Proceedings of the 2013 Winter Simulation Conference
R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, eds.

SIMULATION AIDED, SELF-ADAPTING KNOWLEDGE BASED CONTROL OF MATERIAL HANDLING SYSTEMS

Alexander Klaas
Christoph Laroque
Hendrik Renken
Wilhelm Dangelmaier

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

ABSTRACT

Knowledge based methods have recently been applied to the control of material handling systems, specifically using simulation as a source of knowledge. Little research has been done however on ensuring a consistently high quality of the data generated by the simulation, especially under changing circumstances such as differing load patterns in the system. We propose a self-adapting control that is able to automatically generate knowledge according to current circumstances using a parametrized simulation model, which uses observed system parameters as input. The control automatically triggers generation when necessary, detects changes in the system and also proactively anticipates them, resulting in consistently high performance. For the problem of knowledge generation (determining an optimal control action to a given situation), we present a look ahead simulation method that considers uncertainties. We validated our approach in a real world material handling system, developed by Lödige Industries GmbH.

1 INTRODUCTION

The control of material flow in manufacturing and logistics systems is a highly dynamic and complex decision problem, characterized by an on-line nature, uncertainties and many interdependencies. Real world problems such as warehouse picking or AGV routing are typically NP-hard (infeasible to solve for real-time operation) and treated with heuristics, yet the solutions greatly influence the performance of the whole system (Tompkins 2010).

Specifically, whenever a decision needs to be taken in the system, e.g., a worker needs to be assigned to a pick task, a control situation arises which needs to be answered by the control in real-time. The solution may depend on circumstances in the system, e.g., the current status of other workers and the task location.

Recently, knowledge based methods have been applied to these problems that combine methods of artificial intelligence with simulation, including (Sakakibara, Fukui, and Nishikawa 2008; Kwak and Yih 2004; Aufenanger et al. 2008). Simulation is used in an off-line phase as a source of knowledge, as control situations can be solved and the results stored as training examples. The resulting data is analyzed using machine learning and can then quickly be used for on-line problem solving. As the theoretical amount of control situations is too high, only certain expected ones can be generated.

The performance of the control then greatly depends on the data provided by the simulation. We consider two main issues that impede performance:

- The occurring control situations in the system unexpectedly may not be included in the data, meaning that the control does not know how to decide.
- The simulation does not consider uncertainties in a controlled way when solving a control situation. This means that the provided answer may only perform well under the most likely future circumstances, but not others.

We aim to address these issues by presenting a self-adapting control that considers uncertainties in a novel simulation based training example generation method.

Self-adaptation of the control eliminates the problem of having to define an expected range of occurring control situations. If the control detects a situation for which it does not know the answer to, it is able to re-generate training data for that situation using a parametrized simulation model of the system. As this takes too long for a real-time answer, the control observes all occurring situations, forecasts likely future ones and proactively generates training data for the forecast future situations.

The parameters fed into the simulation models correspond to changing system parameters, such as the amount of workers present. This gives the control the ability to adapt to gradual changes of the system that may lead to different control situations. If for example a company grows and employs more workers, the control automatically reacts to this change, generates appropriate training data given the new circumstances, and also likely future ones with even more workers.

When solving a control situation, we use discrete event simulation to evaluate how the system behaves under each possible solution, combined with possible occurrances of uncertainties. This allows us to generate robust solutions that also perform well under circumstances such as a sudden failure of a machine.

We describe our concept generically and independent of a concrete application in Section 3. It is suitable for control of a wide range of logistics systems, where there are similar dynamic and complex problems across the industry.

We implemented the approach in our material flow simulator d^3fact and applied it to the control of a real world industrial material handling system, which is in the advanced planning stage at Lödige Industries GmbH in Germany (Section 4). We were able to demonstrate the benefit of our approach, as performance of the system increased by 14.2%.

2 RELATED WORK

Using current sensor data in simulation for operational control has been previously explored, for example in (Noack et al. 2011). The authors use a fast real-time simulation to determine the next action in a semiconducting manufacturing process, rather than using traditional problem solving algorithms or heuristics.

When simulating in an off-line phase, rather than on-line, runtime constraints are much weaker and thus has been done in works such as Sakakibara, Fukui, and Nishikawa (2008), Aufenanger et al. (2008), Kwak and Yih (2004), Klaas et al. (2011). The idea is always to simulate the controlled system in advance of application under the expected circumstances and consequently using the generated data in on-line decision making. Typically, the generated data is evaluated automatically using machine learning techniques such as decision trees.

