DEVELOPMENT AND SIMULATION-BASED EVALUATION OF AN ALGORITHM FOR THE RETRIEVAL-IN-SEQUENCE FOR SHUTTLE SYSTEMS

T. Lienert[(a)], W. A. Günthner[(b)]

[(a),(b)]Institute for Materials Handling, Material Flow, Logistics – Technical University of Munich

[(a)]lienert@fml.mw.tum.de, [(b)]kontakt@fml.mw.tum.de

ABSTRACT
Autonomous vehicle-based storage and retrieval systems are used in order to supply picking or production areas based on the goods-to-person principle. In these contexts, it is often required that the point-of-use has to be supplied in a specific sequence of the retrieval loads. This sequence is normally established in the pre-storage area. Depending on the system configuration of an autonomous vehicle-based storage and retrieval system, it is possible to establish the required sequence at every input/output location directly out of the system. In order to enable this retrieval-in-sequence, we present an efficient algorithm that is based on the time-window routing method. For the purpose of evaluation of the performance, we implemented the algorithm in a simulation environment and conducted a series of simulation experiments.

Keywords: autonomous vehicle-based storage and retrieval systems, shuttle systems, time-window routing, sequencing

1. INTRODUCTION

In addition to ordinary automated storage and retrieval systems (AS/RS), a new technology has been developed in the past few years which is based on autonomous vehicles. Autonomous vehicle-based storage and retrieval systems (AVS/RS), also known as shuttle systems, are used for storing small unit loads, as well as pallets, in order to supply picking or production areas based on the goods-to-person principle and to store articles – both for dynamic buffers and for low-access applications (VDI- Richtlinie 2692).
Shuttle systems are characterised by horizontally-operating vehicles. These vehicles travel on each tier along a rail system within the storage rack. For vertical movements, storage and retrieval transactions, the vehicles may use lifts that are positioned at fixed locations along the periphery of the storage rack system (Malmborg 2002).
Typical features of shuttle systems, as compared to conventional stacker-crane-based AS/RS, are a higher performance and a better scalability. The main disadvantages are higher investment and an increasingly complex storage management (Kartnig et al. 2012).
Depending on specific scenarios, storage units might have to be provided in a certain sequence at the point-of-use. This requirement may, for example, arise in the following settings:

- Supply of picking areas in sequence of customer orders
- Supply of gates in sequence of the delivering order of the trucks
- Supply of production areas in the production sequence

Geinitz considered a stacker-crane-based AS/RS and described the effect that retrievals in a required sequence lead to a loss of throughput (Geinitz 1998). Due to this loss of throughput, the required sequence is normally established in the pre-storage area; this consumes space and requires additional material handling systems.
Concerning shuttle systems, neither an algorithm for the retrieval-in-sequence, nor the effect of sequencing, have been described until now. In this paper, we will fill this gap. We first specify the shuttle system configuration that allows sequencing within the storage system. Subsequently, we present an efficient routing-based sequencing algorithm and, finally, we investigate the loss of performance, measured by the throughput, by performing a simulation study.

2. SCOPE OF THE PAPER

This chapter provides an overview of different shuttle system configurations and specifies the configuration that we consider in this paper. Furthermore, we give a literature review about the research done so far which deals with shuttle systems.

2.1. Shuttle System Configurations

In the course of recent developments, different system configurations have evolved. In order to categorise the system configurations, we introduce movement axes that describe the movement space of the vehicles. The x-axis corresponds to the storage aisles. In every system configuration, the vehicles move along the x-axis in order to execute the storage and retrieval requests.

Based on the format of vehicle assignment to storage tiers, Heragu et al. distinguish two different configurations: Shuttle systems that use tier-to-tier vehicles and systems that use tier-captive vehicles. In the tier-to-tier system, the vehicles may move from one tier to another tier using a lift for the vertical movement along the y-axis. In the tier-captive system, each vehicle is dedicated to a single tier and therefore cannot move to another tier. Lifts are used only to move the unit loads to the destination tier (Heragu et al. 2011).We extend that distinction by considering the aisles to which shuttle vehicles are dedicated as well.

In the aisle-to-aisle configuration, there are cross-aisles integrated into the storage rack. These aisles are orthogonally-positioned to the storage racks and correspond to the z-axis. As a consequence, a vehicle can travel from one aisle to another aisle on the same tier. It can reach every position on that tier. In contrast, in the aisle-captive configuration, vehicles are firmly assigned to aisles and movements along the z-axis are not allowed. Figure 1 provides an overview of the four different configurations that result from the different movement spaces of the vehicles.

Autonomous vehicle-based storage and retrieval system				
Aisles-related movements	vehicles are dedicated to a single storage aisle		vehicles can use cross-aisles to change the storage aisle	
Tiers-related movements	vehicles operate on single tier		vehicles can use lifts for horizontal movements and change the tier	
Configurations	aisle - captive tier-captive	aisle-captive tier-to-tier	aisle-to-aisle tier-captive	aisle-to-aisle tier-to-tier
Movement space	x-axis	x-axis, y-axis	a-axis, z-axis	x-axis, y-axis, z-axis

Figure 1: Shuttle System Configurations

A retrieval-in-sequence directly out of the system is possible, only if every storage unit can be provided at every input/output-location (I/O-location). Given the assumption that the I/O-locations are connected to the storage systems by the lifts, the precondition for the retrieval-in-sequence holds for both aisle-to-aisle configurations. In this paper, we consider the aisle-to-aisle and tier-to-tier configuration as it is the more generic one and developed algorithms can be simplified to match the aisle-to-aisle and tier-captive configuration.

