

## **SIMULATION-BASED OPTIMIZATION OF PAINT SHOPS**

Marco Lemessi

Thomas Schulze  
Simeon Rehbein

Deere & Company European Office  
John-Deere-Straße 70  
Mannheim, 68163, GERMANY

Otto-von-Guericke-University Magdeburg  
Universitätsplatz 2  
Magdeburg, 39106, GERMANY

### **ABSTRACT**

Several factors and restrictions affect paint shop design. Due to the high level of complexity of paint shops, classical mathematical optimization methods are generally not applicable. Simulation-based optimization has been often used in recent years as an alternative to classical mathematical optimization methods. This paper presents an optimization function for paint shop design, its constraints, and the optimization algorithms used to evaluate valid alternatives. It also discusses execution speed issues when the proposed optimization process is applied to a set of case studies.

### **1 INTRODUCTION**

The use of simulation models in operational planning and support is well known and has attained a high level of importance in the last few decades. Different system configurations can be simulated and evaluated based on simulation results. As a consequence, interest in simulation-based optimization has been growing over the last few years (Hong and Nelson 2009). Such interest is due to the fact that real-world manufacturing problems are characterized by a high level of complexity and are not solvable by analytic methods or enumeration (Weigert, Henlich and Klemmt 2008). Simulation-based optimization is used in tactical factory-planning process (Gürkan 2000) and in operative decision making (Grewal, Enns and Rogers 2009; Kohn et al. 2009; Moeller et al. 2009; Weigert, Henlich and Klemmt 2008) to improve operational processes and reduce both investment costs and process times. The word “optimization” implies finding the best possible solution. However, a feasible solution, which improves the current state and is found in reasonable time, is generally acceptable in real-world applications. In any case, simulation models are an approximation of reality (Wiedemann and Krug 2003). There is a wide range of possible optimization methods which can be used with regard to simulation, ranging from operation research techniques to metaheuristics or similar methods, to application-specific heuristics.

Paint shop design is a complex task. The main goal is designing a system capable of painting a target throughput (e.g., a given amount of parts to be painted per week). Calculating analytically the results of a specific system configuration is generally not feasible, due to the variety of parts to be painted and different daily schedules (e.g., of workers). This is one reason for using simulation. A specific throughput can be reached with different system configurations. As a consequence, the design goal is finding a configuration which reaches the required throughput on one hand, and minimizes investment costs on the other. For a widespread use of simulation-based optimization in factory-planning activities it is necessary to support the user in the factory-planning process. Dedicated activities within this process, like determination of alternatives or simulation of these alternatives, have to be executed automatically and hidden from the user. To support these activities an existing generic simulation model for paint shops has been embedded into an optimization environment which manages the optimization and offers a graphical user interface (GUI) for users.

The remainder of this paper is structured as follows: Section 2 describes a generic approach for modeling paint shops which is the foundation for the used simulation model. Section 3 introduces the optimization problem and Section 4 describes the used optimization process. Two heuristic optimization algorithms used to minimize path lengths and number of carriers are explained in Section 5. Section 6 discusses some aspects of execution speed, including the concept of Check-Point-Restore. Section 7 presents the activities for needed user support and Section 8 shows optimization results for a set of case studies. Section 9 provides a brief summary and introduces future work.

## 2 GENERIC MODELING OF PAINT SHOPS

A paint shop can be modeled as a conveyor system, which moves different part types on different carrier types through the system. More information about the methodical procedure for describing such systems with simulation is provided by (Williams and Sadakane 1997). The used generic simulation model consists of a flow graph with directional links and nodes. Figure 1 presents a schematic view of the model structure. Major inputs are: conveyor network (length, capacity, routing logic, etc.), carrier types and number of carriers per type, system operating times (number of shifts, breaks, etc.), number of technicians and their assignments (flexing patterns, effective minutes per shift, etc.) and work/equipment times. The simulation generates detailed statistics on throughput per part type, technician utilization and queue lengths. The model takes into account randomness of work times and part arrivals (sequence and interarrival times).

Nodes in the network are referred to as “stops” and links as “paths”. At any stop different tasks (e.g., loading or unloading parts on carriers) or specific value-added processes (e.g., heating parts) can be executed. Carriers move along paths, where equipment (e.g., ovens or washers) might be located.

The basic idea behind generic simulation models is that all needed simulation objects are created and instantiated during run-time by using a model-class library. The complete conceptual model is stored in a database with predefined tables. After launching the simulation model, the database is read and the needed objects are created according to the library. At the end of the simulation, results are written back to the database. The advantages of this technique are two: the conceptual model is created only once, and necessary changes for model experiments are executed in the database. The generation of a simulation model for a specific optimization alternative is performed automatically with the model-class library and the connected database. This approach can be applied for simulation software which allows object creation during run-time. In this case SLX (Simulation Language with eXtensibility) from Wolverine Software Corporation was used as the simulation software (Henriksen 2011).