If the system behaves differently than expected or changes over time, the approaches are not able to react. Thus, self-adapting control has been explored before in Müller-Schloer, Schmeck, and Ungerer (2011). The authors survey various frameworks that allow re-generation of data in a sandboxed simulation. One such framework has been applied in Prothmann et al. (2008) for the control of traffic lights. The system is able to react to actual traffic patterns that may have not been anticipated during design of the control. However, future stochastic events are not considered in a controlled way (i.e., can not be triggered in the simulation), and changes can only be reacted to a posteriori.

3 GENERIC CONCEPT

We present a knowledge based control concept that achieves the desired properties, namely self-adaptation after a detected or anticipated change, and consideration of stochastic future developments using look ahead simulation. As many aspects are independent of the specific system under control, we first lay out a generic concept. Section 3.1 presents an overview, while each component of the concept is detailed in Sections 3.2-3.5. We specify how to apply the concept to a specific system in Section 3.6.

3.1 Overview

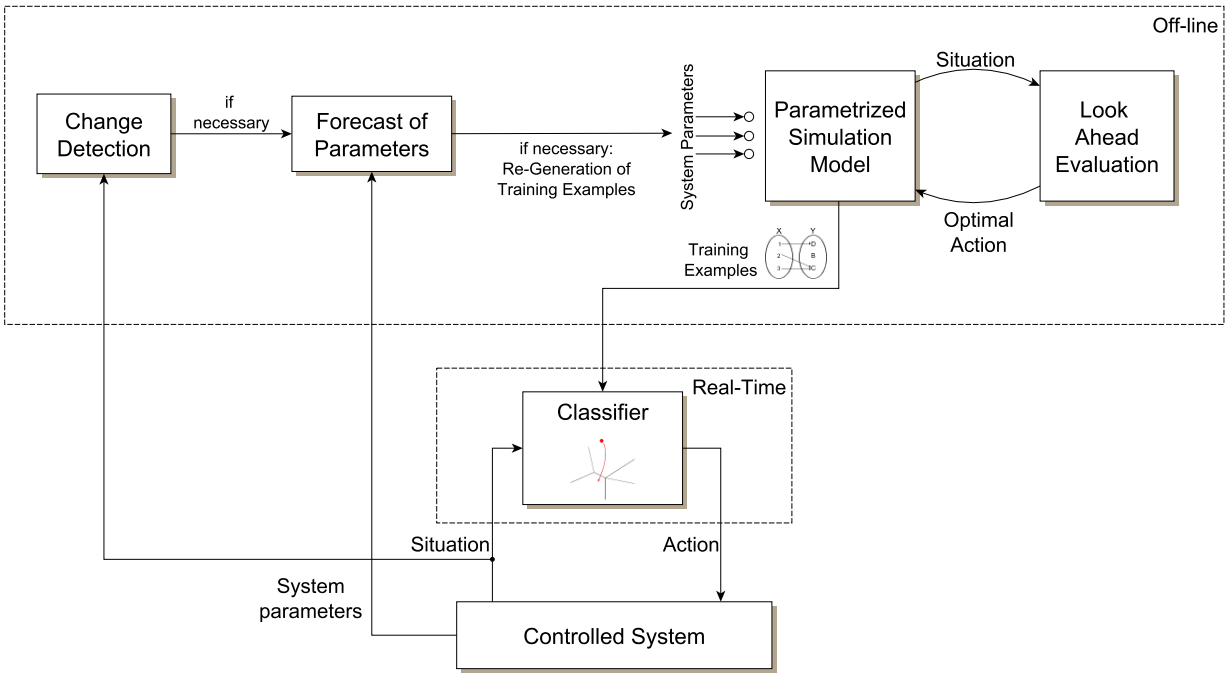

Figure 1: Components of the generic concept in an overview.

Figure 1 illustrates the concept in a broad overview. As is the case with any knowledge based control method, a classifier reacts whenever a control situation arises in the controlled system. It chooses from a predetermined set of actions in order to maximize the performance of the system, i.e., it classifies the situation to an action in real-time. This decision depends on a set of training examples, i.e., situation-action pairs, that the classifier learnt from using inductive machine learning algorithms. We define knowledge in this context as the information contained in the set of training examples.

The goal of our concept is to provide relevant training examples of high quality. Relevance in this context means that the classifier always has learnt suitable training examples to situations that the system encounters. High quality means that the assigned action to a situation actually maximizes the performance of the system, also considering possible future circumstances.

We use a parametrized simulation model of the system to generate the training examples. As the space of possible situations is typically extremely large, the problem arises to predict the situations that will occur. Instead of relying on a priori, fixed assumptions that are not adaptable during operation, we additionally observe system parameters such as load patterns or machine configurations, which influence the occurring situations and may gradually change over time as well. We define system parameters in Section 3.2.