Further advantages of the considered configuration are a simple scalability and a good redundancy. As every shuttle can reach every single position within the system, it is possible to run the whole system with a single shuttle. If needed, more and more shuttles can be added to achieve a higher performance. Should a single shuttle fail, depending on the specific layout, it might nevertheless still be possible to reach every single position within the system.

In more detail, the system configuration, which we are considering, can be described as follows.

- Shuttles travel along the x-axis in order to perform storage and retrieval transactions.
- Shuttles are able to change storage aisles by using cross-aisles along the z-axis.
- Shuttles can operate only on the tier they are currently moving on, as they do not have a lifting unit at their disposal.
- Shuttles have unit load capacity.
- The storage rack is single-deep.
- Shuttles are able to change the tier by using a shuttle lift for vertical movements along the y-axis.
- Lifts have single capacity.
- Shuttles do not leave the storage system. On the input/output-level, shuttles remain in the lift while the handover of storage units takes place.
- Every point of use is supplied by a single lift.

The developed algorithm is generic and does not depend on a specific layout. The layout of the storage system may vary, e.g. the number of tiers, the number of aisles and cross-aisles per tier and the number and positions of the lifts. Figure 2 shows an example of the considered configuration.

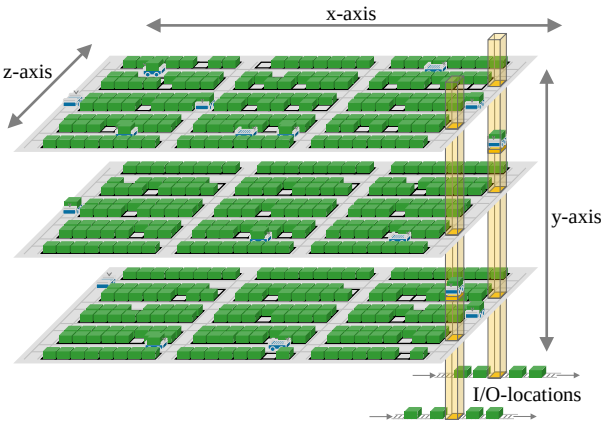

Figure 2: System Example

The most important disadvantage of the considered configuration lies in the complex control strategies that are necessary to run the system in a robust and efficient way. As every single shuttle can reach every single storage position within the system, the question has to be answered as to which shuttle executes which storage or retrieval request. On the other hand, different routing options might exist for reaching a desired position. It therefore has to be decided which path should be taken. Finally, the shuttles share the same rail system, so collisions need to be avoided, as do deadlocks among the shuttles – situations where the shuttles block each other – have to be dealt with.

2.2. Literature Review

Research concerning AVS/RS can be divided into two categories: performance analysis and development of control algorithms. Most of the research papers investigate the performance of the considered system against different parameters, like the storage capacity, rack configuration and numbers of vehicles and lifts. In some papers, various controlling strategies have been developed and tested against each other. Basically, in the literature, two different approaches to performing the system evaluation can be found. Analytical models have been developed, generally based on queuing networks, to investigate the system behaviour. The second approach is based on simulation. In order to identify the impact of different system parameters and controlling strategies, simulation studies were conducted. Malmborg first proposed the idea of an analytical conceptualising tool for AVS/RS. He compared AVS/RS and AS/RS technologies by varying the system configuration (Malmborg 2002). Marchet et al. developed a framework for the design of AVS/RS. The authors considered a tier-captive and aisles-captive configuration and included costs in their model (Marchet et al. 2013). Ekren et al. investigated the effect of several design factors on the performance of a tier-to-tier and aisle-to-aisle AVS/RS. The authors vary the dwell point location, the I/O-location and use basic scheduling rules for both single and dual command scheduling (Ekren et al. 2010). Recent work is more focused on aisle-captive shuttle-systems.

The VDI-Guideline 2962 presents a framework for performance calculation, both for tier-to-tier and tier-captive configurations, in order to estimate possible throughputs achievable with an AVS/RS. The performance calculation is based on the separate determination of the mean cycle times of the lift and of the shuttle. Waiting times of the shuttles for the lift and different controlling strategies are not considered (VDI-Richtlinie 2692). Eder and Kartnig provided an analytical model in order to identify the ideal rack geometry depending on the storage capacity (Eder and Kartnig 2015). Lerher developed an analytical travel-time model for the computation of cycle times for the double-deep storage rack system (Lerher 2015). Carlo and Vis considered an AVS/RS system in which two non-passing lifts share a single mast to transport loads from the horizontally-operating shuttles to the I/O location and vice-versa. The study deals with the scheduling problem of these two lifts, i.e. which lift is going to handle which request and in which order (Carlo and Vis 2012).

Research can be found within the area of automated guided vehicle systems concerning the issue of deadlock-handling that arises in aisle-to-aisle configurations. Kim et al. define a system deadlock as a situation where one or more concurrent processes in a system are blocked forever because the requests for resources by the processes can never be satisfied (Kim et al. 1997). Three different approaches can be distinguished in deadlock handling. Deadlock-prevention is a static approach. A set of generic rules ensures that a deadlock could never occur. This leads to poor resource utilisation and a low performance. In the detection and recovery approach, deadlocks are allowed to occur. They have to be detected and the system uses mechanisms for recovery. As some deadlocks might be hard to discover and, furthermore, deadlocks can overlap each other, detection and recovery might end in an inefficient way. The most frequently used approach is deadlock-avoidance. By using an online control policy, the resources will be dynamically allocated so that a deadlock will never occur (Liu and Hung 2001).