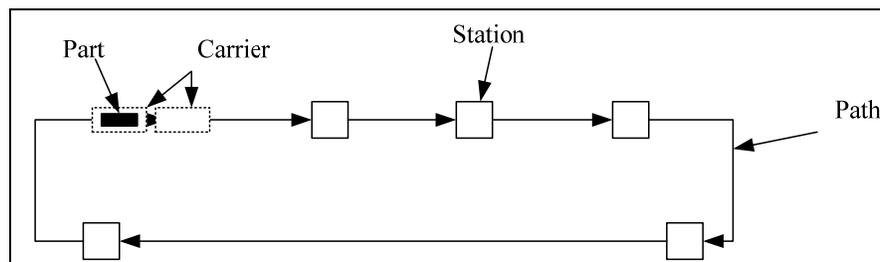


Figure 1: Schematic view of the simulation model

## 3 OPTIMIZATION PROBLEM

Number of carriers and path lengths significantly influence investment costs of paint shops. As a consequence, the optimization target can be defined as the minimization of the investment cost for carriers and paths. This optimization problem can be formulated as follows.

**Objective Function**

$$\min \rightarrow IC_j = C(A_j) = \sum_i^y ICC_i \cdot nC_{j,i} + \sum_k^z ICP_k \cdot \ell P_{j,k}, \forall j=1 \dots m, i=1 \dots y, k=1 \dots z \quad (1)$$

**Constraints**

$$\min \ell P_k \leq \ell P_{j,k}, \forall j=1 \dots m, k=1 \dots z \quad (2)$$

$$\Phi(A_j) = T_{j,i} \geq TT_i, \forall j=1 \dots m, i=1 \dots y \quad (3)$$

**Variable Declaration**

$m$	Total number of alternatives evaluated
$y$	Total number of carrier types in the system
$z$	Total number of paths the conveyor network is made of
$IC_j$	Total investment cost for alternative $j$ , with $j=1 \dots m$
$ICC_i$	Investment cost for carrier type $i$ , with $i=1 \dots y$
$nC_{j,i}$	Number of carriers of carrier type $i$ in alternative $j$ , with $i=1 \dots y$ and $j=1 \dots m$
$ICP_k$	Investment cost per length unit for path $k$ , $k=1 \dots z$
$\ell P_{j,k}$	Length of path $k$ in alternative $j$ , with $k=1 \dots z$ and $j=1 \dots m$
$\min \ell P_k$	Minimum acceptable length of path $k$ , with $k=1 \dots z$
$T_{j,i}$	Throughput of carrier type $i$ in alternative $j$ , with $i=1 \dots y$ and $j=1 \dots m$
$TT_i$	Target throughput of carrier type $i$ , $i=1 \dots y$
$A_j = \begin{pmatrix} nC_{j,i} \\ \ell P_{j,k} \end{pmatrix}$	Alternative $j$ , $j=1 \dots m$ , $i=1 \dots y$ and $k=1 \dots z$
$C(A_j)$	Cost function and also target function
$\Phi(A_j)$	Actual throughput per carrier type calculated via simulation

$$IC_j \in R^+, ICC_i \in R^+, nC_{j,i} \in N^+, ICP_k \in R^+, \ell P_{j,k} \in R^+, \min \ell P_k \in R^+, T_{j,i} \in R^+, TT_i \in R^+$$

Different carrier types with their investment costs are considered, since different carrier types may be used in the system to transport different parts. All paths in the model are considered individually, with their attributes of length and investment cost per length unit. Based on this information an alternative can be defined with a specific number of carriers for each carrier type and a specific length for each path. As a consequence, the objective function presents the total cost of one alternative.

Two constraints are taken into account. The first constraint limits the minimum length which is acceptable for a path, due to layout or geometry constraints. The second constraint takes simulation results into account: a simulated alternative must reach a specific throughput for each carrier type. Simulation results are based on confidence intervals and underlie a defined confidence level, with the assumption of asymptotic normality. If both constraints are met by an alternative, the alternative is considered valid and is one (not necessarily the best) solution for the optimization problem.

**4 OPTIMIZATION PROCESS**

Prior to starting the optimization process a valid simulation model must be created by the user. This start alternative has to meet all defined constraints (see Section 3) and is the basis for optimization. During the optimization process different alternatives are generated. Each alternative consists of a specific number of carriers for every carrier type and a specific length for every path. Alternatives are generated through a heuristic approach, which operates as follows:

- In the first step, path lengths are minimized and the number of carriers in the system is kept constant.