If the occuring situations differ from the generated training examples, i.e., the classifier does not have relevant knowledge to base its decision on, the Change Detection component triggers re-generation of training examples under the current system parameters. Compared to other methods of knowledge acquisition, such as experts, the simulation generates the knowledge autonomously, without any manual interference required. The classifier is continuously updated with relevant knowledge, and the control self-adapts to the current state of the system. The Change Detection component is further explained in Section 3.3.

In addition to the a posteriori self-adaptation after a detected change, future system parameters are forecast in order to anticipate likely changes. This allows for a priori generation of training examples that may be relevant in the future, before a change actually happens. This helps to prevent cases where the classifier does not have any relevant knowledge to an occurring control situation. The forecast component is detailed in Section 3.4.

In order to generate a training example, a control situation must be assigned to the according optimal action. We introduce a novel simulation based look ahead evaluation of each possible action. Basically, the simulation is able to analyze how the system would react after application of each control action. Additionally, scenarios are considered under different stochastic developments - for example possible breakdowns - and are considered in the evaluation. This allows for a greater robustness of the control towards such stochastic influences. We take advantage that generation takes place offline and use as much computing time as is available. The simulation based look ahead evaluation is triggered whenever a new control situation arises in the simulation, and returns the determined optimal action. The approach is detailed in Section 3.5.

The resulting generated training examples are then processed by an inductive machine learning algorithm (the choice of which depends on the application). This allows detection of patterns in the examples and reduction of the necessary data used by the real-time control, therefore leading to a speed-up.

3.2 Parameter Definition

As depicted in Figure 1, we observe control situations and system parameters. Control situations are defined as quickly fluctuating factors of the system that are relevant to the decision of which control action to apply, e.g., the current open order set. In contrast, system parameters are slowly changing circumstances, such as the amount of machines present - they may change, e.g., a new machine is set up. This differentiation allows a small situation space, which speeds up classification.

System parameters also influence which situations may occur - if the amount of machines is high, there will likely be a smaller set of open orders, and vice versa. Therefore, feeding the amount of machines into the simulation model allows us to disregard situations with a large open order set, as they are highly unlikely to occur in the controlled system.

The system parameters form a space $P = P_1 \times P_2 \times \ldots$, consisting of i attributes such as, e.g., the number of machines or average throughput per machine.

The relationship of the space of parameters P to the situation space $S \subseteq \mathbb{R}^n$ and action space A is as follows: $P \times S \to A$ is functional, meaning that for any fixed parameter set, for each situation, exactly one optimal action is assigned. The set of generated training examples is denoted as $T_g \subseteq S \times A$, with situations on the left hand side of all generated examples denoted as S_g

3.3 Change Detection

The goal of the component is to decide, considering an occurred situation $s_o \in S$ and trained for situations S_g, whether the classifier has relevant enough information. We base our decision on a distance measure within S on $s_1, s_2 \in S$, the generalized Hamming Distance (Beierle and Kern-Isberner 2008):

$$d_S(s_1, s_2) = \frac{\sum_{i=1}^{n} w_i \cdot \min(1, \max(0, \frac{|s_{1_i} - s_{2_i}|}{i_o - i_u}))}{\sum_{i=1}^{n} w_i}$$

Where i is the index of the attributes of s, i_o and i_u are given upper and lower bounds of each component of S, and w_i is the information gain of each attribute (compare with Russell and Norvig 2009).

Based on the distance measure, we are able to define a binary criteria:

$$R(S_g, s_o) = \min_{s_e \in S_g} d_S(s_e, s_o) \leq d$$

If there is a trained for situation s_e that is closer to s_o than the threshold value d, we do not need to re-generate examples. We use the minimal distance between two situations that have been assigned different actions to define d:

$$S_{\text{diff}} = \{(s_1, s_2) \in S_g \times S_g | T_g(s_1) \neq T_g(s_2)\}$$
$$d = \min_{(s_1, s_2) \in S_{\text{diff}}} d_S(s_1, s_2)$$

Therefore, we consider how large the distance between situations needs to be so that a different action was assigned. If the occurred action is within that distance, we can reasonably assume that the same action that is assigned to nearby situations is a good choice.

If there is no trained for situation close to the occurred one s_o, we trigger re-generation of training examples. The simulation model will then be run under current system parameters, and given the validation of the model, situations close to s_o should be assigned to optimal actions and subsequently be trained for.

3.4 Forecast of Changes and Adaptation

Re-generation is triggered after a detected change. As this is done off-line, the result of the re-generation comes too late for the current occurred situation in the controlled system. It is therefore the aim of the Forecasting component to prevent such cases from happpening.

