FAILURE-HANDLING STRATEGIES FOR MOBILE ROBOTS IN AUTOMATED WAREHOUSES

Thomas Lienert
Ludwig Stigler
Johannes Fottner
Chair of Materials Handling, Material Flow, Logistics
Department of Mechanical Engineering
Technical University of Munich
Boltzmannstrasse 15, 85748 Garching, Germany
Email: thomas.lienert@tum.de, ludwig.stigler@tum.de, j.fottner@tum.de

KEYWORDS
Automated Warehouses, Mobile Robots, Failure Handling, Time Window Routing Method, Discrete Event Simulation

ABSTRACT

Automated warehouses operated by a fleet of robots not only offer great flexibility, as fleet size can be adjusted easily to throughput requirements, they also provide higher redundancy compared to common solutions for automated storage and retrieval systems. In case a single robot fails, the remaining fleet of robots is able to continue working within the system, so that throughput is only slightly affected.
However, adequate strategies are required for this scenario. In this contribution, we present four different approaches to cope with robot downtimes, which are based on the routing of the robots. These strategies are compared by performing a simulation study in which a robotic mobile fulfilment system is considered.

INTRODUCTION

In addition to common stacker-crane-based automated storage and retrieval systems, a new type of automated warehouse has been deployed within the past few years that is used for part-to-picker order picking. This type basically consists of a rack system containing storage items and a fleet of vehicles moving within the storage area, fulfilling storage and retrieval requests.
There are two main categories that can be distinguished. In the first category, vehicles move on a single storage tier. The products are stored on racks arranged in storage aisles on the ground (figure 1). Whenever a certain storage item is needed, a vehicle travels towards the rack containing the item, lifts the rack and brings it to the picking zone, which is located somewhere near the storage area. The item is picked by an operator and the rack is brought back to an empty storage location. Vehicles travel along the storage aisles as well as underneath the racks, insofar as they are not carrying one of these.
In literature, these systems are denominated as robotic mobile fulfilment systems (RMFS) and the vehicles as robots (Azadeh et al. 2018).

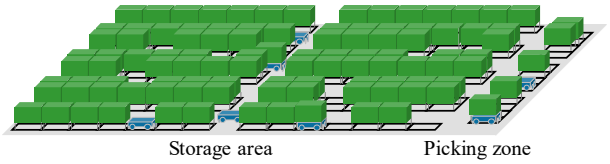

Figure 1: Robotic mobile fulfilment system

The second type of system consists of several storage tiers, connected by lifts (figure 2). In some configurations, these lifts are used only for the vertical transport of the storage items, whilst the vehicles always operate on the same storage tier. In other configurations that we consider, lifts are used to transfer vehicles between the storage tiers and the input/output (I/O) locations. On every storage tier, vehicles use a grid of storage and cross-aisles to reach the storage locations. In literature, these systems are called shuttle systems and the vehicles are denominated as shuttles (Tappia et al. 2018).
We refer to both of theses types – RMFS and shuttle systems – as mobile-robot-based warehouses. One of the main characteristics is that every robot can reach every storage location. The system can therefore theoretically be operated with a single robot. Depending on the required throughput, the fleet size (i.e. the number of robots working in the system) can be adjusted. Furthermore, in the event that a single robot fails, the remaining robots are able to continue responding to storage and retrieval requests and the system's throughput is only slightly affected.

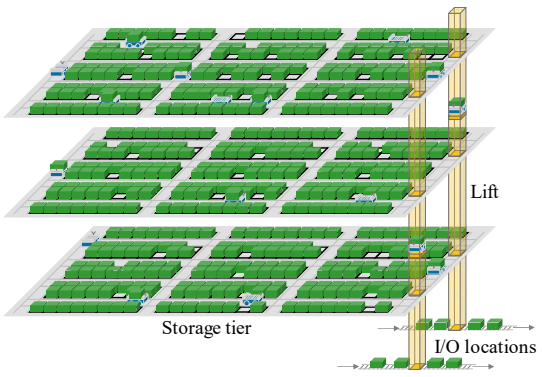

Figure 2: Shuttle system

Communications of the ECMS, Volume 33, Issue 1,
Proceedings, ©ECMS Mauro Iacono, Francesco Palmieri,
Marco Gribaudo, Massimo Ficco (Editors)
ISBN: 978-3-937436-65-4/978-3-937436-66-1(CD) ISSN 2522-2414