- When the first step is complete, the second step minimizes the number of carriers in the system and the lengths of the paths (reduced in the first step) are kept constant.

Executing both steps in sequence is called a “loop”. During the optimization process several loops may be executed. Minimizing path lengths can create a new starting point, from which the minimization of the number of carriers in the system can start and vice versa. The optimization process is terminated when neither the number of carriers in the system nor the lengths of the paths can be reduced any further. The iterative-generated improvements during the optimization are shown in Figure 2 for one test case. Up to alternative #49 path lengths are minimized and the number of carriers in the system is kept constant (step #1). Between alternative #50 and alternative #62 the number of carriers in the system is minimized and path lengths are kept constant (step #2). Since the number of carriers is reduced in the second step, a second loop is executed to evaluate the possibility of further reducing path lengths. In this second loop both number of carriers and path lengths are reduced, even though in a smaller way. Due to the improvements of both targets a third loop is executed, but the third loop does not find any new valid alternative. After alternative #72 neither the number of carriers nor the path lengths can be reduced any further, and therefore the optimization process is terminated.

It is also possible to minimize either the number of carriers in the system or the path lengths. In this case multiple loops are not needed, since there is no interaction between the two targets.

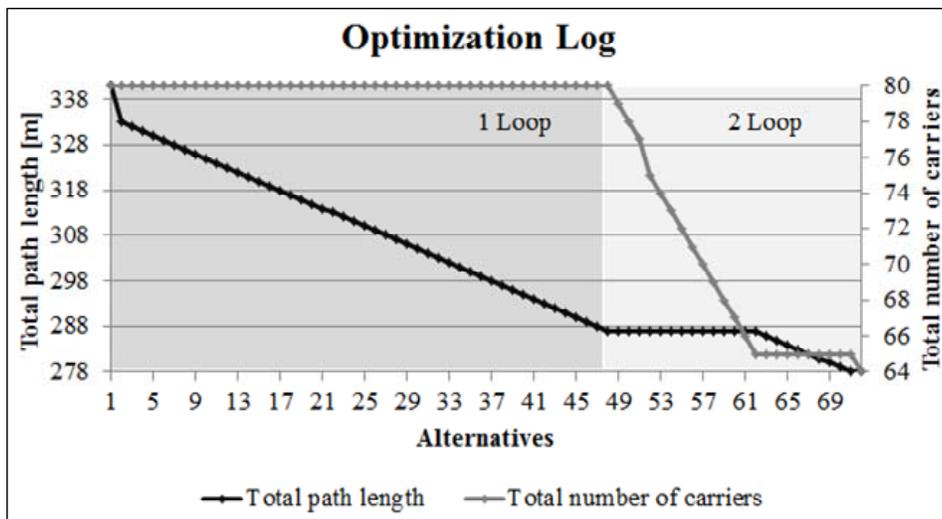


Figure 2: Optimization log

The optimization process for each alternative can be subdivided into the following 3 phases:

- Creating a new alternative,
- Executing the simulation in multiple runs, and
- Evaluating simulation results.

New alternatives are created by algorithms, by changing either the number of carriers or paths lengths. Two algorithms are used, one for each optimization target. As a consequence, in each loop two different algorithms are executed (see Figure 3). Both algorithms are described in Section 5 in more details.

For each alternative multiple simulation runs are executed, due to the stochastic behavior of the model. The optimization process for each run can be subdivided into the following 3 phases:

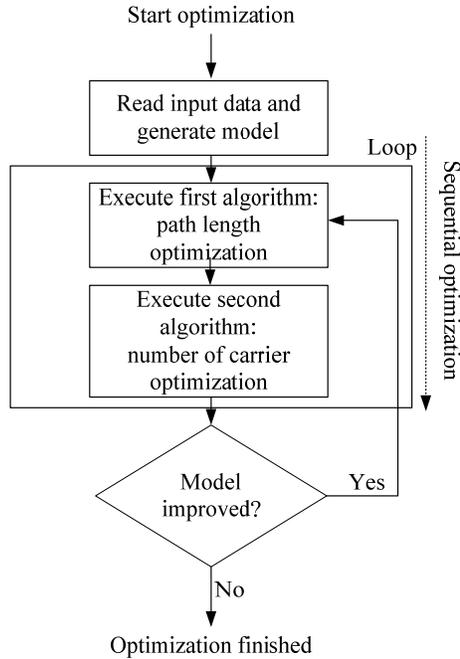


Figure 3: Optimization organization – loop concept

- Starting the simulation.
- Clearing the model at the end of the run (this includes, for example, deleting active objects, changing random-number seeds, and resetting the simulation clock).
- Evaluating simulation results, i.e., throughputs per carrier type. The optimization problem described in Section 3 is used in this phase.