Penners considered a simplified isolated tier of an aisle-to-aisle system (Penners 2015). He adapted two deadlock-avoiding routing-algorithms that have been developed for automated guided vehicles and compared the performance by conducting a simulation study. He came to the conclusion that the time-window routing method, presented by ter Mors et al. (ter Mors et al. 2007), achieves a considerably higher throughput than the modified Banker's routing, which was described by Kalinovcic et al. (Kalinovcic et al. 2011).

In the literature on AVS/RS, it has not been possible to find a paper that deals with sequencing the retrieval loads within a tier-to-tier and aisle-to-aisle configuration. We will therefore present an algorithm that is based on the time-window-routing method and make use of it for the retrieval-in-sequence. As an analytic evaluation of the algorithm is hard due to the complexity of the sequencing problem, we will follow the simulation-based approach to evaluate the performance, which is widespread within the area of warehouse design and management (Roodbergen et al. 2015), (Curico and Longo 2009).

3. SEQUENCING ALGORTIHM

In this section, we briefly describe the system requirements and the options we have for establishing the sequence. We briefly introduce the underlying routing algorithm and finally describe the routing-based sequencing algorithm.

3.1. System Requirements

We follow the assumption that the required sequence of the storage units has to be established at the I/O-location; this means that the final sequence must be made by the lifts. As every point-of-use is supplied by a single lift, we maintain a separate sequence for every lift. As a consequence, we have different sequences within the system that are independent from each other.

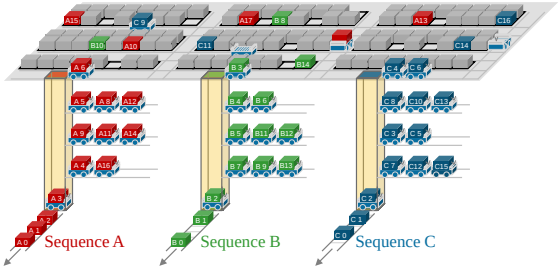

Figure 3: Example with three Independent Sequences

Figure 3 shows a system with three lifts and therefore three independent sequences A, B and C. Within a sequence, the unit loads are marked by increasing sequence numbers. These sequence numbers are unique within a sequence. As the final sequence is established by the lift, the control strategy for the lifts is obvious and simple. For every lift, we will maintain a counter which stores the current sequence number. After having finished a transportation task, the lift will search for the next following sequence number, travel to the corresponding tier, transfer the waiting shuttle to the I/O-location and update the counter. In order to enable this procedure, we have to ensure that on every tier, the shuttles are waiting in an increasing sequence number for the lift; this is the more sophisticated task.

3.2. Sequencing Options

In order to establish the sequence on a single tier, there are different options to be considered:

- Sequencing by dispatching
- Sequencing by clearance
- Sequencing by routing

Dispatching refers to a rule used to select a shuttle to execute a storage or a retrieval request. Egbelu and Tanchoco characterise two different dispatching concepts: tasks-initiated dispatching rules and shuttle-initiated dispatching rules (Egbelu and Tanchoco 1984). In our system, we will use a vehicle-initiated dispatching rule. Whenever a vehicle completes a retrieval request, it will choose the next retrieval request from the set of available requests. The shuttle will choose the retrieval request according to the FIFO rule, this means it will choose a request with the lowest sequence number of one of the independent sequences. By so doing, we pre-sequence the retrieval requests (sequencing by dispatching). Nevertheless, we cannot ensure that the shuttles will arrive in ascending sequence number at the lift. A shuttle to which a retrieval request was assigned with a higher sequence number might be faster in executing the retrieval.
In order to ensure the correct sequence at the lift, we can sequence the shuttles by clearance. In this case, a shuttle waits at its position, after having picked up the retrieval load, until it is cleared by the predecessor on that tier. In figure 4, the shuttle with the sequence number 1 clears the shuttle with the sequence number 2 as soon as it arrives at the lift. We ensure the sequence at the lift, but, depending on the size of a tier, there might be a huge loss of performance, because the second shuttle might take a certain time to travel to the lift. Desirable would be for the shuttle with the sequence number 2 to arrive just after the shuttle with the sequence number 1 had arrived at the lift. This can be achieved by the sequencing by routing, upon which we will focus from now on. The basic idea could be summarised as follows: instead of routing the shuttles from the retrieval location to the lift, we route the shuttles backwards from the lift to the retrieval locations, ensuring the desired sequence at the lift.

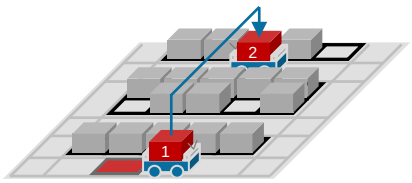

Figure 4: Sequencing by Clearance

3.3. Underlying Routing Algorithm

In order to avoid deadlocks among shuttles moving on the same tier, we make use of the time-window routing method. The basic idea of the algorithm was invented by Kim and Tanchoco, who developed a conflict-free and shortest-time algorithm for routing automated guided vehicles in a bidirectional path network that is based on Dijkstra's shortest path algorithm. The idea of their algorithm consists of modelling the flow path as a graph. Every layout segment corresponds to a single node within the graph (see Figure 5). For each node, the algorithm maintains a list of time-windows reserved by routed vehicles and a list of free time-windows available in which vehicles could be routed. Every free time-window corresponds to a node in the so-called time-window graph. The arcs between these nodes represents the reachability among the free time-windows. The algorithm then routes vehicles through the nodes of the time-window graph instead of the physical nodes of the path network (Kim and Tanchoco 1991). Ter Mors et al. presented an improved version of the time-window routing which is based on the A*-algorithm; their version provides a better worst-case performance and calculates a solution in real-time (ter Mors et al. 2007).
We make use of their time-window routing method in a slightly different way. Instead of constructing the whole time-window graph, we investigate, in every iteration of the routing process, every free time-window on every neighbour node, if that free time-window is reachable from the current time-window. Some conditions must hold for this, e.g. a minimal length or a minimal overlapping of the free time-windows.