Instead of solely reacting to changes in the system, the component uses a forecast of system parameters to anticipate likely future situations. We therefore need a forecasting method that predicts future system parameters $p_k \in P$, where k is a future point in time, based on a number of past system parameters. The method must be able to recognize a trend in the data. We identified the widely used second order exponential smoothing as a suitable forecasting method, as it fulfills our requirements and is universally applicable (Armstrong 2001).

When a change is detected, we re-generate training examples not only using the current system parameters, but also forecast values. If the forecast is correct, this prevents future cases in which classifier encounters a situation it has not trained for - the change has already been anticipated and the new system parameters are already considered in the training examples.

To account for errors in forecasting, we not only simulate using the forecast values, but also parameter values jittered around and inbetween (Figure 2). For each predicted value $p_m \in P$, we may define:

$$p_m^{o_i} = p_m^i + 0.25 \cdot (p_{m+1}^i - p_m^i)$$
$$p_m^{u_i} = p_m^i + 0.25 \cdot (p_{m-1}^i - p_m^i)$$

as lower and upper deviations from the forecast value in each dimension i. In the two-dimensional case depicted in Figure 2, this results in 4 additional simulated parameter values $(p_m^{u_1}, p_m^{u_2}), (p_m^{o_1}, p_m^{u_2}), (p_m^{u_1}, p_m^{o_2}), (p_m^{o_1}, p_m^{o_2})$.

The forecast is run continously, and the anticipation component itself is able to proactively trigger re-generation, rather than waiting for a detected change in situations, when it anticipates that future situations will be significantly different than those considered in the current training examples. This allows the system to have enough time to generate a huge number of training examples before that change occurs.

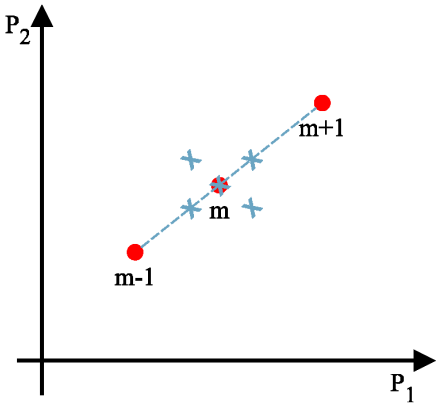

Figure 2: Jitter of system parameters around each forecast value m. Points marked with X are the system parameters that are fed into the simulation model.

3.5 Stochastic Look Ahead Evaluation

During generation of training examples in the simulation of the controlled system, control situations occur and are then assigned to the corresponding optimal action. We determine that assignment by evaluating the performance of the system after application of each possible action using the look ahead simulation.

We represent the look ahead simulation using a scenario tree, depicted in Figure 3. The original control situation, for which we are seeking the optimal action, forms the root. We implemented the look ahead simulation by copying the simulation state in memory, and subsequently simulating the system in parallel under application of each action a_i, until a following control situation occurs. The scenarios further branch out, until a final state in each branch is reached. The optimal action in the root situation can then be determined by comparing which action leads to the best final state, i.e., the state in which the system performed the best, according to a given performance metric.

Definition of a final state depends on the effect horizon of an action, i.e., how long the decision, which action to apply, has influence on the system's performance. Once the simulation time is higher than the effect horizon, a situation is a final state. Looking ahead by a considerable time prevents the evaluation from being too nearsighted, a lesson learned from the analysis of greedy algorithms in computer science (Cormen et al. 2009).

Furthermore, stochastic events are also considered. From each situation on, for each action there exist multiple following situations, as the stochastic events may occur inbetween (see Figure 4). Each action then leads to a multitude of final states, with varying probabilities, which may be multiplied along the path to the final state.

We then need to aggregrate each possible result of an action to a combined score. This problem has been well studied in the field of decision theory, and we identified the Hodges-Lehmann rule (Hodges Jr and Lehmann 1952) as suitable. The Hodges-Lehmann rule is considered robust as it factors in the worst case that may result for each action. In effect, an action that may perform slightly worse in the average case may be favored if it performs better under the influence of disturbing events.

The simulation is able to trigger such events to analyse the performance in each scenario. As the number of scenarios grows exponentially with regard to the effect horizon of an action, we implement several techniques to reduce the number of evaluated scenarios. For example, scenarios in which the system performs very bad may be discarded if it is clear that the action that led to that scenario is not the optimal one. The controlled triggering of events allows the systematic analysis of the scenarios, rather than account for randomness by running multiple repetitions with different seeds.

