Proceedings of the 2011 Winter Simulation Conference
S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, eds.

SIMULATION AIDED, KNOWLEDGE BASED ROUTING FOR AGVS IN A DISTRIBUTION WAREHOUSE

Alexander Klaas
Christoph Laroque
Wilhelm Dangelmaier

Business Computing, especially CIM
Heinz Nixdorf Institute
University of Paderborn
Fürstenallee 11
33102 Paderborn, GERMANY

Matthias Fischer

Algorithms and Complexity
Heinz Nixdorf Institute and
Department of Computer Science
University of Paderborn
Fürstenallee 11
33102 Paderborn, GERMANY

ABSTRACT

Traditional routing algorithms for real world AGV systems in warehouses compute static paths, which can only be adjusted to a limited degree in the event of unplanned disturbances. In our approach, we aim for a higher reactivity in such events and plan small steps of a path incrementally. The current traffic situation and also up to date time constraints for each AGV can then be considered. We compute each step in real time based on empirical data stored in a knowledge base. It contains information covering a broad temporal horizon of the system to prevent costly decisions that may occur when only considering short term consequences. The knowledge is gathered through machine learning from the results of multiple experiments in a discrete event simulation during preprocessing. We implemented and experimentally evaluated the algorithm in a test scenario and achieve a natural robustness against delays and failures.

1 INTRODUCTION

Mobile robots used to transport materials are called AGVs (automated guided vehicles). They increase efficiency and reduce costs across a wide area of applications, including distribution logistics. Operational control of AGVs in a distribution warehouse encompasses the task of routing the vehicles, i.e., navigating them through the warehouse, efficiently to optimize storage and retrieval processes.

The algorithms for routing AGVs in a real world warehouse have many requirements. We consider the vehicles to operate in a dynamic, lively environment where many unplanned changes happen constantly. There are obstacles in motion, like other vehicles, which may not move exactly predictably, failures may occur and also external influences like early arrivals of delivery trucks affect the system. There is no fixed set of tasks that need to be fulfilled, as jobs are continuously added and may be modified.

Most existing algorithms assume a controlled environment however. Paths for an AGV to a given target are pre-computed and cannot, or only to a limited degree, be adjusted once vehicles are underway, should the need arise due to one of the above mentioned changes (Qiu et al. 2002).

In order to improve the ability to deal with such dynamics, we present a routing algorithm that plans small, incremental steps of the path. In each step, the current traffic situation is considered. This way, any unplanned events can naturally be dealt with as no path is pre-computed.

However, contemplating only a local horizon does have its drawbacks, as we illustrate in Figure 1 with a simple example. Two vehicles opposite of each other seek targets that lie behind a narrow corridor from their point of view. When only a small step is planned in advance, both vehicles would move straight into the corridor, as this would reduce their respective distance to the target the most, and therefore is the most

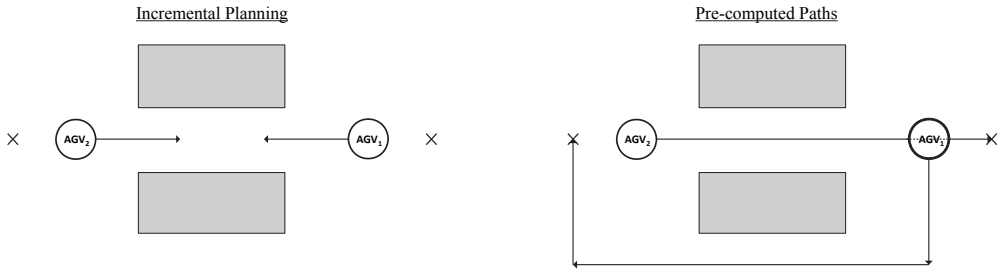

Figure 1: Incremental planning considers only a local horizon, which may lead to bad results.

beneficial action in the short term. This leads to a head-on conflict however, and turns out to be a costly decision in the long term. The conflict can be avoided by pre-computing complete paths, which means planning over a global horizon.

Existing routing algorithms that plan incrementally often use a simple heuristic, such as historic utilization of a floor area (Taghaboni-Dutta and Tanchoco 1995), to estimate future effects, but they can only consider a very limited amount of information.

By combining material flow simulation and its experimental results with machine learning, we have developed a novel incremental routing algorithm without the above mentioned drawbacks.

- Our core contribution is the development of a routing algorithm which is able to react well to dynamic changes in the warehouse, while making more informed decisions than other incremental planning algorithms, and still computing each step suitably fast for real time operation.
  We achieve this result by basing the route computation step not on a simple heuristic, but on a huge amount of processed empirical data. The data includes knowledge about long term consequences of each alternative next step and is gathered in a forward simulation of the system. We therefore use simulation of the system as a source of knowledge for the algorithm. By storing offline simulation results in a knowledge base that is queried during online operation, we reach the required low computation times even though a lot of information goes into each decision.
- The vehicles move freely in 2D space and are not bound to predetermined routes. This means that all available floor space can be used by the vehicles.
- No fixed path or zone definition is required. Given the warehouse floor layout, all computation is done automatically without requiring any adjustment or manual work.
- In addition to routing each AGV to its target as quickly as possible, deadlines for the AGVs to reach the target can be considered. AGVs that may miss their deadline can be given priority over other AGVs, for example.