In literature, RMFS as well as shuttle systems are widely discussed. There are several decision problems that need to be addressed during the planning phase as well as during the operation, such as layout design (Lienert et al. 2018), storage assignment (Boysen et al. 2018), order batching (Boysen et al. 2017), dispatching (Yuan and Gong 2017), traffic management (Merschformann et al. 2017), battery charging and swapping (Zou et al. 2017) as well as dwelling strategies for idle robots (Roy et al. 2016).
In general, different strategies are developed and compared to each other by using analytical models, often based on queuing networks, or by conducting simulation studies. However, robot downtimes are not considered, although these have an impact on the attainable throughput. Reasons for downtimes are manifold – for instance the interruption of the power supply, contaminations, insufficient maintenance and unstable loads.
In this work, we present different strategies for handling downtimes and compare these by performing a simulation study.
The reminder of this paper is organized as follows. Next, we briefly introduce the time window routing method, as our strategies are based on this conflict-free routing approach. Subsequently, we describe four different failure-handling strategies. We apply these to an RMFS and compare the results before we conclude our work.

TIME WINDOW ROUTING METHOD

To run mobile-robot-based warehouses robustly and efficiently, complex control strategies are needed. Amongst others, traffic needs to be managed to avoid congestion, blocking and collisions among robots. One option for coping with these challenges is routing based on time windows: Before a robot starts moving, the whole path is reserved – from its current position towards the destination. To apply this method, the layout (of each storage tier) is modelled as a graph. There is a timeline with reserved and free time windows for each node representing a layout section (figure 3).

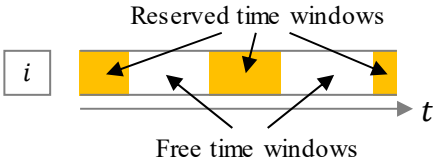

Figure 3: Reserved and free time windows on a node i

If a route has to be calculated, the procedure searches for a conflict-free route through the free time windows using an A*-algorithm. The required time windows are reserved and the robot can start executing the computed route.
The idea of this method was introduced first by Kim and Tanchocco (Kim and Tanchocco 1991) and has been applied in different contexts – for instance routing automated guided vehicles in container terminals (Stenzel 2008), organizing aircraft taxi traffic at airports (Bussacker 2005) or in general moving agents over an infrastructure (ter Mors 2010). Furthermore, the concept has been used for organizing a fleet of robots both in RMFS (Hvězda et al. 2018) and shuttle-systems (Lienert and Fottner 2017).
The absence of deadlocks can even be guaranteed, in the case where some robots are late and do not meet their reserved time intervals. The crucial point is that the node's crossing order of the robots, based on the conflict-free schedule, is maintained (Maza and Castagna 2005).
In a previous work, we modified the method to incorporate acceleration and deceleration processes, which are usually neglected. During the planning, so-called "segments" are created, which describe a movement of a robot over several nodes in a straight line. The computed route is executed segment by segment, respecting the node's crossing order (Lienert and Fottner 2018).
Figure 4 shows a fragment of a layout graph (nodes $i, \dots, m$) and the corresponding timelines. For the robot r_1 there is a segment planned that comprises a movement over five nodes. Furthermore, there are some more reservations that belong to the routes of other robots ($r_2, \dots r_4$). If robot r_2 is delayed, then robot r_1 is forced to decelerate and shorten the segment by introducing an intermediate stop on node l in order to meet the sequence of reservations on node m (robot r_2 before robot r_1). Note, that reservations not only comprise occupation time themselves (marked by the trajectories), but also additional buffer times in advance of the reservations, so as to prevent collisions should delays occur.

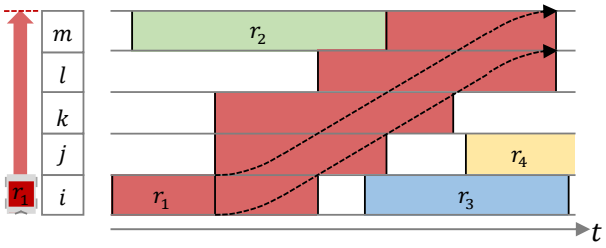

Figure 4: Segment and reservations of other robots

There are two reasons why the routing procedure might not lead to a successful result, and thus no route towards the destination being available at the given time. The first reason is an endless reservation of an idle robot placed on the destination node or on a node that belongs to the only available path towards the destination node.
The second reason is that the time window from which the routing starts is bounded by another reservation on the same node. In figure 5, routing towards the destination node k, starting at timestamp t_{Start}, is unsuccessful because the current free time window on start node i cannot be left before the reservation of robot r_3 starts.