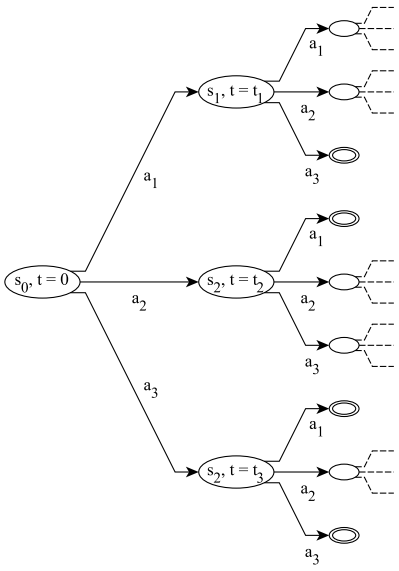

Figure 3: Scenario construction for look ahead simulation after applying each action a_i.

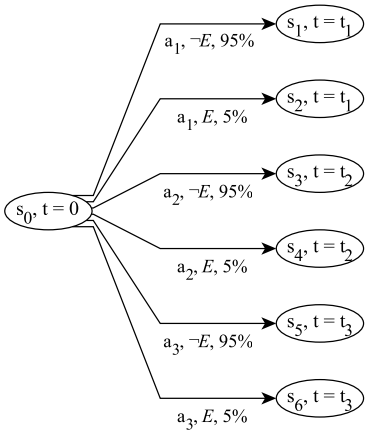

Figure 4: Scenario contruction additionally considering the stochastic occurrence of an event E.

It should be noted that the look ahead simulation based evaluation eliminates the need to use an application specific solving algorithm for the controlled system. Such an algorithm might still be used in place of the evaluation if one exists.

3.6 Application of the Generic Concept

The generic concept is defined independently of a concrete application. The following aspects need to be defined to adapt the concept in order to control a specific system:

- Define control *actions* and their effect horizons.
- Define a control *situation* and each component's bounds i_u and i_o.
- Define the *system parameters*, i.e., gradually changing circumstances that the control should be able to self-adapt to.
- Define any stochastic *events* and their assumed probabilities.
- Define a *performance metric* for the system.

4 APPLICATION TO A MATERIAL HANDLING SYSTEM AND EVALUATION

We illustrate and evaluate the generic concept by applying it to the control of a material handling system. An overview of the system is given in Section 4.1, while each necessary step of the adaptation of the concept is detailed in Section 4.2. We experimentally evaluate the resulting control in Section 4.3.

4.1 Industrial Use Case

We have applied our generic concept to a concrete industrial automated material handling system (MHS), which is currently in the advanced planning stage at Lödige Industries GmbH in Germany. Due to confidentiality agreements, we present the basic functionality of the system (which is sufficient to demonstrate how our concept is applied), rather than a more detailed view of the specific use case and type of material.

The MHS is used to store and retrieve unique objects from a ground floor into underground storage spaces on several floors ($z = -1, \ldots, -n$). It uses two kinds of movers, as depicted in Figure 5. Vertical

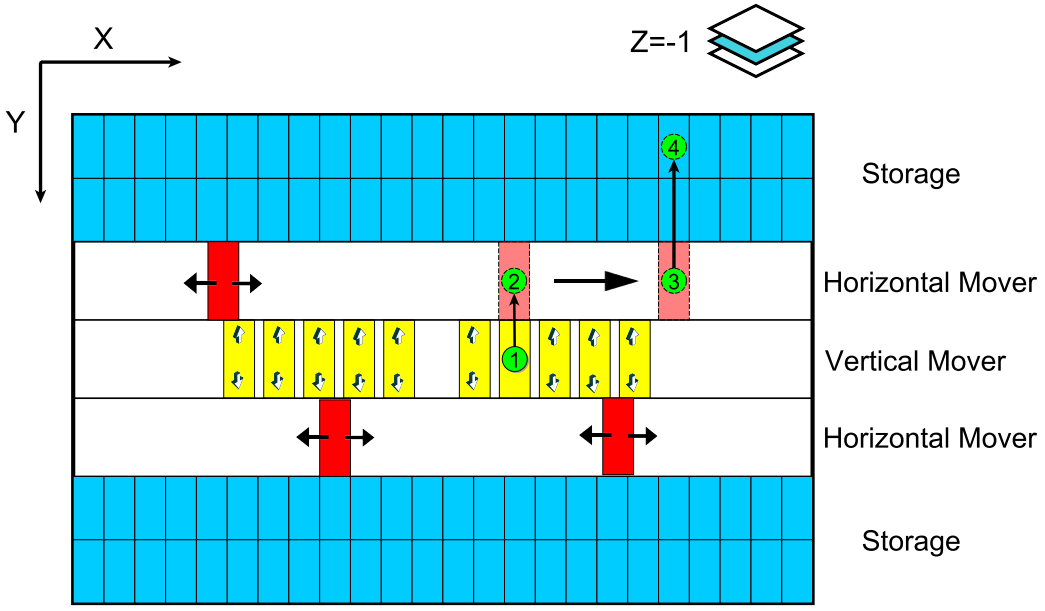

Figure 5: Basic operation of the MHS. A storage sequence is shown (steps 1-4).