Confidence intervals are used to evaluate if an alternative is valid. An acceptable balance must be found between the width of the confidence interval (which influences the accuracy of results) and the number of runs (which influences the execution time of the optimization process). The following constraint must be met for all confidence intervals of the carrier type throughputs:

**Constraint**

$$T_{j,i}^{mean} - TT_i \geq T_{j,i}^{half\ width} \tag{4}$$

**Variable Declaration**

- $TT_i$  Target throughput of carrier type  $i$ , with  $i=1 \dots y$
- $T_{j,i}^{mean}$  Mean throughput of carrier type  $i$  in alternative  $j$ , with  $i=1 \dots y$  and  $j=1 \dots m$
- $T_{j,i}^{half\ width}$  Half width of the confidence interval of mean throughput of carrier type  $i$  in alternative  $j$ , with  $i=1 \dots y$  and  $j=1 \dots m$

$$TT_i \in R^+, T_{j,i}^{mean} \in R^+, T_{j,i}^{half\ width} \in R^+$$

Equation 4 shows that the number of runs per alternative is not constant, but depends on the half width of the confidence interval, and the throughput of the alternative. If an alternative produces very high mean values and therefore a very high throughput, very small confidence intervals are not needed. If the difference is very small the confidence interval must be also very small in order to take an accurate decision on the validity of the alternative.

In addition, the number of runs is also defined with the following criteria:

- At least 10 runs per alternative must be executed.
- After 10 runs, the constraint described in Equation 4 is checked and if the constraint is met no additional runs are needed. Otherwise, one additional run is executed and the constraint is checked again and so forth.
- In case of extremely small differences the maximum number of runs is limited to 50.

If the confidence intervals are small enough for all carrier types the alternative can be evaluated. This is done based on the constraint defined in Equation 3: target and actual throughput of each carrier type are compared. In addition to the described constraint, in the optimization problem, the evaluation is carried out by using confidence information as well. This process extends the prior-defined optimization constraint in the following way:

**Constraint**

$$\Phi(A_j) = T_{j,i} = T_{j,i}^{mean} - T_{j,i}^{half\ width} \geq TT_i \tag{5}$$

**5 OPTIMIZATION ALGORITHMS**

The optimization algorithms are essential parts in the optimization phase “Creating a new alternative” (see Section 4) and are responsible for changing number of carriers and path lengths in the model. The algorithms are performed in sequence and each one of them focuses on one optimization target. Both algorithms have a heuristic nature and a similar simple structure. The algorithm for optimizing path lengths is executed first.

Each path is assigned a length and a capacity in terms of “places” for carriers. Similarly, each carrier type is assigned a length and a specific capacity size. As a consequence, two conditions must be met for a carrier to enter a path (see Figure 4):

- First, the path needs to be long enough and have enough capacity. Path length and path capacity can be defined independently from each other. This gives the possibility to model horizontal curves and to prevent many carriers from being on the path at the same time.
- If different carrier types are used in a model, all carrier types must have the same ratio between length and size (later referred to as *length/size ratio*). Example: if carrier type *i* has a length of 1.5 m and its size is 1, carrier type *i + 1* must have a length equal to 1.5 m times its size.

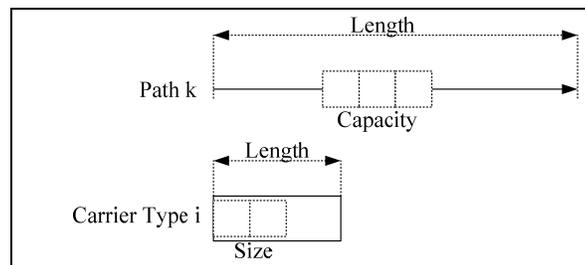


Figure 4: Path and carrier definition

The algorithm takes into account the average number of carriers of size 1 being observed on the path over the entire simulation (later referred to as *observed average*). The largest the difference between path capacity and observed average, the highest the potential for the path to be reduced.

The algorithm changes both path capacities and lengths during the optimization process. The length of a path is updated by multiplying its capacity times the length/size ratio. So the two path attributes become linearly dependent during the optimization process.