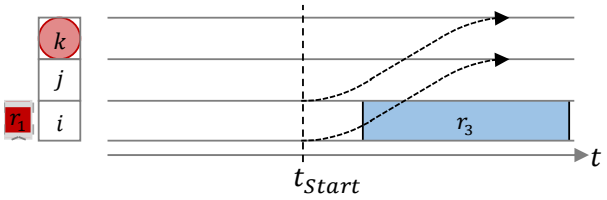

Figure 5: Routing is unsuccessful due to another reservation on the start node

The failure-handling strategies presented in the following section are based on the time window routing method.

FAILURE-HANDLING STRATEGIES

In this section, we describe four different strategies to cope with downtimes. We assume that a downtime only occurs before a robot starts with the execution of the next segment of its route. This limits the state space of the system and the strategies can be simplified – for instance, a breakdown can never occur during a loading or picking process. Furthermore, we assume that a robot resumes working after a certain time span, the mean time to repair (MTTR) has elapsed.
All presented strategies are generic in the sense they are not designed to fit a certain layout. The strategies only manipulate the routes. Orders – retrieval or storage tasks – that are already assigned to robots are not modified.

Strategy 1: Ignore

The first strategy is simple and straightforward: robot failures are strictly ignored by the control. As soon as a failure occurs, the corresponding robot stops and remains stationary on the node where it is currently located. As a consequence, this node is blocked and other robots are prevented from passing that node. These robots are forced to interrupt the execution of their routes. Furthermore, all robots that reserved a time window on one of the nodes which are part of the remaining route of the broken robot have to stop, to the extent that they are supposed to pass the node after the broken robot.
With reference to figure 4, if robot r_1 breaks down and thereby blocks its current node i, it is not only robot r_3 which is affected, but also robot r_4. Note, that these robots might block yet others.
As soon as the MTTR has elapsed, the broken robot resumes the execution of its originally computed route, and blocked robots are triggered to continue their routes as well.

Strategy 2: Pause

The second approach is as simple as the first one, but more restrictive. As soon as a failure occurs, all robots operating within the system are forced to interrupt the execution of their routes. More specifically, if there is a breakdown present in the system, a robot will not start with the execution of the next segment of its route. As soon as the MTTR elapsed, all robots resume with the execution of their originally computed routes. Note, that all robots will be significantly delayed.
This strategy does not take advantage of the higher redundancy mentioned above. However, it serves as a lower bound for the comparison of the throughput.

Strategy 3: Restart

The idea behind the third strategy is to stop all robots once again, but then to perform a restart where all routes are recalculated avoiding the node blocked by the broken robot.
As soon as all robots have come to a standstill, their remaining routes – and more specifically their reserved time windows – are deleted. Next, each robot reserves an endless time window on its current node. That prevents any other robot from routing over that node. All robots (besides the broken one) are added to a list of robots that need to be routed.
Next, the restart procedure as described by the flowchart in figure 6 commences.

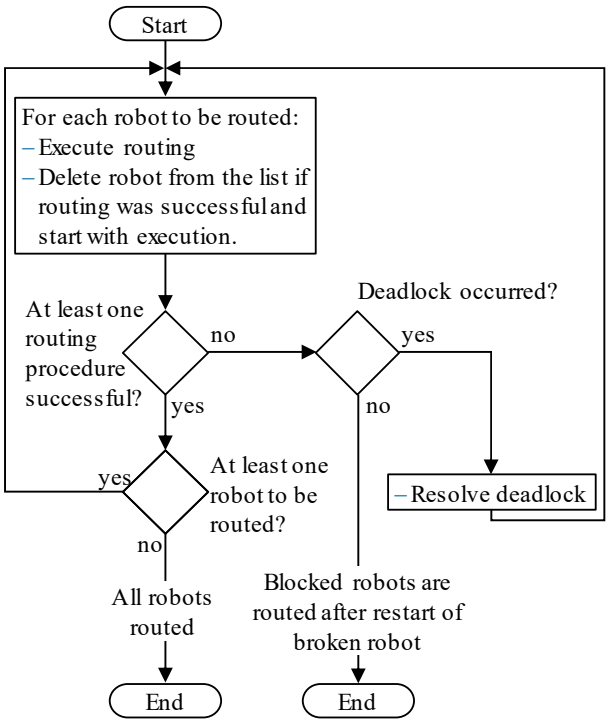

Figure 6: Procedure: restart of all robots to be routed

Routing is executed for each robot on the list. If routing is successful, the robot is removed from the list, the endless time window is deleted so the node can be accessed by other robots, and the execution of the newly constructed route starts. Otherwise, the robot will remain on the list and will be routed once again in the subsequent iteration.
If, in an iteration, at least one robot could be routed successfully, the routing will restart with the first robot remaining in the list again. The iterations continue until all robots have been routed successfully.