Movers (VM) (in yellow) load the material at the ground level and transport them to a specific underground level, where they are picked up by a Horizontal Mover (HM). The HMs move in a fixed lane in one dimension (x-direction) and store or retrieve objects to or from storage spaces, which are aligned in two rows. In order to access a back storage space, the material in the front storage space must be removed first and stored in a different storage space. This procedure is called a shuffle.

The goal of the MHS is to serve external requests (which are issued on-line, i.e., at unknown times to the system) for storage or retrieval of specific objects using the available VMs and HMs. The control's task is to select a storage space for storage and issue movement signals to the HMs and VMs. It is also able to relocate objects between storage spaces in order to optimize their placement in the system, so to minimize future shuffles or the distance an HM has to travel from the VMs. The overall goal is to serve the requests as quickly as possible and therefore to minimize the waiting times between creation and fulfillment. It can be assumed that at the end of a fixed time period, e.g., a day, all objects have been retrieved and the MHS is empty, therefore a new cycle begins.

A simulation model of the system was developed using the material flow simulator d^3fact, which is developed at the Chair for Business Computing, esp. CIM at the Heinz Nixdorf Institute at the University of Paderborn (Dangelmaier and Laroque 2007, Renken 2013). The model allowed both initial analysis of the system using a 3D visualization (depicted in Figure 6), as well as generation of training examples as the parametrized simulation component described in Section 3. Using a default control strategy, optimization potentials were identified using further statistics panels, such as a status chart of movers, shown in Figure 7.

4.2 Adaptation of the Concept

As the control task is a highly dynamic problem and depends on the used storage spaces, current and future external requests and availability of movers, our concept is well suited for application. Especially the self-adaptation of training data generation is required as the external request behaviour is expected to change gradually over time in the real world MHS use case scenario. Adaptation of the control concept

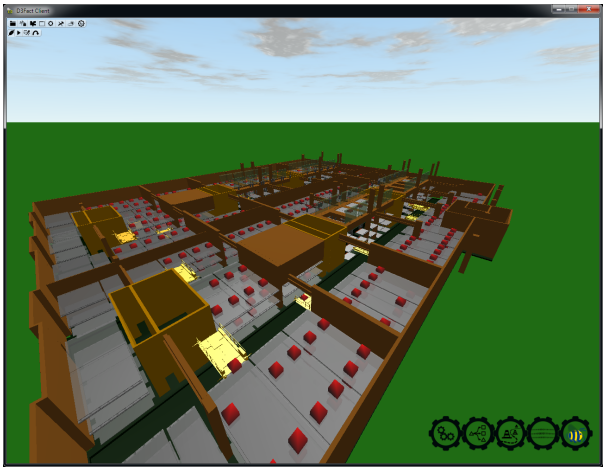

Figure 6: 3D visualization of the MHS in d^{3}fact. An HM is shown in the foreground moving an object into a storage space.

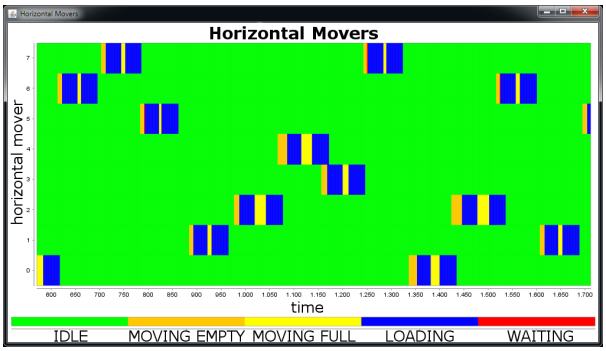

Figure 7: Status Chart depicting the status of each HM over time.

to the MHS requires definition of the application specific parts of the concept, as detailed in Section 3.6. Adaption was performed as follows:

The *actions* that the control chooses from are a combination of smaller strategy choices. The first one is the selection of a storage space, where the control may use spaces "inside out", meaning starting at the space in the middle, i.e., close to the Vertical Movers, and only using the outer, far away spaces when inner ones are full. The HMs would then have to travel less distance and could serve requests quicker. The "outside in" strategy is the opposite direction and may be useful in times of a low request load, when the HMs have enough time to travel the farther distances. This would save the nearer middle spaces for times of many storage requests.

The second strategy is "prefetching" for storage, which means that HMs move in front of a VM when a new object is stored, even if the VM has not arrived at the level yet. This allows the VM to immediately load the object to the HM. The third strategy activates relocating of objects from inner spaces to outer ones.