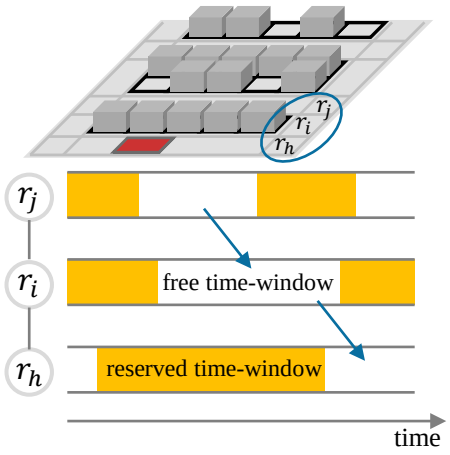

Figure 5: Concept of the Time-Window Routing

The algorithm finds the fastest path for the vehicle from the start node to the destination node at the specified start time under the given reserved time-windows for other vehicles, if such a path exits. The output of the algorithm consists of the sequence of nodes along the path which have to be visited in order to reach the destination and the time intervals, the shuttle will occupy these nodes. The corresponding time-windows will be reserved and the vehicle can travel deadlock-free through the layout.
One might object in that, should a shuttle be delayed, the routing is then no longer robust and deadlocks might occur. However, as Maza and Castagna proved, if the node's crossing order of the shuttles, based on the conflict-free scheduled dates, is fulfilled, then the absence of conflict is guaranteed even if the arrival times are not (Maza and Castagna 2005). In summary, using the time-window method, we do know when a shuttle is likely to arrive at the lift but, furthermore, we also know the sequence in which shuttles arrive at the lift. If the sequence is not correct, we could intervene.

3.4. Routing-Based Sequence-Algorithm

Whenever a shuttle selects the next retrieval request after having finished the current retrieval request, it will travel to the corresponding tier by lift. As soon as the shuttle arrives at that tier, we will run the routing-based sequence algorithm, shown in figure 6.

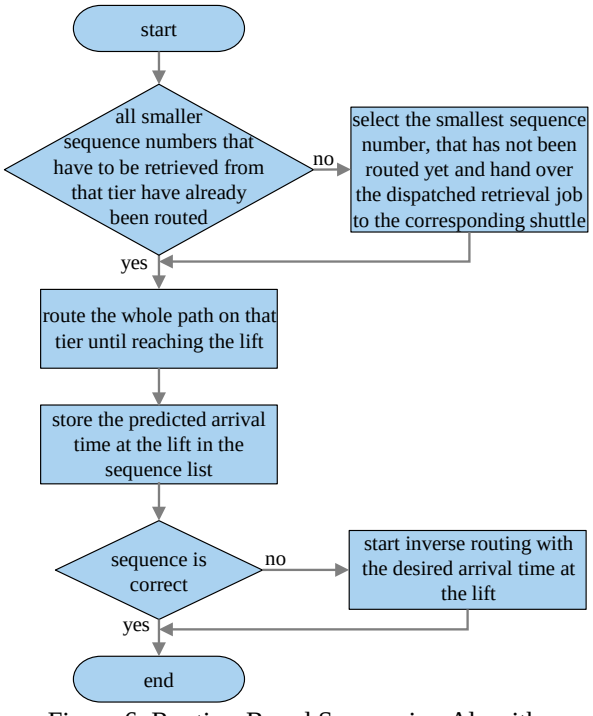

Figure 6: Routing-Based Sequencing Algorithm

Firstly, we will check if all the smaller sequence numbers that have to be retrieved from that tier have already been routed; this would mean that the assigned shuttles are already travelling on that tier and performing the actions. Note that, due to the described dispatching rule, all the smaller sequence numbers must have already been assigned to a shuttle. If all the smaller sequence numbers have not yet been routed, the shuttle would switch its assigned retrieval task and would retrieve the load with the smallest sequence number which has not yet been routed, but is already assigned to another shuttle. As a consequence, this shuttle would retrieve the released sequence number.
Subsequently, the whole path will be routed on that tier; this means, in case of a double-cycle, the path from the entering lift to the storage location, the path from the storage location to the retrieval location and, finally, the path from the retrieval location to the outgoing lift. After having finished the routing, the computed arrival time will be stored in the sequence list and the sequence can be checked. Should the sequence be incorrect, we start the inverse routing from the lift to the retrieval location. Instead of the start time, the inverse routing algorithm requires the desired arrival time at the destination node. The arrival time has to be chosen in such a way that the shuttle might arrive right after the predecessor at the lift.
In the example of figure 7, there are currently three sequence numbers that have to be retrieved from the considered tier. A shuttle travels on that tier and stores the blue box at the designated location. Afterwards, it moves to the storage location of the sequence number 1, picks it up and travels to the lift. The estimated arrival time at the lift is stored in the corresponding sequence list.

Table 1: Sequence List

Sequence No.	Arrival Time
1	38
2	
3	

Now that a second shuttle has arrived at the tier that has been assigned to retrieve the sequence number 2, we run the algorithm. Firstly, it can be seen that all the smaller sequence numbers have already been routed by searching for the arrival time in the sequence list. Then, the path to the storage location of the green box, the path from the storage location to the retrieval location of sequence number 2 and, finally, the path to the lift will be completely routed. According to the routing, the shuttle will arrive at the lift at $T_{arrival(lift)} = 30s$. This information will be stored in the sequence list.