The algorithm is presented in detail in Section 3, which also includes a proof of termination, i.e., iterated steps will result in a deadlock-free path to the target for all vehicles. In Section 2, we present motion planning and AGV routing theory and classify our approach. We implemented the algorithm in a discrete event simulator to demonstrate its functionality and experimentally evaluate its performance. We created a model of a distribution warehouse and compared our approach under sample load scenarios to appropriate alternative configurations in respect to several key performance measures, such as average task processing time. Details of the implementation and evaluation results are given in Section 4.

2 RELATED WORK

Much research has been done in the field of AGV routing and various solutions have been developed for a wide variety of different routing problems. They are uniquely characterized by parameters such as the application the AGVs are used for, vehicle capabilities and details of the physical space they move in. The

generic models created in the field of Motion Planning theory (Latombe 1991, LaValle 2006) are useful to understand and classify each AGV routing problem.

We consider vehicle movement without rotations on a 2D floor with obstacles, which is also known as the Piano Mover's Problem. One of the most advanced and fastest algorithms for the problem of finding the shortest path between two points in 2D space is the so called framed quadtree method (Chen, Szczerba, and Uhran 1997). However, as we control multiple vehicles simultaneously, the method shall only be called as a subroutine. As centralized path computation for all vehicles has been shown to have exponential runtime in respect to the number of vehicles (Latombe 1991), all practical algorithms implement decoupled planning (LaValle 2006), i.e., independent path computation for each vehicle with some coordination mechanism that allows vehicles to consider each other as moving obstacles and avoid conflicts. A wide variety of such mechanisms have been employed (for an overview, see Qiu et al. (2002); Krishnamurthy, Batta, and Karwan (1993) detail a method that has been improved on in Oboth, Batta, and Karwan (1999)). One popular mechanism are time windows (Kim and Tanchoco 1991; Desrosiers et al. 1986; Möhring et al. 2004), which are particularly efficient.

Time window based methods are *static* solutions to the routing problem, referring to the time point at which path computation occurs. Given a problem instance (a target position for each AGV to reach from its current position), paths are computed once before the vehicles start moving and are not changed during operation, hence remain static. While this allows the computation to find a globally optimal path to the target, changes in the system cannot be considered (Maza and Castagna 2005). It is assumed that all vehicles follow the path as planned. Unplanned disturbances can only be handled under specific circumstances that the plan can be adjusted to (Stenzel 2008, ter Mors and Witteveen 2009).

Dynamic, or *incremental* methods perpetually compute successive small steps towards a target instead of the whole path. Once a vehicle reaches the end of the current step, the next step is determined. The current system state can be considered in the computation and dynamic routing algorithms are generally well suited to react to changes and interruptions (Maza and Castagna 2005). To avoid conflicts between vehicles, AGVs are typically re-routed in a step (Narasimhan 1999). A major drawback is that the global optimality of the complete path cannot be guaranteed as only local optimization is done. Taghaboni-Dutta and Tanchoco (1995), Sakakibara, Fukui, and Nishikawa (2008) and Fanti et al. (1997) detail further examples of dynamic routing algorithms.

The information that goes into the computation of an incremental step is a distinct feature of a dynamic routing algorithm. Heuristics such as historic utilization have been used to evaluate different alternative next steps and determine a solution that is not only locally beneficial, but leads to a globally valuable path (Taghaboni-Dutta and Tanchoco 1995). So far, only relatively simple heuristics that only include a small fraction of the relevant available information about the system state have been used. Artificial intelligence methods have been applied to the problem in Sakakibara, Fukui, and Nishikawa (2008) and Jeon, Kim, and Kopfer (2011) (the latter method using static paths however) by examining a large number of traffic situations in a simulation and processing the analysis results. In the related field of job scheduling, similar ideas involving the use simulation and artificial intelligence methods have been applied (Aufenanger et al. 2008).

For our application, we propose a dynamic routing algorithm that bases the computation of each step on extensive information about encountered system states. We aim to combine the advantages of dynamic methods (reactivity and robustness) with those of static methods (global optimality) through the use of artificial intelligence.

3 DYNAMIC KNOWLEDGE-BASED ROUTING

We present a solution to the problem of routing multiple AGVs within a distribution warehouse between pickup and drop off points under consideration of a dynamic environment. We first provide an overview of how our algorithm operates in Section 3.1 and break down its individual components in Section 3.2 and Section 3.3.

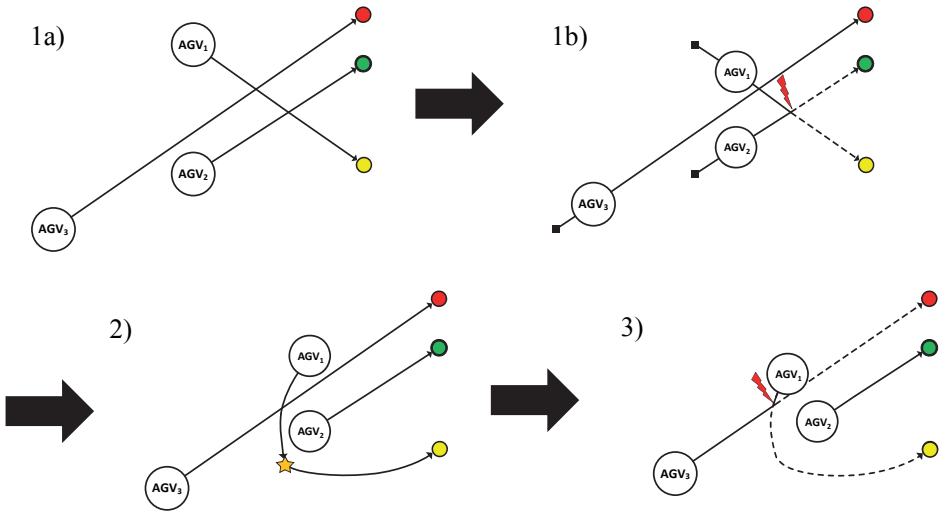

Figure 2: Example illustrating the principal operation of the routing algorithm.