The algorithm logic is shown in Figure 5 and follows a two-step procedure:

- When the algorithm is executed for the first time, statistical information (e.g., observed average number of carriers per path) is collected. After the first execution, all path capacities are set equal to the maximum number of carriers observed on the path, and the lengths are updated accordingly. The new alternative is then simulated in multiple runs. If the target throughput is not met, the alternative is considered not valid and the algorithm resets the model to its last valid state.
- The second step starts by selecting a path to be changed. The algorithm selects the path with the largest difference between path capacity and *observed average* (see above). The path is then reduced by one capacity unit and its length is reduced accordingly. If the alternative is not valid the algorithm resets the model to its last valid state and tags the path as unchangeable. The algorithm terminates if no path can be reduced any further.

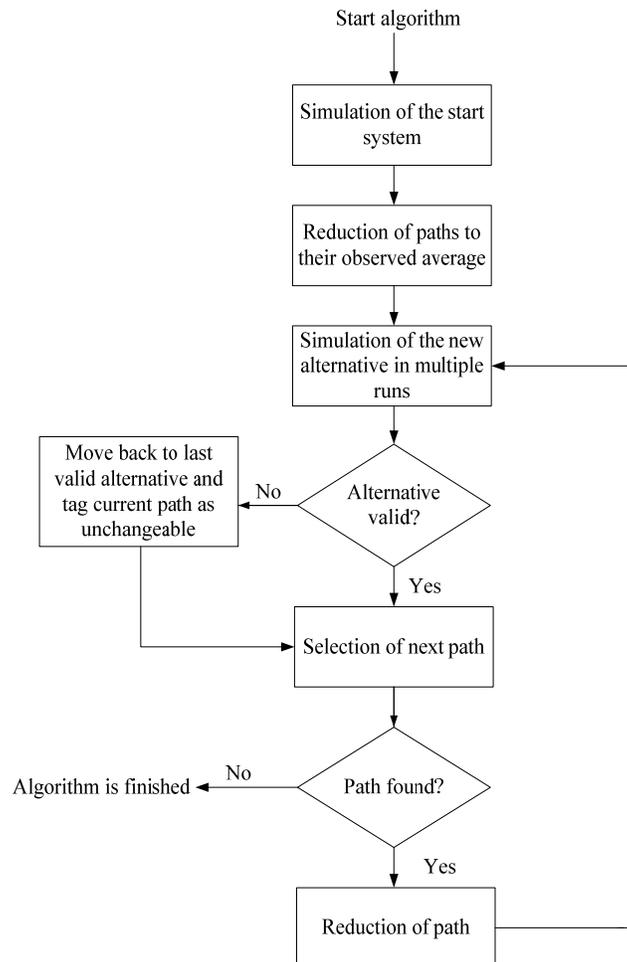


Figure 5: Optimization algorithm for reducing path lengths

Figure 6 shows the optimization results for one of the case studies. The model consists of a closed loop, which includes an area for loading and unloading carriers with parts. The optimization process is capable of reducing path lengths by 26%; reducible paths are highlighted with ellipses.

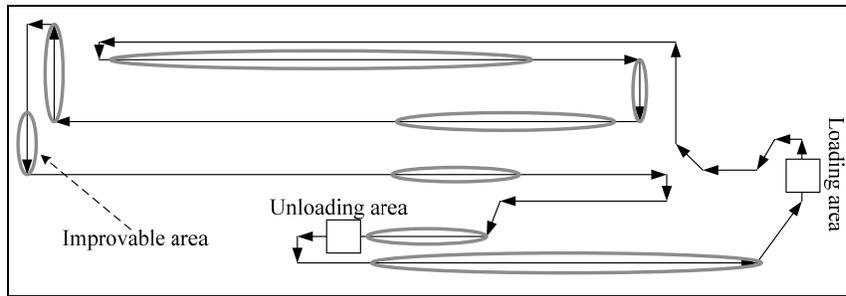


Figure 6: Case study network with improvements

Once the minimization of path lengths is complete, the second algorithm (minimization of the number of carriers in the system) is executed (see Figure 7). A paint shop may use different carrier types, and each type has its own number of carriers. The algorithm selects the first carrier type in the list of all carrier types (there is no additional logic for carrier type selection), reduces its number of carriers by one unit, and generates a new alternative. The simulation is executed in multiple runs and confidence intervals are taken into account to determine whether the alternative is valid. If yes, the algorithm reduces the number of carriers of the selected carrier type by one more unit. If the alternative is not valid, the algorithm loads the last valid alternative, tags the selected carrier type as unchangeable and looks for another carrier type to optimize.