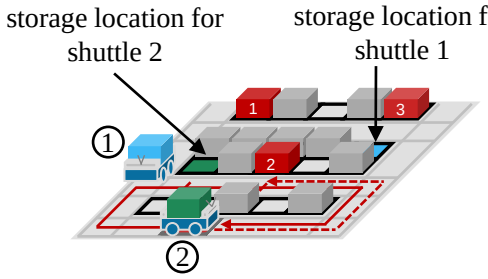

Figure 7: Inverse Routing from the Lift to the Retrieval Location

Table 2: Updated Sequence List

Sequence No.	Arrival Time
1	38
2	30
3	

As can be seen, sequence number 2 will arrive at the lift before sequence number 1. As a consequence, we start the inverse routing with the desired arrival time $T_{arrival(lift)} = 42s$. Assuming a shuttle needs four seconds for entering an intersection, changing the moving direction by 90 degrees, and leaving the intersection entirely, this is the earliest possible arrival time at the lift for the second shuttle after the arrival of the first shuttle.
Only the last segment from the retrieval location to the lift will be rerouted, marked by the dotted line in figure 7. As the shuttle will now arrive later than originally planned, it will wait for a certain time at the retrieval location after having picked up the load.
We follow the idea of the time-window routing, but instead of routing the shuttle ahead, we let the shuttle route backwards. We therefore have to modify the achievable conditions to decide whether a free time-window on neighbour node is reachable from the current free time-window or not. A free time-window is defined by its start and its end time.

- $Start_{i,k}$: Start time of the k-th free time-window on the node r_i
- $End_{i,k}$: End time of the k-th free time-window on the node r_i

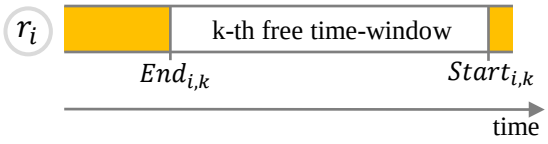

Figure 8: Free Time-Window

Note, that as we are routing backwards, it counts as $Start_{i,k} > End_{i,k}$.
We define the following time-stamps and time durations which describe the movement of a shuttle through the nodes of the layout-graph.

- $T_{entry(r_i)}$: Time-stamp, when the shuttle enters the node r_i
- $T_{arrival(r_i)}$: Time-stamp, when the shuttle resides completely on the node r_i and has completely left the previous node
- $T_{exit(r_i)}$: Time-stamp, when the shuttle has left the node r_i completely
- t_{trans} : Time a shuttle needs to enter or exit a node. More precisely, for the time t_{trans} the shuttle occupies two subsequent nodes
- $t_{cross(r_i)}$: Time a shuttle needs to cross the node r_i, should the node's length exceed the length of a shuttle
- $t_{wait(r_i)}$: Waiting time on the node r_i

The following figure illustrates the relations between the defined variables:

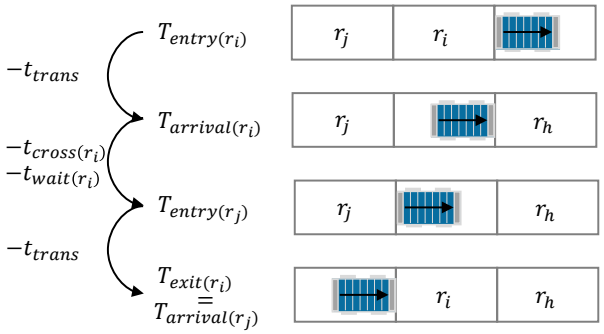

Figure 9: Time-Stamps and Time Durations Describing the Shuttle Movement.

To decide whether a free time-window on a neighbour node is reachable from the current time-window, there are some conditions that must hold:

$$Start_{j,l} - End_{j,l} \geq t_{trans} + t_{cross(r_j)} + t_{trans} \quad (1)$$

$$Start_{j,l} > End_{i,k} \quad (2)$$

$$Start_{i,k} > End_{j,l} \quad (3)$$

The first condition states that the free time-window on the neighbour node has the minimal size; this means that the shuttle might have time to enter the node, cross it and leave it within the free time-window. Condition (2) and (3) ensure that the two time-windows overlap. If, until now, all the conditions have held, the entry time into the free time-window and thereby into the node is calculated:

$$T_{entry(r_j)} = \min\{Start_{j,l}, T_{arrival(r_i)} - t_{cross(r_i)}\} \quad (4)$$

The entry into the free time-window cannot occur before the free time-window starts and, of course, not before the shuttle has crossed the current node in its entirety. If the entry time corresponds to the start time of the free time-window, the shuttle has to wait on the current node. After having calculated the entry time, the last conditions can be verified:

$$T_{entry(r_j)} - t_{trans(r_i)} = T_{exit(r_i)} \geq End_{i,k} \quad (5)$$

$$T_{entry(r_j)} - End_{j,l} \geq t_{trans} + t_{cross(r_j)} + t_{trans} \quad (6)$$

Condition (5) ensures that the shuttle might leave the current node before the time-window ends, and condition (6) ensures that the remaining size of the free time-window on the neighbour node is sufficient to enter the node, cross and leave it again. If all these conditions also hold, the free time-window can be reached from the current time-window.
In the example shown in figure 10, the free time-windows on both nodes overlap. The entry time corresponds to the arrival time on the node r_i minus the time it takes to cross this node. Furthermore, the shuttle can leave the node r_i before the free-time window ends as $T_{exit(r_i)} \geq End_{i,k}$. Finally, the remaining size of the free time-window on node r_j is sufficient to cross it and leave it, which is marked by the hatched elements. As a result, the free time-window on node r_j is reachable form the current time-window on node r_i with the given entry time $T_{entry(r_i)}$ into that node.