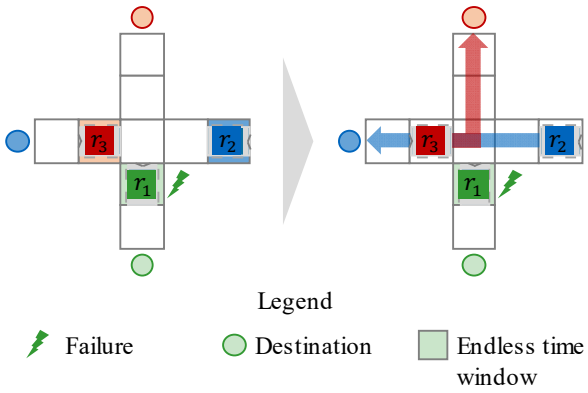

Figure 7: Robot r_2 and r_3 are restarted

In the example in figure 7, the routing of robot r_2 will fail during the first iteration and is only possible after robot r_3 has been routed.
There are two reasons why it is not possible to route a single robot during an iteration. First, a robot is prevented from reaching its destination by the endless reservation of the broken robot. In that case, the robot remains blocked until the broken robot resumes working.
As soon as the MTTR has elapsed, the flowchart in figure 8 is executed.

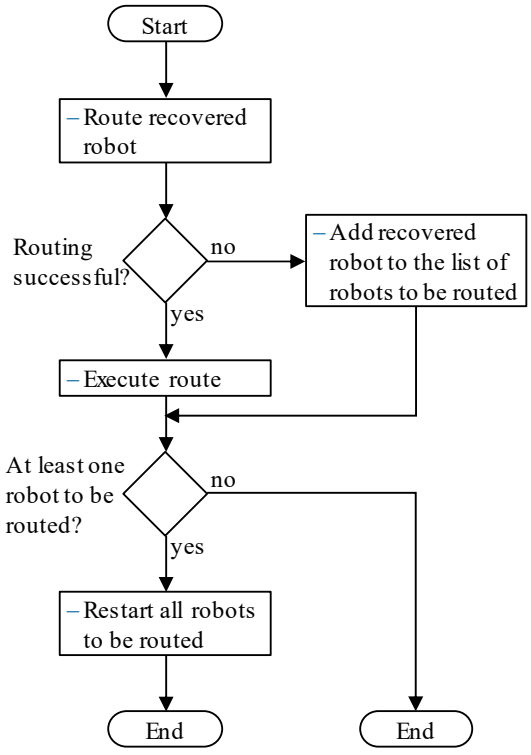

Figure 8: Procedure: restart of a single robot

After the recovered robot has been routed, blocked robots are dealt with by performing the restart procedure (flowchart in figure 6) once again.
Note that the recovered robot is added to the list of robots to be routed if routing fails.

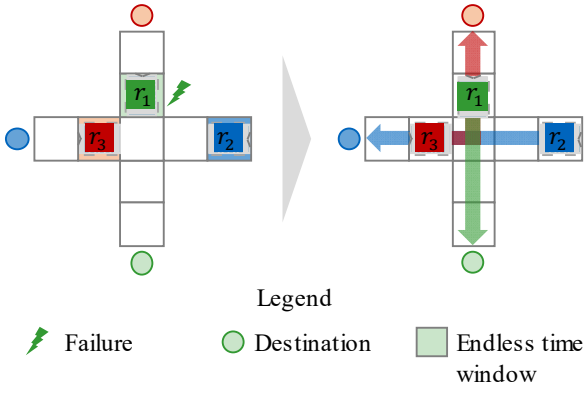

Figure 9: Robot r_2 and $r_3$can only be routed after broken robot r_1 resumes working

In the example in figure 9, robot r_3 cannot be routed successfully, as the broken robot r_1 blocks the only available path. Robots r_3 as well as r_2 will be restarted after the MTTR has elapsed and a route for robot r_1 has been recalculated, such that the endless time window has been deleted. The second reason is that two or more robots block each other for endless time (Figure 10, a). In that case, an intermediate destination will be assigned to one of the robots involved in that deadlock (Figure 10, b). As soon as the robot reaches that intermediate destination (Figure 10, c) the routing towards the original destination takes place (Figure 10, d).