3.1 Principal Operation

Given an initial position and goal for each AGV, the algorithm incrementally computes path segments for the vehicles to follow. The computation consists of three steps as described below and illustrated conceptually in Figure 2:

1. a) For each AGV, compute the euclidean shortest path from its current position to its goal using the framed quadtree method (Chen, Szczerba, and Uhran 1997). Each path is cut off just at the point where it would first cross another AGV's path and a collision would occur (highlighted by a lighting bolt in Figure 2). b) The AGVs move along this path with constant velocity until they reach their cut off point, minus an offset.

2. The collision is avoided by re-routing the AGVs according to one of a limited set of defined re-routing rules. In Figure 2, AGV_1 seeks an evasion point so that AGV_2 can continue following its shortest path. Which rule to use is determined based on the current system state by accessing the knowledge base. It contains information about situations previously encountered in the offline simulation and for which the optimal rule is given.

3. After successful re-routing, go to step 1, i.e., compute the shortest paths to the targets from the AGVs' new positions. In Figure 2, the paths of AGV_1 and AGV_3 cross and the process is repeated.

Once all AGVs have reached their goals, the algorithm terminates.

Our algorithm is dynamic, as only incremental steps (until the next predicted conflict) are planned and therefore, the re-routing procedure can be based on the current system state. Dynamic processes like additions or modifications to the task set, delays in AGV motion or failures can naturally be handled. We simulate the system a priori and solve conflicts between AGVs by evaluating the effects of applying each different re-routing rule. We are able to consider not only local implications of the re-routing step (i.e., the situation of the two involved AGVs directly after the diversion), but rather global and long term effects as we look ahead in the simulation by a significant amount of time.

The knowledge gathered allows the algorithm to determine the action to take when encountering a new situation, based on what proved to be a good action in previous, similar situations. It is therefore

necessary to provide the system with an appropriate amount of relevant example situations. The exact number required can be determined by measuring the performance with increasing quantity, until no further improvement can be seen. We store these situations, along with the corresponding optimal rule to apply, and build up a knowledge base by learning from these experiences using methods of artificial intelligence, specifically inductive learning algorithms. Further details concerning the knowledge base are laid out in Section 3.3.

Figure 3 gives a broad overview of how the different components of the system are interconnected. The central AGV Routing component manages the vehicles A_i and repeatedly calls the Shortest Path Search, i.e., the framed quadtree method, to compute paths between two points, like between an AGV's current position and its target, or an evasion point. When vehicles reach the cut off points, calculated by the integrated collision detection algorithm, it accesses the knowledge base in step 2 of the algorithm to determine the rule to re-route the vehicles. The knowledge base contains the processed data from the forward simulation.

Access to the knowledge base during online operation takes very little computation time, which is crucial for the use of the algorithm in a real-time system, as running the forward simulation in each instance would not be feasible. The re-routing rule must be determined quickly enough so that vehicle motion is not interrupted. The computation time of each round of the algorithm also includes the time needed by the framed quadtree algorithm (step 1), determining collision points and applying the re-routing rule.

We are able to show that iteratively executing the algorithm will result in all vehicles reaching their targets, i.e., the algorithm terminates and deadlocks are avoided, for two vehicles ($N \leq 2$) as each conflict between two vehicles is treated independently. We have implemented some measures to resolve conflicts between more than two vehicles, for example in some cases one vehicle waits for the other two to resolve their conflict. These cases have been found to occur very rarely and do not impede the algorithm's functionality in practice.

We design each re-routing rule to fulfill the following requirement: at the end of the re-routing step, at time point $x+1$, the distance of at least one vehicle i to its target, $d_{i,x+1}$, has been reduced by at least some constant ϵ, while the distances for all other vehicles do not increase. Therefore, the sum of all distances decreases between each time point:

$$\sum_{i=1}^{N} d_{i,x} > \sum_{i=1}^{N} d_{i,x+1} + \epsilon \text{ for some constant } \epsilon \tag{1}$$

Due to the decrease in each time step, expressed in (1), the sum of all distances converges with increasing time x and will become zero:

$$\lim_{x \to \infty} (\sum_{i=1}^{N} d_{i,x}) = 0 \Rightarrow \exists x : d_{i,x} = 0 \ \forall \ i \tag{2}$$

If at each time point x, a re-routing rule can successfully be applied with the specified requirement, this succession of applying re-routing rules will lead all vehicles to their target, as (2) shows due to the distances all becoming zero. The ability to execute the re-routing procedure and fulfill the requirement will be analyzed for each rule in the following section along with a detailed presentation.