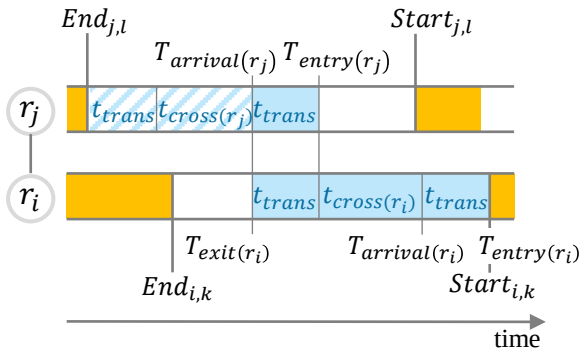

Figure 10: From the Current Time-Window on Node r_i the Free Time-Window on Node r_j is Reachable.

There are two cases which cause difficulties when applying the inverse routing. Firstly, it might happen that there is no route found which ends at the desired arrival time. Appling the time-window routing algorithm, the shuttle can delay its departure on the start node until a route is found. However, as we are routing backwards, we have to delay the arrival time manually and start the inverse routing again.
The second problem might arise under certain traffic conditions within the system. The sequence at the lift is incorrect and an inverse routing is required, given the desired arrival time at the lift. The inverse routing leads to an earlier departure time at the retrieval location than was originally predicted, by routing the path from the storage to the retrieval location on the tier. This is, of course, not valid as the shuttle could not start travelling to the lift before it had finished loading the retrieval load. We have to delay the desired arrival time for the inverse routing again until a valid departure time is found. Figure 11 clarifies the described problem.

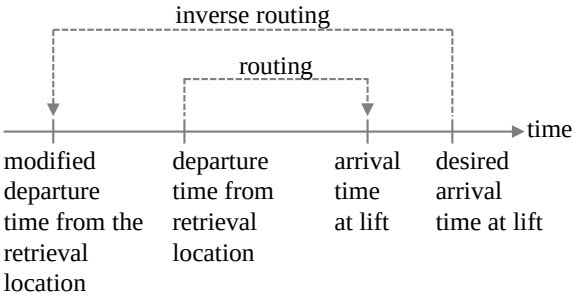

Figure 11: Invalid Inverse Routing

In the subsequently-described simulation, we delayed the arrival time in both cases in steps of one second.
Once the route is found, the time-windows will be reserved. The shuttle can leave the lift and start travelling on the tier. So, the requested sequence at the lift will be guaranteed.

4. SIMULATION-BASED EVALUATION

In order to test the sequencing algorithm and to quantify the loss of throughput by the sequencing, we implemented the algorithm in a simulation environment and conducted a serious of simulation experiments. The modelling and implementation is briefly described before the experiment settings are clarified and the results presented and discussed.

4.1. Modelling and Implementation

We implemented the simulation model using the discrete-event simulation software, Plant Simulation. This software offers various customisable modules, e.g. roads, warehouses, working stations and vehicles. The control of information and material flows is realised by procedures programmed by the user and assigned to a specific event.
In order to model the storage system, we divided the layout of a single tier into various resources. A resource is a layout segment with capacity for exactly one shuttle. Every resource corresponds to a node within the graph that is used for the time-window-based routing. Resources are storage elements, cross-aisles segments and intersections. In order to avoid having to control the driving direction of the vehicles, we allowed the vehicles to only travel in one direction. We modelled the different directions by two opponent path segments which are part of the same resource.

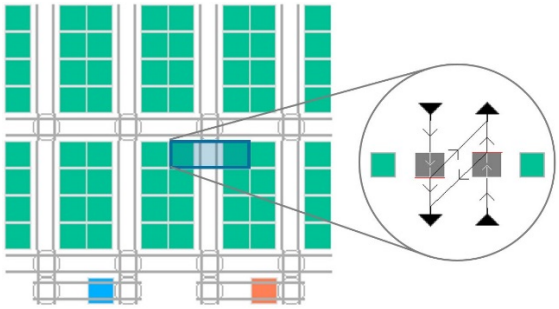

Figure 12: Modelling Concept

Figure 12 shows a single tier and a storage element resource. As time-windows are maintained for every resource, it is ensured that only one shuttle can occupy a resource at a time, even though it consists of several path segments. The routing and sequencing algorithm is implemented in a central controlling module.
The model allows for the varying of a number of parameters, such as the number of shuttles or the used dispatching strategy. The layout of the considered storage system can be generated automatically or individually constructed. As the shuttles can only reside on a resource they have reserved in advance, the question arises as to how to initialise a simulation run. We solved this problem by introducing a virtual buffer for every lift within the system. At the beginning, the shuttles were distributed equally to these virtual buffers. From these buffers, the shuttles requested the lift one after another and were inserted into the storage system.

4.2. Simulation Experiments

In order to test the sequencing algorithm, we built a model consisting of four tiers with 320 storage locations each. We integrated four lifts, which are supplied by the shuttles, to obtain more than one independent sequence. Basically, the sequencing algorithm runs on a single tier, but in order to also respect the influences of the lifts, we extended the system to three more tiers. Variations of the layout were not part of the simulation study and should be considered in further research.

