Figure 8a and are not parallel lines. So, we proceed to step 2b. As can be seen in Figure 8a, $f_1(t) \neq g_1(t)$ in the interval $[0, \beta_1]$ and no conflicts will occur. We return to step 1 and consider the two lines as being parallel. We continue with $j=2$ and $i=1$ where as $b_1 > \beta_1$.

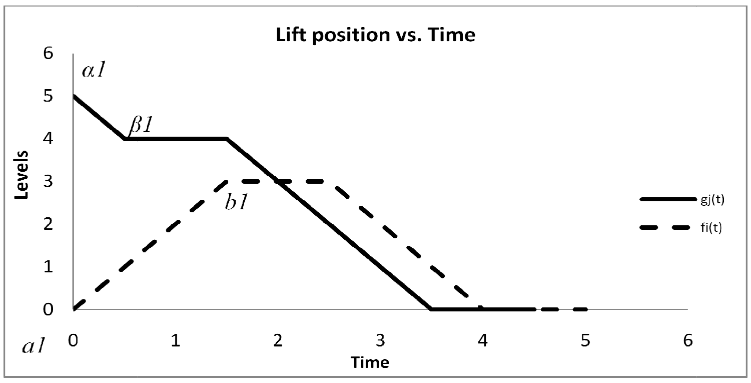

Figure 8a: Showing heuristic step-by-step

We now consider the lines $f_1(t)$ and $g_2(t)$. Those two lines are not parallel. In figure 8b we note that $f_1(t) \neq g_2(t)$ in the interval $[\alpha_2, b_1]$ and that b1= β_2. As can be concluded from step 2a, we continue with i =2 and j = 3.

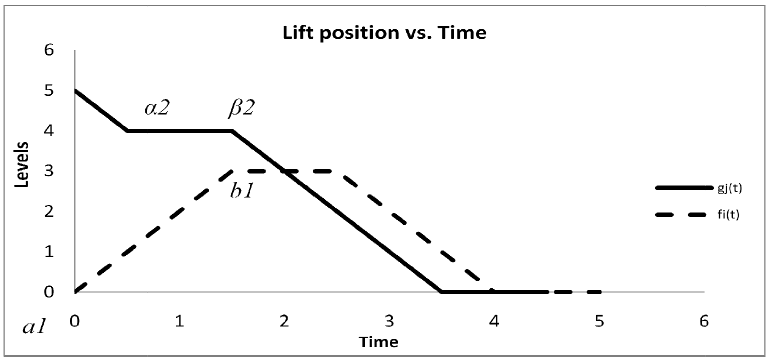

Figure 8b: Showing heuristic step-by-step

We now consider the lines $f_2(t)$ and $g_3(t)$. Those two lines are not parallel. In figure 8c we note that $f_2(t) = g_3(t)$ in the interval $[a_2, b_2]$. So, we continue with step ii). Lift 1 is handling an item at level 3 and not moving. So, $f_2(a_2) = f_2(b_2)$. Lift 2 is forced to wait at level 4 for lift 1 and is allowed to continue after lift 1 finishes handling the item. This situation is depicted in Figure 8c. We continue with i = 3 and j =3 .

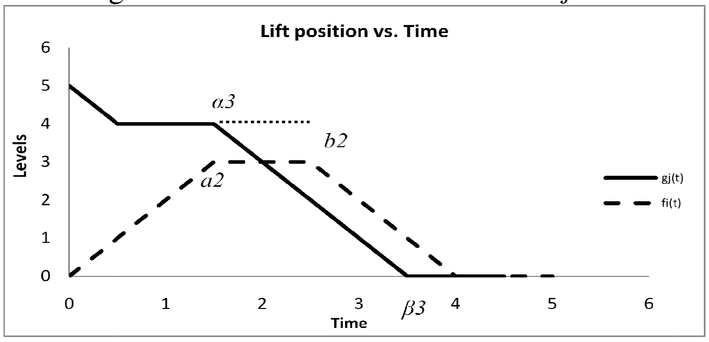

Figure 8c: Showing heuristic step-by-step

We now consider the lines $f_3(t)$ and $g_3(t)$. $g_3(t)$ is now represented by the dotted line in Figure 8d to show the time required to wait for lift 1. Those two lines are parallel and $b_3 < \beta_3$. So, we continue with $i = 4$ and $j = 3$.

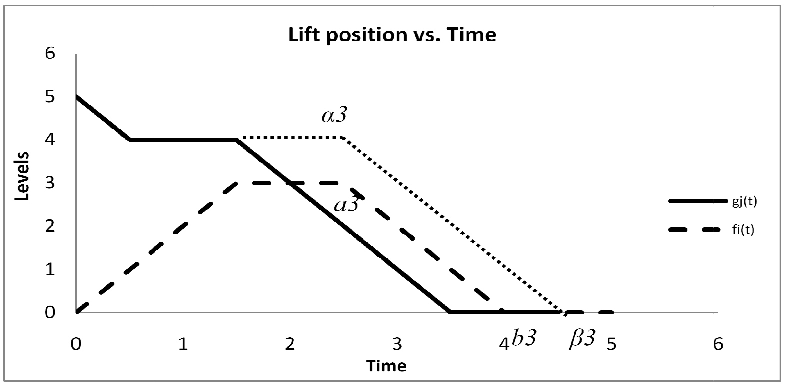

Figure 8d: Showing heuristic step-by-step

We now consider the lines $f_4(t)$ and $g_3(t)$. Both lines are not parallel and they will intersect between time 4 and 5 at the I/O point.