3.2 Rules for Re-routing

The re-routing rules represent high level choices made by the algorithm and they must specify how both vehicles (AGV_1 and AGV_2) involved in a conflict continue their path computation and resolve the situation. We follow an idea from (Sakakibara, Fukui, and Nishikawa 2008) and provide three rules corresponding to different degrees of prioritization, adapted to free vehicle motion in 2D. For each rule, we examine the computation cost and how to fulfill the requirement that the sum of the distances decreases, as mentioned in Section 3.1.

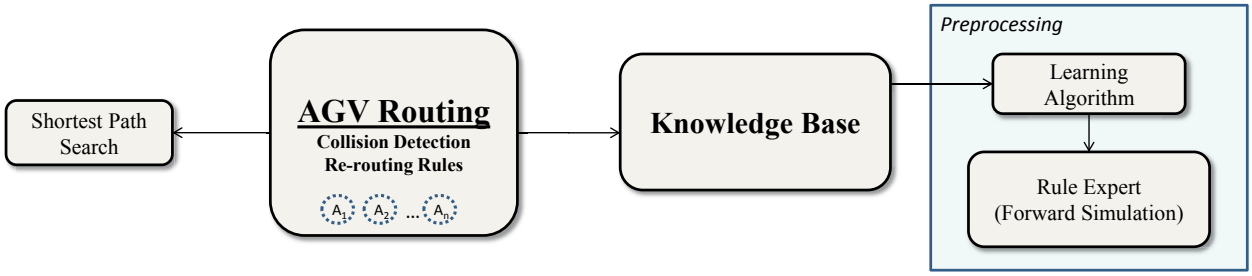

Figure 3: A broad overview of the algorithm.

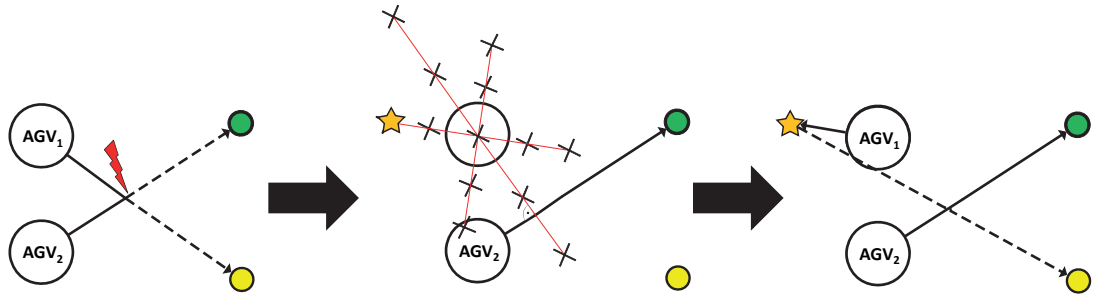

Figure 4: Application of Rule 2: AGV_2 is given priority, AGV_1 computes a point to evade to (indicated by a star). AGV_2 continues on its original path, while AGV_1 evades and then resumes moving towards its target.

Rule 1: AGV_1 simply remains stationary for a certain amount of time, while AGV_2 considers it a static obstacle and resumes moving, if AGV_1 does not block its path. AGV_1 waits as long as its future path, after starting up again, does not cross AGV_2's path. As AGV_1 keeps its distance to its goal and does not interfere with AGV_2, the sum of the distances is reduced. Rule 1 can be applied with reversed roles of the vehicles as well. The rule is realized by adding a bounding box of AGV_1 to the set of static obstacles, creating a new framed quadtree from it, and perform a search on it.

Rule 2: AGV_1 evades by seeking an evasion point so that AGV_2 can simply follow its shortest path to its target (see Figure 4). The evasion point is only accepted if AGV_1 is able to move to it, and from there to its target(perhaps after waiting at the point) without again conflicting with AGV_2 before having reduced its distance to its target. This shows that Rule 2 fulfills the requirement for re-routing rules. As checking an evasion point involves multiple calls to the framed quadtree algorithm, we define a search strategy for evasion points. As shown in Figure 4, points in varying distance of AGV_1 on a line perpendicular to AGV_2's path are evaluated, which correspond to a natural sidestep motion. If they are not accepted, other points on radial rays are checked.

Rule 3: Both vehicles evade each other (see Figure 5). They try to find paths to their targets without conflicting before having reduced their distances to the targets. The rule is implemented by first searching in a new framed quadtree that treats the respective other vehicle as static obstacles, similar to Rule 1. If this fails, a search pattern on radial rays, similar to Rule 2 is applied, and both vehicles search for valid evasion points in parallel. The strategy is to construct the rays in a way that the vehicles move away from each other. As no vehicle is given priority, only one instance of Rule 3 exists.

It is noted that Rule 2 displays the highest priority for the respective prioritized vehicle, as it is guaranteed to follow its shortest path. Rule 1 corresponds to modest prioritization, while Rule 3 fairly resolves the conflict.