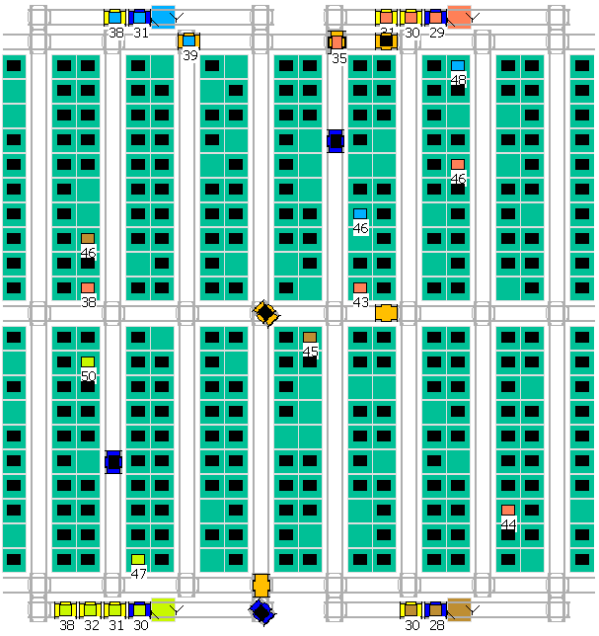

Figure 13: A Single Tier of the Simulation Model

The shuttles work constantly in double-cycles. Every time a retrieval request is fulfilled, the shuttle selects the retrieval job with the oldest time-stamp (and with the smallest sequence number) from the available retrieval requests. Furthermore, a storage load is handed over to the shuttle. The storage location is chosen randomly on the same tier upon which the assigned retrieval load is situated. By so doing, additional travel with the lifts is avoided and the storage ratio on every tier is kept constant. We generated a new retrieval task for one of the sequences every time a retrieval of that sequence was completed. The retrieval location was chosen randomly.
Figure 13 shows the layout of a single tier from the simulated system. Shuttles are moving on that tier and performing retrieval and storage transactions. As soon as the storage loads are requested, they are coloured according to the lift that connects the storage system with the corresponding I/O-location. Furthermore, the sequence numbers are assigned. As can be seen, the shuttles wait for the lift in increasing sequence number of the retrieval loads. Numbers that do not appear have to be retrieved from a different tier than the one that is shown. We varied the numbers of shuttles working in the system from 5 to 65 in steps of 5. We ran the simulation twice with every number of shuttles. Firstly, the retrieval-in-sequence was required and then it was not. We conducted 5000 double-cycles and measured the time needed to fulfil these cycles. As the storage and retrieval locations were randomly determined, we did three replications per experiment and calculated the mean time for the evaluation.

4.3. Results and Discussion

The developed algorithm for the retrieval-in-sequence was validated by conducting the simulation experiments. At every I/O-location, the sequence numbers of the retrieval units were stored and the correctness of the sequence was verified.

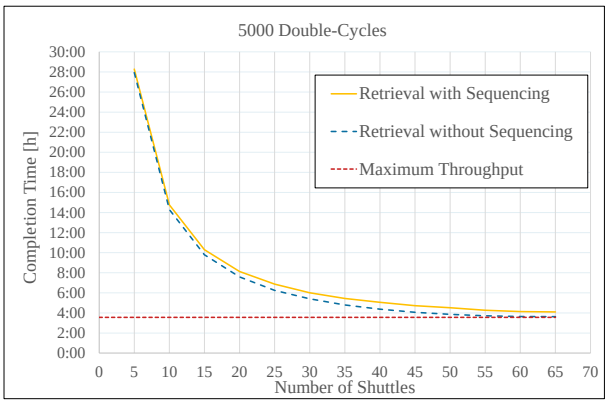

Figure 14: Results of the Simulation Experiments

Figure 14 shows the evolution of the completion time for 5000 double-cycles as a function of the numbers of shuttles working in the system, both for the retrieval with sequencing and retrieval without sequencing. As expected, the completion time lessens with the number of shuttles working in the system, and the retrieval with sequencing requires a higher completion time than the retrieval without.
The loss of performance caused by the retrieval with sequencing for the different numbers of shuttles is presented by figure 15. As can be seen, the loss of throughput is small when there are only a few shuttles

in the system. With only a few shuttles, the interference among these is relatively small. In other words, the probability that two shuttles execute a retrieval request on the same tier for the same sequence is small.
The loss of throughput rises to 14.60% with 50 shuttles working in the system. With a higher number of shuttles, the loss of throughput lowers slightly again.

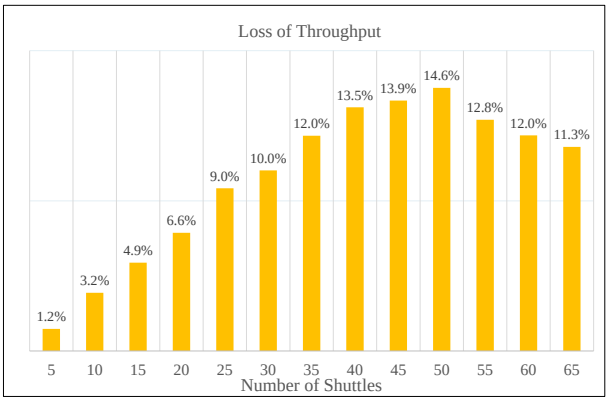

Figure 15: Loss of Throughput caused by the Retrieval in Sequence

The maximum system throughput is limited by the lifts. If the lifts work continuously, the capacity cannot be augmented by adding more shuttles. Using a chaotic storage assignment strategy, every tier is piloted by the shuttles with the same frequency. The mean time of a double-cycle of a lift can be calculated using a combinatorial approach, similar to the one presented in the VDI-Guideline (VDI- Richtlinie 2692). The result of this maximum throughput calculation is presented by the dotted line in figure 14. At a certain number of shuttles, the throughput cannot be further significantly augmented. This number is reached earlier if the retrieval in sequence is not required which explains the decreasing loss of throughput for a high number of shuttles.
From the chart in figure 14, information can also be obtained as to how many more shuttles are needed to compensate for the loss of throughput caused by the retrieval-in-sequence. For instance, if there are currently 30 shuttles in the system, approximately five more shuttles are needed to reach the same throughput with the retrieval-in-sequence.