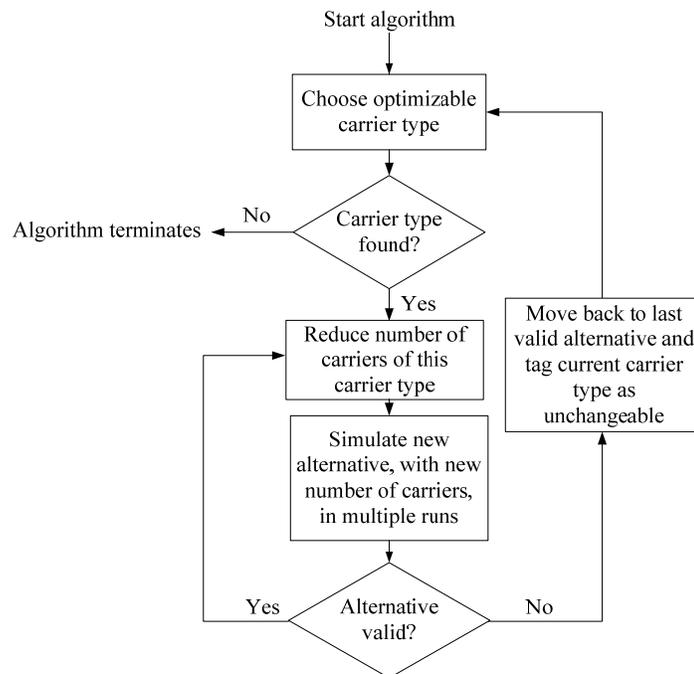


Figure 7: Optimization algorithm for reducing number of carriers

## 6 EXECUTION SPEED OF OPTIMIZATION

A key requirement for the optimization process is that it must be completed in a reasonable time. In typical business cases 80 to 100 alternatives have to be evaluated, with 10 to 15 runs per alternative due to the stochastic nature of input data. That amounts to approximately 1125 runs (90•12.5) per optimization. If the goal is for the optimization process to be complete in 8 hours or less, and assuming that simulation runs are executed in sequence, each simulation run should not take more than 21 seconds.

In order to meet the 21-second-per-run goal, only statistics which are relevant for the optimization process are collected. All additional calculations, including detailed output statistics and animation-related information, are not executed in order to save time. Also, static inputs (i.e., inputs which are not changed during the optimization process) are read only once, and all objects which are not affected by the optimization process are instantiated only one time too. At the end of each run all temporary dynamic objects are deleted and the remaining static objects are reset to their initial state, and all random numbers are re-initialized. By doing so, execution time can be significantly reduced.

Table 1 presents an overview of some case study parameters. This table shows the number of generated moving and passive objects being created during the optimization, and the average execution time per run. It can be seen that the execution time is relatively short compared to the high number of generated objects. In this context all case studies are run for 10 weeks, with a one-week warm-up (transition phase needed to reach a steady state).

Table 1: Case study parameters

Model ID	Generated moving objects	Generated passive objects	Average Execution time per run in seconds
1	73,127,241	566,678	1.98
2	352,946,395	684,291	2.35
3	90,601,525	2,943,199	3.44
4	11,232,257,058	5,926,572	23.27
5	1,652,994,615	2,513,588	7.30
6	2,009,867,637	4,079,507	14.04
7	1,586,238,176	3,987,449	13.46

The execution time per run can be subdivided into three phases (see Figure 8):

1. Warm-up phase, which consumes about 9% of the total execution time,
2. Data-collection phase, which consumes about 88% of the total execution time, and
3. Clearing phase, which consumes about 3% of the total execution time.

The presented values are averages generated from the case studies. Figure 8 gives a graphical representation of the 3 phases described above.

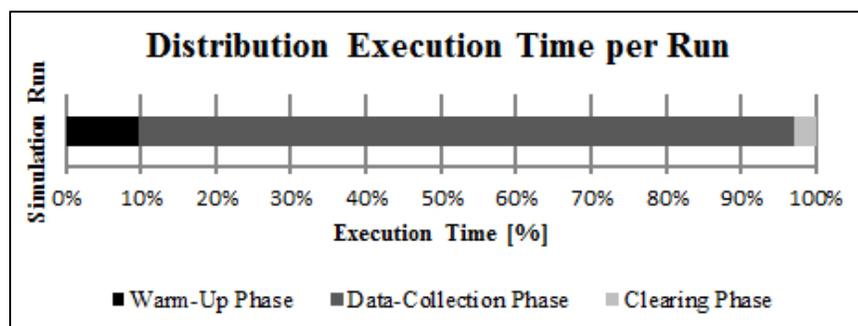


Figure 8: Phases of execution time per run

Based on this analysis, execution time can be reduced if the warm-up phase is skipped. One possibility is to use the Check-Point-Restore concept implemented in SLX. Check-Point-Restore allows making a complete copy of a model state and later jumping back to this state (Henriksen 1989). Check-Point-Restore can be used for skipping the warm-up phase for runs > 1. So warm-up and model clearing are executed only once per alternative.