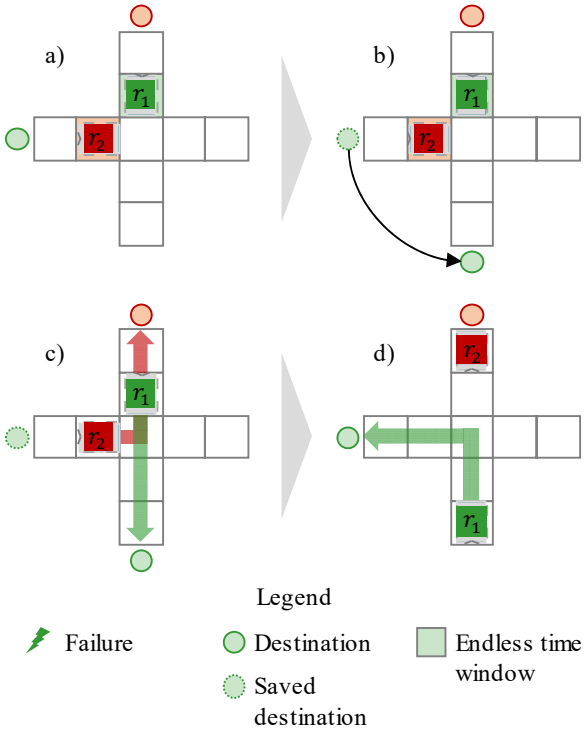

Figure 10: Deadlock resolution

The strategy *Restart* can also be used to initialize a system without failures, adding all robots located somewhere in the system to the list of robots to be routed.

Strategy 4: Reroute

The fourth strategy consists in rerouting only robots that are directly affected by the broken robot.
If a failure occurs, the execution of the computed route of the robot is stopped. As the flowchart in figure 11 describes, all robots that reserved a time window on that node are identified and added to the list of robots to be rerouted.
Referring to figure 4, if robot r_2 breaks down on node m, robot r_1 has to be rerouted. Next, the route of the broken robot is deleted as well as all reserved time windows on its current node. An endless time window on the node is reserved.

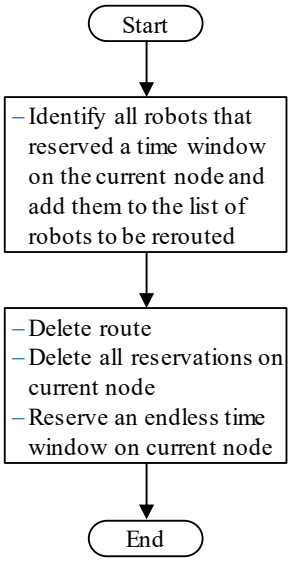

Figure 11: Procedure: end execution

Whenever a robot is ready to execute the next segment of its route, it will be checked whether that robot must be rerouted. In that case, the remaining route of the robot is deleted and the routing is again executed. If routing is successful, an alternative route without the blocked node is found and the execution of the newly calculated route starts.
If there is no route available due to an endless reservation, the execution is ended as described previously. The robot is treated as if it were broken, and other robots which routed using the robot's current node are affected, and need to be rerouted. As soon as the MTTR has elapsed, the recovered robot is restarted as in the previously described strategy (flowchart in figure 8), which leads to a restart of all blocked robots to be routed (flowchart in figure 6).
If another time window on the same node impedes a successful routing as shown in figure 5, the execution is once again ended, though this is immediately followed by a restart (flowchart in figure 8). Note that ending execution deletes the limiting time window.
The flowchart in figure 12 summarizes the described procedure.