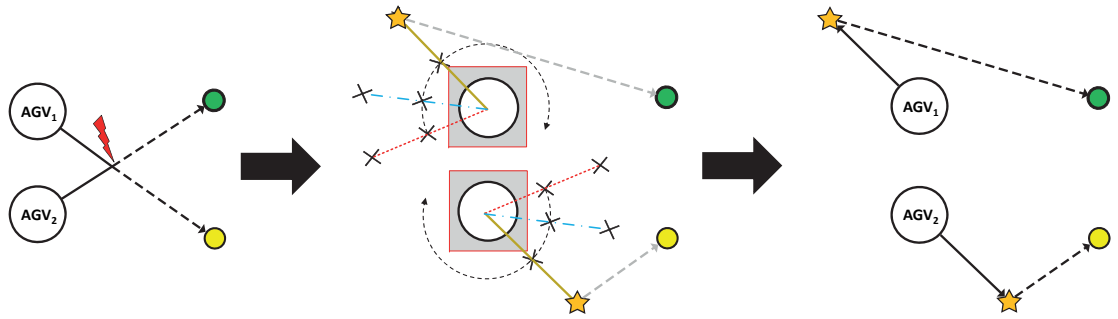

Figure 5: Application of Rule 3: Both AGVs evade each other by treating each other as obstacles at first, which fails in this case, and then computing evade points in parallel.

It is now clear that the routing algorithm chooses between five discrete instances of the three re-routing rules to resolve each conflict, while every instance specifies the exact path for the two involved AGVs. The knowledge base must then map every situation to one of these instances.

3.3 Knowledge Base and Information Acquisition Through Simulation

In step 2 of the routing algorithm, we need to determine the "best" rule in respect to long term benefit, depending on the current traffic situation. Key issues in the design of the knowledge base are defining a situation, how to measure the long term benefit, and how to gather and store all accumulated knowledge.

A situation definition needs to contain all system parameters relevant to making the decision of which re-routing rule to apply. We therefore include the following attributes:

- The positions $(p_{1_x}, p_{1_y}), (p_{2_x}, p_{2_y})$ of both vehicles involved in the conflict. They are in absolute coordinates as the location within the warehouse (as neighboring obstacles have an effect on how to resolute the conflict) is expected to be relevant.
- Target positions $(z_{1_x}, z_{1_y}), (z_{2_x}, z_{2_y})$ of both vehicles.
- Deadlines d_1, d_2 of both vehicles, which are needed to prioritize a vehicle with a deadline in the near future, should this be beneficial.
- The positions (p_{i_x}, p_{i_y}) of all other vehicles in the warehouse. This information allows to take future collisions with nearby AGVs into account.

A situation is then defined by the product of all attributes.

During preprocessing, we generate the example situations, based on application load scenarios. We measure the long term benefit of each possible action by simulating the system forward, in order to determine the best one. The module that computes the best action to a situation is called the Rule Expert, as depicted in Figure 3.

We simulate the different scenarios after applying each of the five rule instances, described in Section 3.2. At each consecutive conflict, all five rules are applied again, so that all possible "futures" are explored. The problem can be seen as a graph search (we specifically use uniform cost search (Russell and Norvig 2009)), with each node corresponding to a situation. To reduce the computation time of the exploration, we employ a heuristic to determine which subtrees in the graph can safely be disregarded.

In any graph search, the nodes are explored until a goal node is found, which we define as a situation where each AGV is at its target position. We define a cost for each node as the time needed to get to that state from the root node, and add a penalty for each vehicle that missed its deadline in that situation.

Looking back at the original situation, we then evaluate each of the five rule instances by looking at the long term consequences. Once a goal node has been found, we can look at which rule is the first on the path to that goal node, i.e., which rule lead to the situation. A rule that results in late arrivals is deemed

to be bad, and the rule that leads to the goal node with the fewest cost is determined to be the best one in that situation.

By storing the results to the so called training set, we will later, during online routing, base the decision for new situations on them. We exploit the natural observation that an action, which proved to perform well in a similar situation, will typically also be good in the new one. The training set is processed by a learning algorithm that is designed to extract patterns and knowledge from the data, and outputs a trained knowledge base. The process is also known as inductive learning (Russell and Norvig 2009). New situations can then classified, or assigned to a group of known situations using a measure of similarity. Each learning algorithm employs different approaches to measuring similarity however.

In addition to learning from the preprocessed training set, we are also able to learn from decisions made online in the real system, as they constitute further examples and can simply be added to the training set.

We use two different learning algorithms that have found wide spread application. The first is called C4.5 and builds up a "decision tree" (Mitchell 1997). The second outputs a trained "artificial neural network", it is called the back propagation algorithm (Hassoun 1995). Both can be queried, i.e., determine a rule given a situation, very quickly. Specifically, the computation time is independent of the amount of training examples, it only depends on the fixed amount of attributes in the situation definition.

The knowledge stored in the trained models is based on what the Rule Expert computes. Through forward simulation, it is able to evaluate a rule based on its long term effects, and involves expensive computations of different scenarios. Yet the computation results can quickly be accessed during online decision making.

4 EVALUATION

We have implemented our routing algorithm in the discrete event simulator d3fact (Dangelmaier and Laroque 2007) as a means to validate and evaluate our approach. We used the artificial intelligence library "Weka 3.6" (Witten and Frank 2005) as it contains suitable implementations for the knowledge base and learning algorithm (as in Figure 3). During preprocessing, d3fact outputs the generated training set to a file that Weka processes, and the resulting trained models are then saved to hard disk. During simulation of the online system, the trained models can easily be accessed from d3fact through Weka interfaces, as both are written in the Java programming language.