In combination, three actions were defined:

Action	Prefetching	Relocation	Storage Space
"Default"	No	No	Inside Out
"Storage Priority"	Yes	No	Inside Out
"Maintenance"	No	Yes	Outside In

The control chooses between these actions at fixed intervals (each hour), based on the classification of the current system *situation*, which was defined for the MHS based on analysis of which circumstances are relevant to the decision. Here the action choice depends on the system load, which is the only external influence on the system that is not directly controlled by the system. It also fulfills the other criteria (i.e. it is a fluctuating circumstance) for a *situation* outlined in Section 3. Therefore the number of current storage and retrieval requests constitute the *situation* definition.

The control is required to adapt to changing system load distributions, as the real world use case of the MHS demands handling two types of changes in request behaviour: The first type of change is a long term shift, e.g., a gradually increasing amount of requests. The request distribution over a day therefore fulfills the criteria for *system parameters* and is monitored by the control and fed into the simulation for re-generation in the event of a registered or an anticipated change.

The second case of change is a sudden and unforeseen surge of retrieval requests, i.e., quickly emptying the system. Such a surge is defined as an *event* for the simulation based look ahead evaluation of actions.

This means that the system considers the possibility of a surge when evaluating which action to choose, as well as already providing some training examples for situations that arise after a surge occurs.

The event horizon of each *action* for the simulation based evaluation here is defined as the time left in the current day. This is due to the fact that actions chosen during the day may change where objects are located for later retrieval. Therefore, e.g., selecting "Maintenance" early in the day may influence waiting times for retrieval of objects in the evening, and this must be considered when evaluating whether to choose "Maintenance" early. The *performance metric* for the evaluation is the overall average waiting time for requests.

4.3 Experimental Evaluation

We have run the training data generation using a sample load distribution with a duration of 12 hours. It includes hours of high loads to stress the system, as well as low loads inbetween, as this is expected in the undisclosed use case. The simulation based look ahead evaluation of actions then determined the best action for each hour and generated a Weka ARFF file with the corresponding training examples. As it was required to look ahead in the simulation until the end of the day, pruning had to be rather conservative, which led to ca. 1300 different nodes to be simulated. The evaluation was run in parallel on a quadcore, 3.4ghz Intel Core i7 processor and finished in 2.5 hours of computation.

The look ahead simulation also considered the possibility of a surge, resulting in a different action selection in one instance. Specifically, at the 7th hour, there are many storage requests and a few retrieval requests expected. In the expected case, choosing "Maintenance" in the hour before (a period of low load) results in 1.5% lower waiting times, compared to the "Default" case without relocations. However, in the event of a surge of retrieval requests, the 7th hour consists of the opposite load pattern: many retrievals and no storage requests. The effect of "Maintenance" in the 6th hour under that scenario is an increase of 2.8% of waiting times. This is due to the fact that relocations move cars to the far side of the MHS, away from the VMs. The Hodges-Lehmann rule aggregrates both scenarios and gives a slightly better performance to the "Default" action in the hour before.

In order to benchmark our approach, we measured the waiting times when dynamically switching the action using the classifier trained by the generated data under the sample load distribution. Neural Network (trained by Backpropagation) was found to perform the best in this application. The waiting times include the fixed times for moving objects between movers, which cannot be improved by the control, so that only moderate gains could be expected here. Factors that could be influenced include blocking times (e.g., a VM has to wait for an HM). Furthermore, travel times by the HMs can be reduced by minimizing the traveled distances by placing objects closer to the VMs. We compare the performance relative to fixed application (i.e. constantly for the whole day) of each of the three actions.

The results are illustrated in Figure 8. It can be seen that of the three actions, "Storage Priority" performs the best by a considerable margin (6% and 8% better than the "Default" action, incoming and outgoing requests respectively). We conclude that prefetching of the HMs is beneficial in most cases, as VMs are slower in comparison to HMs and therefore form a slight bottleneck in the MHS. Prefetching increases VM capacity utilization, leading to higher overall performance. The "Maintenance" action by itself led to a similar performance than "Default".

Dynamically switching between the three actions based on the generated training data in the simulation led to lower waiting times for requests by an average of 14.2%, compared to the "Default" action, and it performs 9.8% better compared to the best single action "Storage Priority". We conclude that it is in fact beneficial to use the times of low system load for "Maintenance" and alternating storing the material in close and far away locations, based on system load. We measured that the total amount of distance that HMs travel in periods of high load is decreased by 2.7%, and subsequently the amount of time that HMs are waiting for VMs is cut almost in half.