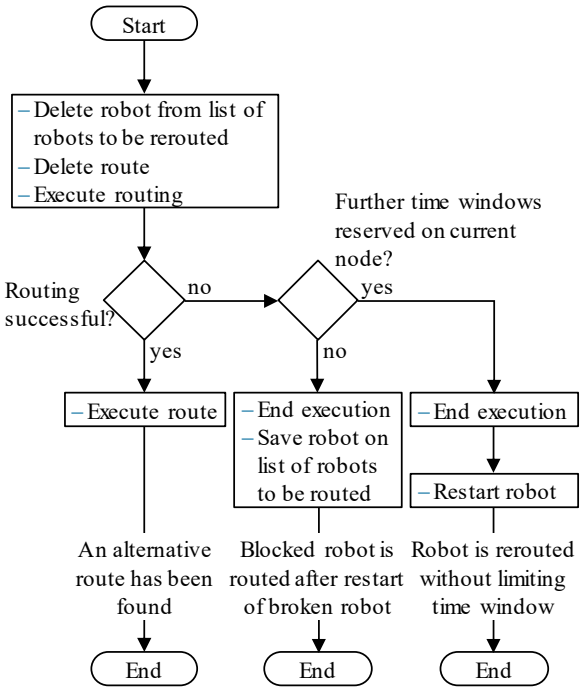

Figure 12: Procedure: reroute

SIMULATION STUDY

In this section, we compare the previously described strategies by performing a simulation study, considering an RMFS.

Considered System

We apply the strategies to a fleet of robots moving within an RMFS with 336 storage locations, which are arranged in seven double rows divided by storage aisles. There are two cross-aisles located at one third and at two thirds of the aisle length. All aisles can be used for bi-directional traffic.
There are four picking areas with five picking places each arranged before the storage system. In front of these places, there are two uni-directional cross-aisles (see figure 13). A replenishment area, where empty racks are refilled, is located on the opposite side of the storage area.
The robots are dedicated to a picking zone and perform three different cycles to maintain the material flow between storage locations, picking area and replenishment area. For a more detailed description of the system, we refer to (Lienert et al. 2018).
We implemented the RMFS using the discrete event simulation environment *Tecnomatix Plant Simulation*. Figure 13 shows a screenshot of the simulation model, comprising the picking area and part of the storage aisles. Note that loaded robots must use aisles and cross-aisles whereas unloaded robots are also free to use storage locations for navigation towards their destination. Each rectangle represents a node in the layout graph that is used for the time window routing.

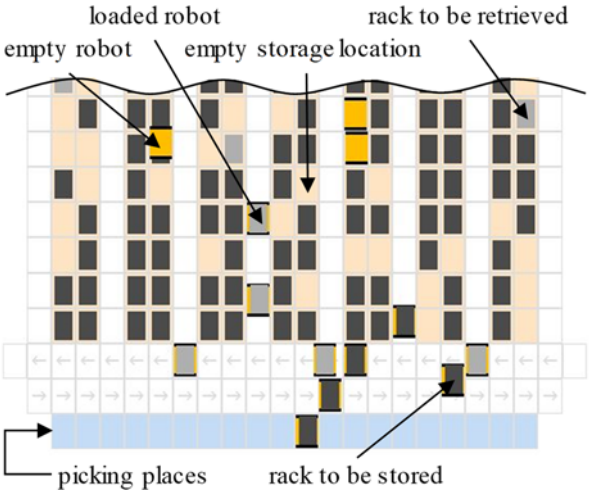

Figure 13: Screenshot of the simulation model

Parameter Settings

In a first experiment, availability is set to 100 percent – so no downtimes occur – so as to determine maximum system throughput. Subsequently, availability is set to 99 percent and MTTR to three minutes.
Failures are generated based on these parameters by the software using an *Erlang* distribution for the duration of a failure and an exponential distribution for the time between them. As failures are generated for each robot independently, several robots can be affected by a breakdown at the same time.
We vary the number of robots, starting with four robots (one for each picking zone), and going up to 60 robots working in the system in steps of four, and repeat the experiments for each strategy. All the remaining parameters, such as robot's acceleration and maximum speed, remain the same. Simulation time is set to 72 hours. No warm-up time is taken into account. We conduct five replications for each parameter setting.

Results

The chart in figure 14 shows the throughput measured in cycles per hour. Regarding the idealized system, performance scales (in a quasi-linear manner) with the number of robots until saturation is reached. The curves for the idealized systems *Reroute* and *Restart* show a small "knee" between 16 and 20 robots. As each robot is assigned to a certain picking zone, with 20 robots, the number of robots per picking zone equals the number of picking places, and a different strategy for the supply of the picking zone is used (see Lienert et al. 2018).
The more robots are operating in the system, the clearer the differences between the strategies. As expected, *Pause* shows the worst performance. Throughput peaks at just 20 robots, and then starts decreasing. With an increasing number of robots, the probability of a breakdown grows and the whole fleet is prevented from working. *Ignore* shows a similar behaviour, although throughput is slightly higher. The more robots operating in the system, the more robots affected by a breakdown.