We created a model of an exemplary distribution warehouse as a test setting for the algorithm. As illustrated in Figure 6, it consists of different types of rectangular storage racks, which are static obstacles for the AGVs. The racks form hierarchical zones that resemble the structure of typical real world warehouses. From the input load ramps, goods are first transported to a nearby commissioning zone, from there to a block storage area for temporary buffering, and later to a high capacity pallet rack for long term storage. The same process happens in reversed order for retrieval of goods to fulfill request at the output loading ramps. An additional mechanism is needed to avoid overcrowded loading ramps: we allow only one AGV to work at the same ramp simultaneously. The remaining vehicles may process tasks in other zones.

The AGVs are assigned tasks to move materials between defined pick up and delivery points in the lanes between the racks of the different zones. AGVs move to the points to access or place goods in the racks as forklift trucks would. We model the process of accessing the racks and pickup up or dropping off goods by a time delay. The space between two pallet racks is wide enough for just one vehicle, which has a diameter of $2,50m$. We assumed the standardized base area of a pallet ($1,20m \times 0,80m$) and typical warehouse capacities when creating the model, which has an area of $32,61m \times 46,60m$.

Given a random initial inventory in the warehouse, we simulate the dynamic processes. Delivery trucks arrive at a loading ramp and deliver or retrieve a set of goods (here 17 out of 9 categories of materials), which implies a set of tasks for the AGVs, of which there are three in the setting. They are required to transport the delivered goods to the storage racks, or transport requested goods to the output loading

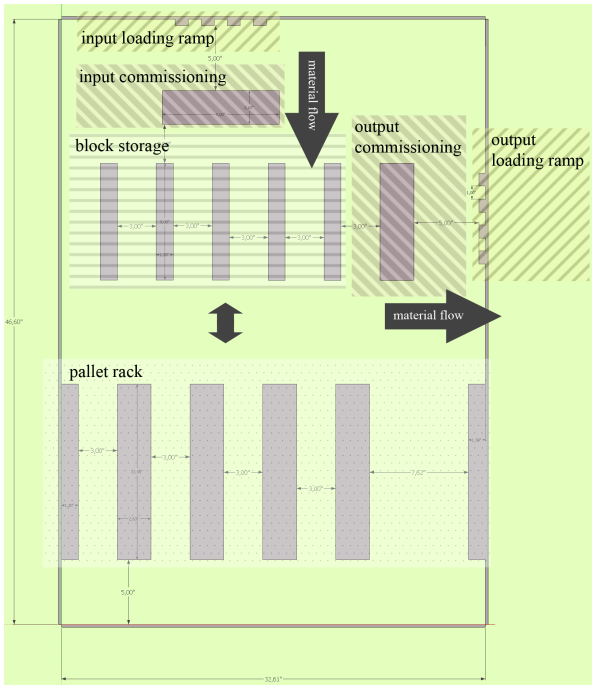

Figure 6: Layout of the test case warehouse. There are distinct, hierarchical zones for storage and retrieval of goods that define a direction of material flow.

ramp. We use simple strategies to determine how tasks are assigned to vehicles, as we aim for an isolated evaluation of the routing algorithm.

We use the simulation for two purposes: As a benchmark for the routing algorithm, and also to generate the training set during preprocessing. We run the simulation repeatedly with a load configuration that we anticipate for later operation: 80 to 140 trucks, evenly distributed over a time period of 1.5 million time units. We save any encountered situations, of which we generated approximately 45000, along with the result of the Rule Expert search (see Section 3.3).

For the application, the primary optimization goal of the routing algorithm is to serve the set of delivery and retrieval tasks within a specified time that the delivery trucks demands from the warehouse. Following this thought, we measure two metrics in an experimental simulation run to quantify the routing algorithm's performance:

1. Average task completion time for critical tasks, which are tasks to move goods from or to loading ramps that directly affect the delivery truck's waiting time. The task completion time is considered relative to the task's deadline.
2. The average delay per delivery truck, which is the latest completion point each truck's demands.

We are not able to quantify the benefit that our approach has compared to other AGV routing algorithms, as no known one exists that exactly fulfills our requirements (e.g., dynamic task set, free roaming vehicles) or that could be easily adapted to suit our specific application. We rather demonstrate the algorithm's functionality and evaluate components of our algorithm by exchanging them with other alternative solutions.

Instead of accessing the knowledge base to determine a rule to resolve a conflict (in step 2 of the algorithm, see Section 3.1), we use simpler heuristics. The heuristics compute a value between -2 and $+2$, which corresponds to a rule as detailed in Section 3.3: A value of -2 equates to Rule 2 (strong prioritization of one vehicle), -1 to Rule 1 (modest prioritization), 0 to Rule 3 (no prioritization), with $+1$ and $+2$ meaning the respective rule with reversed roles. We have evaluated the following heuristics:

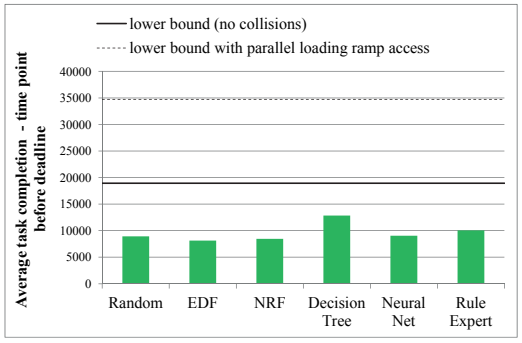

Figure 7: Comparison of rule decision, average task completion. Lower is better.