After simulating the warm-up phase in the first run of any alternative, a copy of the current model state is stored and a checkpoint is set. At the end of a simulation run the simulation will jump back to this stored model state and the state indicated by this checkpoint will be restored. This approach skips the warm-up phase for all runs > 1 within one alternative. One important consideration must be made when using Check-Point-Restore for these purposes: jumping back to a stored state affects random number generators. In order to prevent generators from producing again the same random numbers, all random number generators are assigned new random seeds after jumping back.

Figure 9 illustrates the Check-Point-Restore concept as described above. Using this concept it was possible to reduce execution time by about 10%.

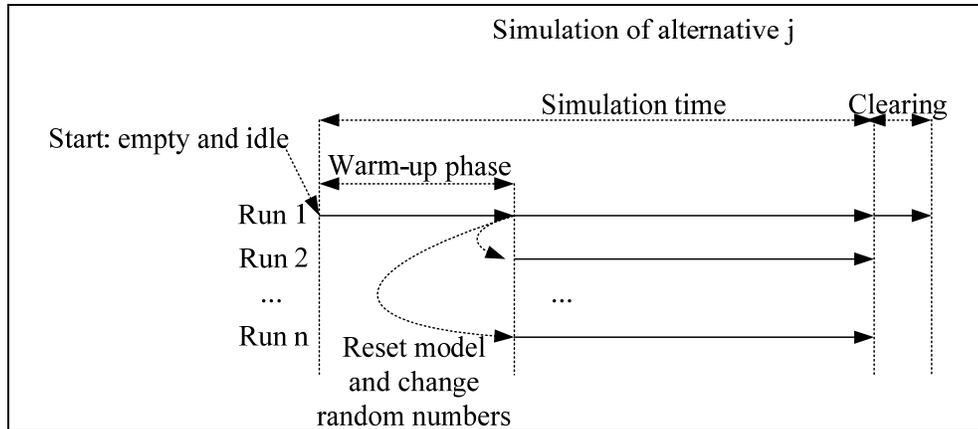


Figure 9: Check-Point-Restore concept

## 7 USER SUPPORT DURING OPTIMIZATION

In addition to a reasonable execution time, another key requirement is supporting the user during the optimization process (Wiedemann and Krug 2003) through automatic features like alternative evaluation, new alternative generation, and optimization termination. These features require deep knowledge of simulation and optimization, so it is preferable to hide them from the user.

The support system for optimization is integrated into the existing GUI for the SLX-based paint shop simulator. This supporting system is called “Optimization Commander”. The commander offers 3 possibilities to the user:

- Optimization configuration: The user has to specify the optimization targets (only one or both), and the target throughput for each carrier type. Moreover, the user has the possibility to define the minimum length of specific paths and he/she can specify detailed configuration settings for confidence interval calculation, like the confidence level.
- Information during optimization: The user gets visual feedback about the optimization process via charts, which show the current number of carriers and the current total length of the system. A log-file is also generated to log the optimization process in detail.
- Final information: Optimization results are automatically compared to the initial system settings and differences are highlighted. Results can be exported to Microsoft Excel for further analysis. Moreover, the user gets direct feedback about the potential savings in terms of money and also a comparison between the investment costs for the initial system and the final one.

## 8 CASE STUDIES

Table 2 summarizes simulation-based optimization results for seven different paint shop simulation models. The table is made of two blocks:

- The first block displays general information about the models: number of paths, total conveyor length, number of different carrier types, and total number of carriers.
- The second block displays information about the optimization process: number of tested alternatives, runs needed to evaluate the alternatives, reduced length and number of carriers, and optimization execution time.

Table 2: Overview of Case Studies

Model ID	1	2	3	4	5	6	7
Number of paths	20	38	41	145	46	56	66
Start total length [m]	231	1343	739	2401	315	384	399
Number of carrier types	1	2	2	5	1	1	1
Start total number of carriers	15	73	26	171	60	80	83
Number of tested alternatives	19	56	25	221	88	64	64
Number of executed runs	205	665	250	2457	933	646	660
Reduced length [m]	91	55	104	1018	131	100	90
Reduced number of carriers	1	21	3	30	18	16	5
Execution time [min]	6.77	26.06	14.33	953.03	113.59	151.13	148.04
Cost reduction [%]	22.09	6.17	9.62	28.97	21.52	15.21	10.06

The table shows that all models can be significantly improved. Execution times are fairly short, with the only exception of model 4, which has a high level of complexity. Cost reductions are not reported in the table due to confidentiality reasons.