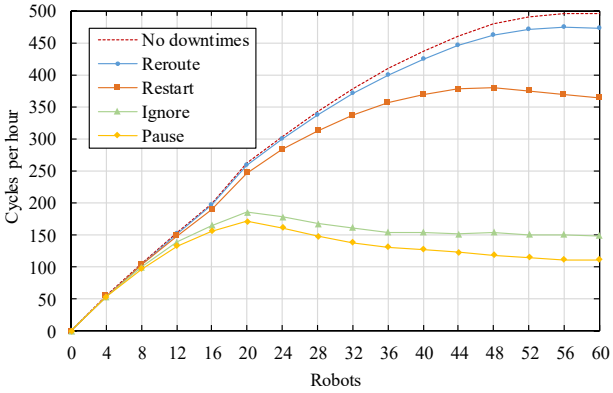

Figure 14: Throughput reached by the strategies

In the worst-case scenario, all robots have to interrupt the execution of their routes in order to maintain the node's crossing order.
Restart reaches a significantly higher throughput than *Ignore*, which peaks with 48 robots before throughput decreases slightly. However, the highest throughput is reached by *Reroute*. The curve is similar to the one for the idealized system, but saturation is reached at a lower number of cycles.
In a second experiment, we vary the availability of the robots from 97.5 to 99.5 percent in steps of 0.5 percent, and repeat the experiments applying only *Reroute*. The chart in figure 15 shows the corresponding throughput.

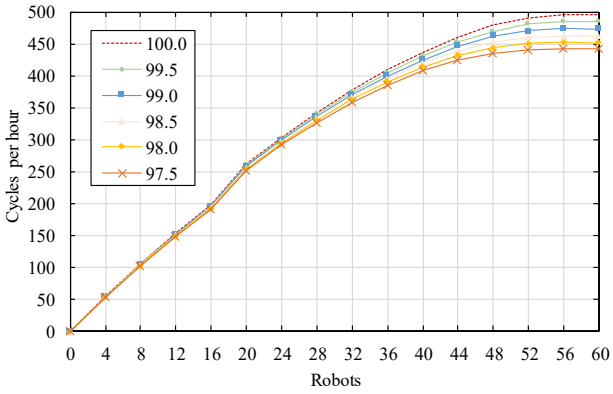

Figure 15: Throughput for different availabilities

As can be observed, all curves are similar and differ mainly in the maximum throughput that can be reached.
The chart in figure 16 provides a closer view of the loss of throughput compared to the idealized system without downtimes. In the case of high availability (99.5 percent), the loss of throughput remains relatively small and closed to the lower bound of 0.5 percent. However, with an increasing number of robots, the loss of throughput generated by the failures also increases. With 60 robots, the loss of throughput is above 2 percent.
The more evident this behaviour is, the lower the availability. With an availability of 97.5 percent, loss of throughput reaches as high as 11 percent.

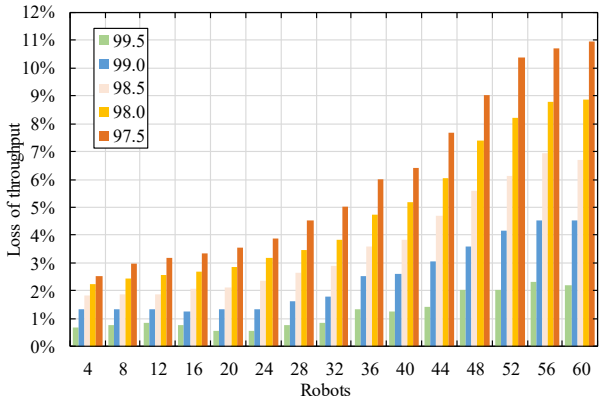

Figure 16: Loss of throughput for different availabilities

CONCLUSION

In this contribution, we considered mobile-robot-based warehouses. We presented four different strategies for dealing with failures that are based on the time window routing method. We applied these strategies to an RMFS and conducted a simulation study to compare the performance. Rerouting robots showed the best results and should be used for further investigations. Furthermore, we showed how throughput is affected for different levels of availability.