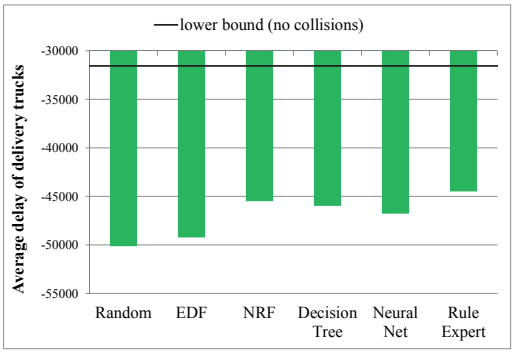

Figure 8: Comparison of rule decision, average truck delay. Lower is better.

- *Random*: Choose a rule randomly.
- *Earliest Deadline First* (*EDF*): Give priority to the AGV i with the earliest deadline d_i. Compute the rule according to the formula: $\frac{d_1 - d_2}{norm}$, where $norm$ is a normalization factor. We chose $norm = 500$, meaning that we assume a difference of 500 time units to make a significant impact.
- *Nearest Robot First* (*NRF*): The AGV i with the lowest straight line distance ($dist_i$) to the goal is chosen. Compute the rule according to the formula $\frac{dist_1 - dist_2}{dist_1 + dist_2} \cdot norm$. $norm$ normalizes the impact of one unit in space on the computation, we chose $norm = 20$.

Additionally, we computed lower bounds by disregarding collisions of vehicles: they are allowed to simply follow their shortest paths and move through each other. Obviously, no routing algorithm can achieve a better result. We also evaluate the Rule Expert by performing the forward simulation in each instance, which cannot be done in real time operation, but is possible to do in this experiment.

Figure 7 and Figure 8 show the results of experimental simulation runs. While the times for average task completion time show relatively little difference, the average delay of delivery trucks varies by about 10% between the best and worst alternative. It can be seen that the NRF heuristic and both knowledge bases perform significantly better than the remaining two heuristics. It is surprising however that consulting the Rule Expert in each new instance does not lead to better results. It explains why both knowledge bases do not score higher as well, as they process what the Rule Expert provides and have to rely on that data.

To support this assumption, we have evidence that both learning algorithms more closely resemble the Rule Expert's decisions with an increasing amount of training examples. We have measured the amount of situations that are mapped to the same action as the Rule Expert, and it grows from 42.2% (2500 training examples) to 52.4% (45000) for the decision tree knowledge base.

We were able to show that the system functions as was expected and is able to fulfill the dynamically demanded tasks for delivery trucks. The system was able to learn from artificial experiences, if not to the expected degree, as it relies on generated knowledge that cannot be guaranteed to be optimal.

5 CONCLUSION

In this paper we present our approach for the routing of free roaming AGVs in a distribution center. The free roaming ability allows for a more efficient usage of available floor space and does not require any prior path definition.

We introduce an improved decision making process for routing multiple vehicles and prove its strength by a first utilization within a given application area. With our approach, we gain a higher flexibility in reacting to occurring disturbance events during the execution by computing subparts of the entire transportation path. In order to overcome the limitations of existing approaches, we use a machine learning algorithm

to select the best situation-dependent decision rule, based on each single situational setting. For real-time application of our method, an initial training through a learning algorithm has to be done in advance. Here we use discrete, event-based simulation for the creation of multiple training situations with lots of decisions to be made, and again use the results of a simulation of different alternative decisions in order to judge the quality of each decision and save the best alternative to a training set. By supplying the results of the computations, we improve the training set until the created knowledge base works sufficiently. Then, we may again learn from our decisions in the real-time application and improve the training set with each decision. Moreover, within this approach, we are able to react to environmental changes and learn good decision rules for the changed underlying system.

The presented approach is today limited to the decision making process of the concrete routing of each AGV in the given setting, although there are multiple decisions to be taken for the application of a usable, knowledge-based floor control for a distribution system. During the next steps, we will therefore expand our approach to the selection of a specific combination of decision rules for routing, dispatching and selection of transport gates for delivering trucks, etc. This way, we will iteratively build up a knowledge base which is able to select a good combination of decision rules, depending on a specific overall situation for the distribution center. A possible answer for the breakdown of a specific AGV then might be the change of an arrival gate for a certain delivery truck, a change in the dispatching of orders and a following rearrangement of the priorities of the AGVs and their successive routing.

ACKNOWLEDGMENTS

This work was supported by the German Research Foundation (DFG) within the 3-year funded project "MMeAS - Modellbasierte Methoden zur echtzeitnahen Adaption und Steuerung von Distributionssystemen" (model-based methods for real-time adaption and control of distribution systems).

REFERENCES

Aufenanger, M., W. Dangelmaier, C. Laroque, and N. Rüngener. 2008, December. "Knowledge-based event control for flow-shops using simulation and rules". In *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R. R. Hill, L. Moench, O. Rose, T. Jefferson, and J. W. Fowler, 1952–1958. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Chen, D., R. Szczerba, and J. Uhran, J.J.. 1997, October. "A framed-quadtree approach for determining Euclidean shortest paths in a 2-D environment". *Robotics and Automation, IEEE Transactions on* 13 (5): 668 –681.