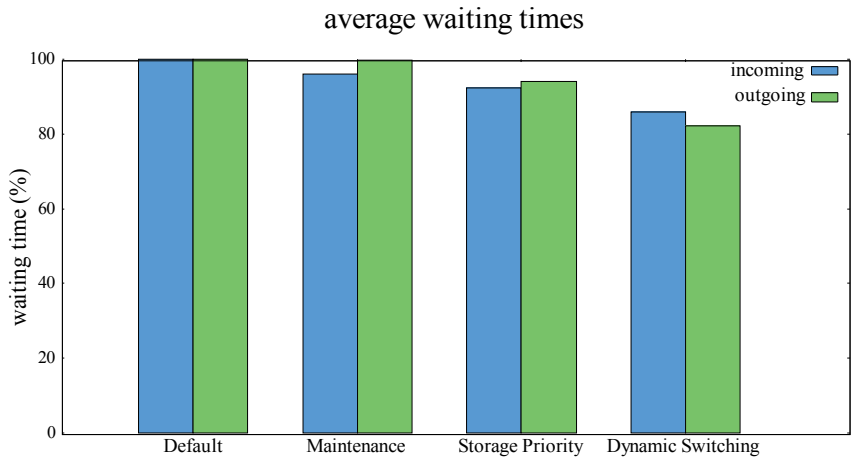

Figure 8: Results (lower is better)

In sum, our generic concept was successfully implemented in the control of a real world industrial MHS. The control was able to show performance improvements. The criteria defined for each application specific adaptation were useful in identifying each control component such as the *actions*.

5 CONCLUSION

Our approach of a self-adapting control was generically laid out and successfully applied to an industrial material handling system. Results show a performance increase due to the dynamic action selection. We demonstrated how the control is able to self-adapt to differing load patterns and account for unforeseen events in the application.

The generic nature of the concept promises successful future application of the concept to other systems across logistics, for which we laid out the necessary steps.

Further extensions of the concept include a higher degree of self-adaptation, such as an automatic evaluation and modification of the action set and situation definition.

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

Armstrong, J. 2001. *Principles of Forecasting: A Handbook for Researchers and Practitioners*. International Series in Operations Research & Management Science. Springer.

Aufenanger, M., W. Dangelmaier, C. Laroque, and N. Rüngener. 2008. "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. Mnch, O. Rose, T. Jefferson, and J. W. Fowler, 1952–1958. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Beierle, C., and G. Kern-Isberner. 2008. *Methoden wissensbasierter Systeme: Grundlagen, Algorithmen, Anwendungen*. Wiesbaden: Vieweg+Teubner / GWV Fachverlage GmbH, Wiesbaden.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein. 2009. *Introduction to Algorithms, Third Edition*. 3rd ed. The MIT Press.

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

Hodges Jr, J. L., and E. L. Lehmann. 1952. "The use of previous experience in reaching statistical decisions". *The Annals of Mathematical Statistics*:396–407.

Klaas, A., C. Laroque, M. Fischer, and W. Dangelmaier. 2011. "Simulation Aided, Knowledge Based Routing for AGVs in a Distribution Warehouse". In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, 1673–1684. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Kwak, C., and Y. Yih. 2004. "Data-mining approach to production control in the computer-integrated testing cell". *Robotics and Automation, IEEE Transactions on* 20 (1): 107–116.

Müller-Schloer, C., H. Schmeck, and T. Ungerer. 2011. *Organic Computing - A Paradigm Shift for Complex Systems*. Basel: Springer Basel AG.

Noack, D., M. Mosinski, O. Rose, P. Lendermann, B. P. Gan, and W. Scholl. 2011. "Challenges and solution approaches for the online simulation of semiconductor wafer fabs". In *Proceedings of the Winter Simulation Conference*, edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, 1845–1856: Institute of Electrical and Electronics Engineers, Inc.

Prothmann, H., F. Rochner, S. Tomforde, J. Branke, C. Müller-Schloer, and H. Schmeck. 2008. "Organic Control of Traffic Lights". In *Proceedings of the 5th international conference on Autonomic and Trusted Computing*, ATC '08, 219–233. Berlin, Heidelberg: Springer-Verlag.

Renken, Hendrik 2013. "d^{3}fact project website". Last modified July 15. http://www.d3fact.de.

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.

Tompkins, J. 2010. *Facilities Planning*. John Wiley & Sons.

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, intelligent 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 e-mail is Christoph.Laroque@hni.upb.de.

HENDRIK RENKEN studied computer science at the University of Paderborn, Germany. Since late 2007 he is a research assistant at the Heinz Nixdorf Institute. His research interests are material flow simulation models. His e-mail address is Hendrik.Renken@hni.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 e-mail address is dangelmaier@hni.upb.de.