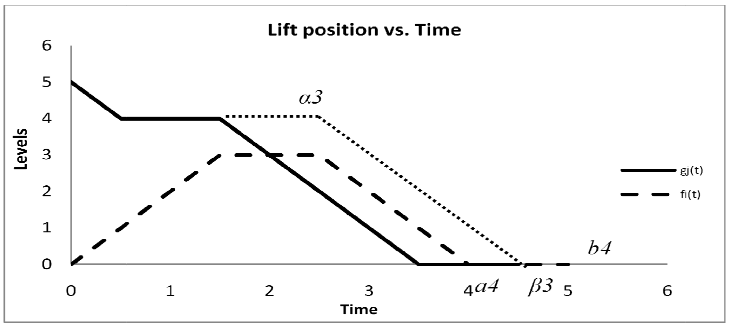

Figure 8e: Showing heuristic step-by-step

Lift 1 is handling an item at the I/O point and as a result lift 2 cannot arrive at the I/O point in the time interval [a_4, b_4]. As a result, step 2b-ii(1) of the heuristic indicates that lift 2 needs to move to level 1. Lift 2 can start moving to level 1, the moment lift 1 starts moving to level 0. To allow lift 2, to reach the I/O-point, lift 1 needs to move to level -1 after finishing it's job. Figure 9 presents the resulting solution for this candidate solution after following the heuristic to derive total times to handle items as presented in Section 4.2.

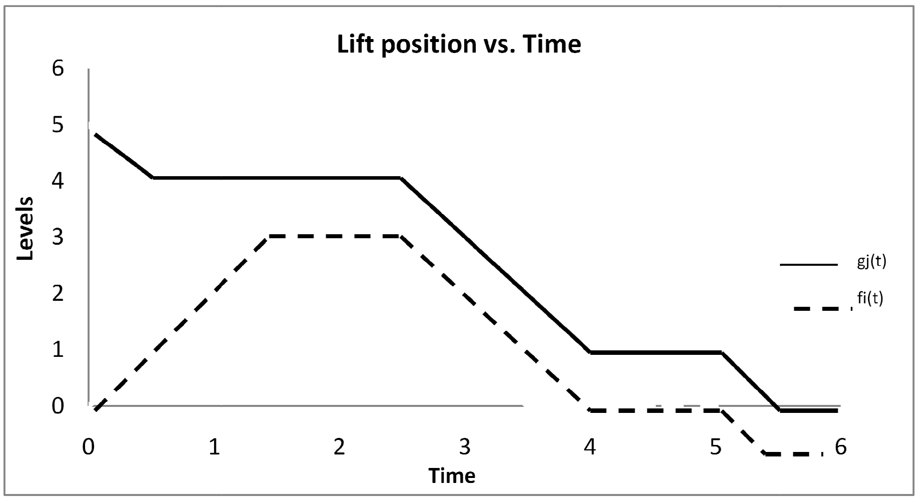

Figure 9: Candidate solution for example in Figure 5

6. Conclusions and Further Research

In this paper, we have studied a dynamic storage system, developed by Vanderlande Industries. The system under study consists of a conveyor, two lifts, multiple transfer shuttles, and a storage rack. New control policies have to be developed to take full advantage of this system in terms of productivity. In this paper, we studied the scheduling problem of the two lifts. An integrated look-ahead heuristic has been developed to both assign requests to lifts and the order in which they will be handled. We propose a heuristic to check for each feasible solution the total time required to handle all requests taking into account constraints with regard to avoiding conflicts between the two lifts. We show by means of an example how this heuristic can be applied. Further studies will be focused on checking the effect of having different priority rules for the lifts. In other words, we will study what happens if we allow lift 1 to have priority, lift 2 to have priority or to alternate between both lifts. Also the strategy will be tested where the lift only starts moving when it can travel directly to its destination (so, we do not allow intermediate parking positions, as for example shown in Figure 9).

References

[1] Zollinger H., "AS/RS application, benefits and justification in comparison to other storage methods: a white paper," Automated Storage Retrieval Systems Production Section of the Material Handling Industry of America, source: http://www.mhia.org/PSC/pdf/AS/RSwhitepaper.pdf (1999).

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

[3] Tompkins, J. A., White, J. A, Bozer, Y. A. and Tanchoco, J. M. A., *Facilities Planning*. (3rd ed.). John Whiley & Sons, New York (2003).

[4] Allen, S.L., "A selection guide to AS/R systems," *Industrial Engineering,* 24, 3, 28-31 (1992).

[5] Vis, I.F.A. and Carlo, H.J., "Sequencing two cooperating automated stacking cranes in a container terminal," *Transportation Science,* 44(2), 169-182.