Dangelmaier, W., and C. Laroque. 2007, January. "d3FACT insight - Immersive Ablaufsimulation von richtungsoffenen und wahlweise zeitorientierten Materialflussmodellen". *Industrie Management* 2:73–76.

Desrosiers, J., F. Soumis, M. Desrochers, and M. SauvéGerad. 1986. "Methods for routing with time windows". *European Journal of Operational Research* 23 (2): 236 – 245.

Fanti, M. P., B. Maione, S. Mascolo, and B. Turchiano. 1997. "Event-Based Feedback Control for Deadlock Avoidance in Flexible Production Systems". *IEEE Transactions on Robotics & Automation* 13:347–363.

Hassoun, M. H. 1995. *Fundamentals of Artificial Neural Networks*. Cambridge, MA, USA: MIT Press.

Jeon, S., K. Kim, and H. Kopfer. 2011. "Routing automated guided vehicles in container terminals through the Q-learning technique". *Logistics Research* 3:19–27.

Kim, C. W., and J. M. A. Tanchoco. 1991. "Conflict-free shortest-time bidirectional AGV routeing". *International Journal of Production Research* 29 (12): 2377–2391.

Krishnamurthy, N. N., R. Batta, and M. H. Karwan. 1993. "Developing Conflict-Free Routes for Automated Guided Vehicles". *Operations Research* 41 (6): 1077–1090.

Latombe, J.-C. 1991. *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers.

LaValle, S. M. 2006. *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press.

Maza, S., and P. Castagna. 2005. "A performance-based structural policy for conflict-free routing of bi-directional automated guided vehicles". *Comput. Ind.* 56 (7): 719–733.

Mitchell, T. 1997, October. *Machine Learning*. 1st ed. McGraw-Hill Education (ISE Editions).

Möhring, R. H., E. Köhler, E. Gawrilow, and B. Stenzel. 2004. "Conflict-free Real-time AGV Routing". In *Operations Research Proceedings*, edited by H. Fleuren, D. Hertog, and P. Kort, 18–24. Berlin, Heidelberg: Springer-Verlag.

Narasimhan, R. 1999. "Routing automated guided vehicles in the presence of interruptions". *International Journal of Production Research* 37 (3): 653–681.

Oboth, C., R. Batta, and M. Karwan. 1999. "Dynamic conflict-free routing of automated guided vehicles". *International Journal of Production Research* 37 (9): 2003–2030.

Qiu, L., W.-J. Hsu, S.-Y. Huang, and H. Wang. 2002. "Scheduling and routing algorithms for AGVs: A survey". *International Journal of Production Research* 40 (3): 745–760.

Russell, S., and P. Norvig. 2009. *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, USA: Prentice Hall Press.

Sakakibara, K., Y. Fukui, and I. Nishikawa. 2008. "Genetics-Based Machine Learning Approach for Rule Acquisition in an AGV Transportation System". *Intelligent Systems Design and Applications, International Conference on* 3:115–120.

Stenzel, B. 2008. *Online Disjoint Vehicle Routing with Application to AGV Routing*. Ph. D. thesis, TU Berlin.

Taghaboni-Dutta, F., and J. M. A. Tanchoco. 1995. "Comparison of dynamic routeing techniques for automated guided vehicle system". *International Journal of Production Research* 33 (10): 2653–2669.

ter Mors, A., and C. Witteveen. 2009. "Plan Repair in Conflict-Free Routing". In *Next-Generation Applied Intelligence*, Volume 5579 of *Lecture Notes in Computer Science*, 46–55. Springer Berlin / Heidelberg.

Witten, I. H., and E. Frank. 2005, June. *Data Mining: Practical Machine Learning Tools and Techniques*. 2nd ed. Morgan Kaufmann series in data management systems. Morgan Kaufmann.

AUTHOR BIOGRAPHIES

ALEXANDER KLAAS studied computer science at the University of Paderborn, Germany. Since 2010 he is a research assistant at the Heinz Nixdorf Institute. His research interests are knowledge based methods, AGV control algorithms, online optimization methods and discrete event simulations. His email address is Alexander.Klaas@hni.upb.de.

CHRISTOPH LAROQUE studied business computing at the University of Paderborn, Germany. Since 2003 he has been a Ph.D. student at the graduate school of dynamic intelligent systems and, in 2007, received his Ph.D for his work on multi-user simulation. Since then he is team leader of the "simulation & digital factory" at the chair of Business Computing, esp. CIM. He is mainly interested in material flow simulation models and the "digital factory". His email is Christoph.Laroque@hni.upb.de.

MATTHIAS FISCHER studied computer science at the University of Paderborn, Germany. Since 1995 he has been a research assistant at the Heinz Nixdorf Institute. In 2005, he received a Ph.D. for his work on distributed virtual environments. His research interests are computer graphics, real-time rendering algorithms, and distributed computing. His email address is mafi@upb.de.

WILHELM DANGELMAIER studied Mechanical Engineering at the University of Stuttgart, Germany. In 1981, he became director and head of the Department for Corporate Planning and Control at the Fraunhofer Institute for Manufacturing. In 1991, Dr. Dangelmaier became Professor for Business Computing at the Heinz Nixdorf Institute. In 1996, he founded the Fraunhofer Center for Applied Logistics (Fraunhofer ALB). His email address is Wilhelm.Dangelmaier@hni.upb.de.