5. SUMMARY

In this paper, we considered an autonomous vehicle-based storage and retrieval system, where the vehicles move along the x-axis and the z-axis on a tier and use lifts to change the tier along the y-axis. We described an algorithm for the retrieval-in-sequence that is based on the time-window routing method and which ensures the absence of deadlocks. The basic idea of the sequencing algorithm consists of an inverse routing that is applied if the required sequence is not correct. We modelled and implemented the storage system in a simulation environment in order to test the algorithm and to quantify the loss of throughput by the retrieval-in-sequence.
An interesting topic for future work would be the determination of the number of shuttles needed to achieve the maximum system throughput. This number clearly depends on the number of lifts, the layout and the specific parameters of the lifts and vehicles. Furthermore, different dispatching strategies, such as task-initiated despatching rules and different layout options, should be investigated and analysed.

REFERENCES

Carlo H. J., Vis I. F. A., 2012. Sequencing dynamic storage systems with multiple lifts and shuttles. International Journal of Production Economics 140: pp. 844-853.

Curico, D., Longo, F., 2009. Inventory and internal logistics management as critical factors affecting the Supply Chain performances. International Journal of Simulation and Process Modelling. International Journal of Simulation and Process Modelling 5: pp. 278-288

Eder M., Kartnig G., 2015. Throughput analysis of S/R shuttle systems and ideal geometry for high performance. Proceedings of the XXI International Conference MHCL, pp. 193-198. September 23-25, Vienna (Austria).

Egbelu P. J., Tanchoco J. M. A., 1984. Characterization of automatic guided vehicle dispatching rules. International Journal of Production Research 22: pp. 359-374.

Ekren B. Y., Heragu S. S., Krishnamurthy A., Malmborg C. J., 2010. Simulation based experimental design to identify factors affecting performance of AVS/RS. Computers & Industrial Engineering 58: pp. 175-185.

Geinitz, J., 1998. Unerkannte Abhängigkeiten mindern die Leistungsfähigkeit automatisierter Lager. Marktbild Lager 18: pp. 16-18.

Heragu S. S., Cai X., Krishnamurthy A., Malmborg C. J., 2011. Analytical models for analysis of automated warehouse material handling systems. International Journal of Production Research 49: pp. 6833-6861.

Kalinovcic L., Petrovic T., Bogdan S., Bobanac V., 2011. Modified banker's algorithm for scheduling in multi-agv systems. Automation Science and Engineering (CASE), pp. 351–356. August 24-27, Trieste (Italy).

Kartnig G., Grösel B., Zrnic N., 2012. Past, State-of-the-Art and Future of Intralogistics in Relation to Megatrends. FME Transactions 40: pp. 193-200.

Kim C. W., Tanchoco J. M. A., 1991. Conflict-free shortest-time bi-directional AGV routing. International Journal of Production Research 29: pp. 2377-2391.

Kim C. W., Tanchoco J. M. A., Koo P., 1997. Deadlock Prevention in Manufacturing Systems with AGV Systems: Banker's Algorithm Approach. Journal

of Manufacturing Science and Engineering 119: pp. 849-854.
Lerher T., 2015. Travel-time model for double-deep shuttle-based storage and retrieval systems. International Journal of Production Research, pp. 1-22.
Liu F., Hung P., 2010. Real-time deadlock-free control strategy for single multi-load automated guided vehicle on a job shop manufacturing system. International Journal of Production Research 39: pp. 1323-1342.
Malmborg C. J., 2002. Conceptualizing tools for autonomous vehicle storage and retrieval systems. International Journal of Production Research 40: pp. 1807-1822.
Marchet G., Melacini M., Perotti S., Tappia E., 2013. Development of a framework for the design of autonomous vehicle storage and retrieval systems. International Journal of Production Research 51 pp. 4365-4387.
Maza S., Castagna P., 2005. A performance-based structural policy for conflict-free routing of bi-directional automated guided vehicles. Computers in Industry 56: pp. 719-733.
Penners L. T. M. E., 2015. Investigating the effect of layout and routing strategy on the performance of the Adapto system. Master's Thesis. Eindhoven University of Technology.
Roodbergen, K J., Vis I. F. A., Taylor, G.D., 2015. Simultaneous determination of warehouse layout and control policies. International Journal of Production Research 53: pp. 3306-3326.
ter Mors A. W., Zutt J., Witteveen C., 2007. Context-Aware Logistic Routing and Scheduling. Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, pp. 328-335. September 22-26, Providence (USA).
VDI-Richtlinie 2692 Blatt 1, 2015. Automated vehicle storage and retrieval systems for small unit loads. Berlin: Beuth.

AUTHORS' BIOGRAPHY

Thomas Lienert, M. Sc. was born in Hamburg, Germany, and studied industrial engineering and management at the Karlsruhe Institute of Technology and the University of Seville with the focus on operations research and logistics systems. He has been working as a research assistant at the Institute for Materials Handling, Material Flow and Logistics, Technical University of Munich since 2014. His research mainly deals with the development of control strategies for autonomous vehicle-based storage and retrieval systems.

Prof. Dr.-Ing. Dipl.-Wi.-Ing. Willibald A. Günthner is head of the Institute for Materials Handling, Material Flow and Logistics, Technical University of Munich. He is co-founder of the Wissenschaftliche Gesellschaft für Technische Logistic e.V. and advisory board member of several associations, federations and companies.