## 9 CONCLUSIONS AND FUTURE WORK

This paper described simulation-based optimization of paint shops. The proposed heuristic optimization approach reduces investment costs by minimizing number of carriers and path lengths. The optimization is implemented as an extension to an existing generic simulation model for paint shops. However, the proposed approach can be applied to similar modeling techniques and factory-planning scenarios. In addition the optimization framework allows adding other optimization algorithms and objectives with minor effort. A user support system is presented, which helps users in conducting optimization-related tasks and interpreting results. The functionality and effectiveness of the implemented system is tested on a set of case studies.

Future work will focus on two main topics:

- Visualization of optimization results, and
- Automatic comparison of investment costs between any pair of simulated alternatives.

## ACKNOWLEDGEMENT

The authors would like to thank Jakob Hölzler and Jan Steinfurth for all their interesting discussions, valuable comments, development work and research activities, and Valnea Pinezic for reviewing the English language of this paper.

## REFERENCES

Grewal, C., S. Enns, and P. Rogers. 2009. "Performance Effects of Setup Time Reduction without Decision Variable Re-Optimization: A Simulation-Optimization Study." In *Proceedings of the 2009 Win-*

- ter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, 2135-2144. Austin, TX. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.
- Gürkan, G. 2000. "Simulation Optimization of Buffer Allocations in Production Lines with Unreliable Machines." *Annals of Operations Research* 3(1-4):177-216.
- Henriksen, J. O. 2011. "Wolverine Web." Accessed March 4, 2011. <http://www.wolverinesoftware.com>.
- Henriksen, J. O. 1989. *The GPSS/H Reference Manual*. Wolverine Software Corporation.
- Hong, L. J., and B. L. Nelson. 2009. "A Brief Introduction to Optimization via Simulation." In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, 75-84. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.
- Kohn, R., D. Noack, M. Mosinski, Z. Zhou, and O. Rose. 2009. "Evaluation of Modeling, Simulation and Optimization Approaches for Work Flow Management in Semiconductor Manufacturing." In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, R. G. Ingalls, 1592-1600. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.
- Moeller, A., M. Prox, M. Schmidt, and H. Lambrecht. 2009. "Simulation and Optimization of Material Flow Systems." In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, R. G. Ingalls, 1444-1455. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.
- Weigert, G., T. Henlich, and A. Klemmt. 2008. "Methoden zur Modellierung und Optimierung von Montageprozessen." In *Advances in Simulation for Production and Logistics Applications*, edited by M. Rabe, 479-488. Stuttgart. Fraunhofer IRB-Verlag.
- Wiedemann, T., and W. Krug. 2003. "Actual and Future Options of Simulation and Optimization." In *Manufacturing, Organization and Logistics. Proceedings 15th European Simulation Symposium*, 627-637. Delft: SCS Europe BVBA.
- Williams, E., and S. Sadakane. 1997. "Simulation of a Paint Shop Power and Free Line." In *Proceedings of the 1997 Winter Simulation Conference*, edited by S. Andradóttir, K. J. Healy, D. H. Withers, B. L. Nelson, 727-732. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

## AUTHOR BIOGRAPHIES

**MARCO LEMESSI** is leading simulation projects for Deere & Company in Europe, North Africa, Near and Middle East, and is the main developer of Deere-owned simulation models for paint, machining, and welding systems. He received his Ph.D. (2002) in Traffic Engineering and M.S. (1998) in Civil Engineering from the University of Rome "La Sapienza", Italy. His email address is [LemessiMarco@JohnDeere.com](mailto:LemessiMarco@JohnDeere.com).

**THOMAS SCHULZE** is a professor in the School of Computer Science at the Otto-von-Guericke-University, Magdeburg, Germany. He received the Ph.D. degree in civil engineering in 1979 and his habil. Degree for computer science in 1991 from the University of Magdeburg. His research interests include modeling methodology, public systems modeling, manufacturing simulation, distributed simulation with HLA and online simulation. He is an active member in the ASIM, the German organization of simulation. His web page can be found via <http://www.ums.ovgu.de>.

**SIMEON REHBEIN** is a fifth-year student at the Otto-von-Guericke-University Magdeburg, Germany. His major for the diploma is industrial engineering with a focus on logistics. During his university education he has undertaken several internships and part-time jobs in the area of simulation and application development. His e-mail address is [mail@SimeonRehbein.de](mailto:mail@SimeonRehbein.de).