By way of next research, we suggest introducing safety corridors for accessing broken robots. In this respect, two questions have to be answered. First: Which path should be taken by an operator to access the broken robot? And second: How can the corridor be established in order to guarantee that no robot will enter that corridor while an operator is accessing the broken robot?

One option is routing an operator through the system applying the time window routing as well. However, all reservations must be endless to ensure no robot enters the corridor as long as the operator moves within the system.

REFERENCES

Azadeh, K.; de Koster, R. and Roy, D., 2018, "Robotized and Automated Warehouse Systems: Review and Recent Developments." Available at *SSRN*.

Boysen, N.; Briskorn, D. and Emde, S., 2017, "Parts-to-picker based order processing in a rack-moving mobile robots environment." *European Journal of Operational Research* 262, No.2, 550-562.

Boysen, N; de Koster, R. and Weidinger, F., 2018, "Path planning for robotic mobile fulfillment systems.", "Warehousing in the e-commerce era: A survey." *European Journal of Operational Research,* in press.

Busacker, T. 2005. *Steigerung der Flughafen-Kapazität durch Modellierung und Optimierung von Flughafen-Boden-Rollverkehr – Ein Beitrag zu einem künftigen Rollführungssystem*. Dissertation. Technische Universität Berlin.

Havězda, J.; Rybecký, T.; Kulich, M. and Přeučil, L., 2018, "Context-Aware Route Planning for Automated Warehouses." In *Proceedings of the 21st International Conference on Intelligent Transportation Systems*, 2955-2960.

Kim C. W. and Tanchoco J. M. A., 1991, "Conflict-free shortest-time bi-directional AGV routing." *International Journal of Production Research* 29, No.12, 2377-2391.

Lienert, T. and Fottner, J., 2017, "No more deadlocks – applying the time window routing method to shuttle systems." In *Proceedings of the 31st European Conference on Modelling and Simulation*, 169-175.

Lienert, T; Wenzler, F. and Fottner, J., 2018, "Robust integration of acceleration and deceleration processes into the time window routing method." In *Proceedings of the 9th International Scientific Symposium on Logistics*, 66-86.

Lienert, T.; Staab, T.; Ludwig, C. and Fottner, J., 2018, "Simulation-based Performance Analysis in Robotic Mobile Fulfilment Systems." In *Proceedings of the 8th International Conference on Simulation and Modeling Methodologies, Technologies and Applications,* 383-390.

Maza, S. and Castagna, P., 2005, "A performance-based structural policy for conflict-free routing of bi-directional automated guided vehicles." *Computers in Industry* 56, No.7, 719-733.

Mershformann, M.; Xie, L.; Erdmann, D., 2018, "Path planning for robotic mobile fulfillment systems", arXiv preprint arXiv:1706.09347

Roy, D.; Krishnamurthy, A.; Heragu, S. and Malmborg, C., 2015, "Queuing models to analyze dwell-point and cross-aisle location in autonomous vehicle-based warehouse systems", *European Journal of Operational Research* 242, No. 1, 72-87.

Stenzel, B. 2008. *Online Disjoint Vehicle Routing with Application to AGV Routing*. Dissertation. Technische Universität Berlin.

Tappia, E.; Roy, D. de Koster, R. and Melacini, M., 2018, "Modeling, Analysis, and Design Insights for Shuttle-Based Compact Storage Systems", *Transportation Science* 51, No.1

Ter Mors, A. W. 2010. *The world according to MARP.* Dissertation. Delft University of Technology

Yuan, Z.; Gong, Y.Y., 2017, "Bot-In-Time Delivery for Robotic Mobile Fulfillment Systems." *IEEE Transactions on Engineering and Management* 64, No.1, 83-93

Zou, B.; Xu, X.; Gong, Y.Y., de Koster, R., 2017, "Evaluating battery charging and swapping strategies in a robotic mobile fulfillment system." *European Journal of Operational Research* 267, No. 2, 733-753.

THOMAS LIENERT has been working as a research assistant at the Chair of Materials Handling, Material Flow and Logistics, Technical University of Munich, since 2014. His research deals with the simulation of mobile-robot-based warehouses. His email address is: `thomas.lienert@tum.de`.

LUDWIG STIGLER graduated from the Technical University of Munich. His master's thesis deals with failure-handling strategies for mobile robots. His email address is: `ludwig.stigler@tum.de`.

JOHANNES FOTTNER is professor and head of the Chair of Materials Handling, Material flow, Logistics at the Technical University of Munich. His email address is: `j.fottner@tum.de`.