

## SEMI-AUTOMATIC SIMULATION-BASED BOTTLENECK DETECTION APPROACH

Marco Lemessi  
Simeon Rehbein

John Deere GmbH & Co. KG  
Mannheim Regional Center  
John-Deere-Straße 70  
Mannheim, 68163, GERMANY

Gordon Rehn

Operations Simulation Consultant  
404 Cedar Street  
Andover, IL 61233, USA

Thomas Schulze

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

### ABSTRACT

This approach combines in a semi-automatic way known simulation-based bottleneck detection methods. It considers the integration of these methods into the simulation, significantly influencing execution speed and acceptance of the industrial environment. Even if the majority of detection tasks are automatically driven some user interaction is needed to find the bottlenecks. The paper describes common bottleneck definitions; already published bottleneck detection methods; and deployment of the new approach. The approach consists of a two-step procedure, first analyzing the system, and then generating scenarios testing the system's sensitivity against changes. Based on the scenarios, the bottleneck is derived. The applicability of the approach is discussed on a real-world paint shop system and items limiting system performance are identified.

### 1 INTRODUCTION

Bottlenecks are everywhere and become increasingly more important by playing a significant role in understanding the environment we live in. Understanding the environment provides opportunities to gain a competitive business advantage in improving market position. Understanding always relates to planning, and planning theory has been developing constantly since the 1980's (Theory of Constraints, Lean Manufacturing, Six Sigma) (Burton-Houle 2001). They attempt to organize systems to provide maximum benefit for all participants. Since bottlenecks limit performance it is important to identify and eliminate these limitations to maximize utilization of existing equipment and reduce investment costs (Williams and Sadakane 1997).

Manufacturing systems influence the overall design of a factory and hence a special focus must be placed on systems during planning. This is often done by *simulation*, which provides the possibility to evaluate systems prior to realization and investment of capital. Simulation also provides the possibility to use a dynamic model perspective and use random events to describe production processes more realistically (Law and Kelton 2000). In combination with bottleneck detection, simulation can help improve the overall manufacturing process and create a better work flow resulting in more throughput with less investment. During the last decades, research has been done to find bottlenecks in manufacturing systems by using simulation (Heinicke and Hickmann 2000, Roser, Nakano and Tanaka

2003, Sengupta, Kanchan and Van Til 2008). That research focused most on output-analysis and calculating coefficients which try to indicate the bottlenecks. Output-analysis was used to study systems in terms of coefficients like utilizations (Lima, Chwif and Barreto 2008). However, these coefficients are only indicators of potential bottlenecks without proving that the bottleneck is the true system constraint. This is one limitation of the existing approaches. Another limitation is current methods require manual user interaction and interpretation, and are unable to detect bottlenecks directly. These two limitations restrict the actual usage of bottleneck detection to experts. The new semi-automatic bottleneck detection approach is developed to overcome these restrictions. The basic idea of the new approach is an automatic integration of simulation in the bottleneck detection process, with a limited amount of user interaction. Moreover, the success and efficiency to detect bottlenecks using the new approach depends on the architecture of integrating it into simulation. This relates mainly to the used simulator. The presented approach considers this challenge by being generic and useable with any available commercial simulator. The duration of bottleneck detection depends on the needed simulation execution time.

The presented example uses SLX (Simulation Language with eXtensibility) from Wolverine Software Corporation as the simulation engine. The approach combines different published bottleneck detection methods into one approach and tries to automate as many bottleneck detection related tasks as possible. The approach also proves if a detected bottleneck is really the system constraint.

The remainder of this paper is structured as follows: Section 2 focuses on defining the term bottleneck and shows reasons for bottlenecks within manufacturing systems. Section 3 shows the existing research on simulation-based bottleneck detections methods. Section 4 describes the new presented bottleneck detection approach and Section 5 provides an example for using the approach.

## **2 REASONS FOR BOTTLENECKS**

(Lima, Chwif and Barreto 2008) define a bottleneck, also called constraint, as the root of system's performance problem. Moreover, the term performance means how agile a system works considering its goals. As for manufacturing systems, the goal of a factory is to make money and consequently produce products to sell at a profit (Goldratt and Cox 2010). Hence performance can be measured in money from sold products or in throughput per time period.

The term bottleneck is defined in two different ways. The first way takes into account the potential performance increase a bottleneck can provide when it gets solved (Li et al. 2007). Bottlenecks are those system's items which increases performance when the item's negative impact is reduced or eliminated. The second way focuses on the limitations a bottleneck creates for the system (Roser, Nakano and Tanaka 2003). Hence a bottleneck is a production stage that has the largest effect on slowing down or stopping the entire system.

In addition to these definitions, a variety of other definitions exist. Roser, Nakano and Tanaka distinguish within production networks between **primary**, **secondary** and **non-bottlenecks**. A primary bottleneck has the largest effect on the system. Secondary bottlenecks limit the system's performance as well, but to a fewer extent. Non-bottlenecks do not have any influence on system's performance (Roser, Nakano and Tanaka 2001).

Also defined are **static** and **dynamic** bottlenecks, considering the time when they occur. Dynamic bottlenecks change over time, influencing a system during a specific time frame, while static bottlenecks are influencing the system all the time (Lima, Chwif and Barreto 2008).

Related to this definition a **momentary** and **average** bottleneck can be differentiated. Bottlenecks at a specific time are called momentary, but average bottlenecks relate to all momentary bottlenecks and consider the most significant one as the primary factor of system's limitation (Roser, Nakano and Tanaka 2002).

Taking the definitions into account a queuing system can be studied with regard to two main coefficients. First the arrival rate  $\lambda$ , which describes how many products arrive at the system during a time frame. Second the service rate  $\mu$ , which describes how many products can be processed by the system

during a time frame. As soon as the arrival rate becomes greater than the service rate, inventory gets created in front of the system, and the system is unable to keep pace. Hence a bottleneck exists.

However, there are also situations in which the arrival rate is only occasionally higher. This can be due to variation along the process (Sengupta, Kanchan and Van Til 2008). Variation can be defined as the quality of non-uniformity of items. For example, products which do not have exactly the same weight, or processes which vary in terms of time (Hopp and Spearman 2008). In some cases, both the arrival and the service rate can vary over time, impacting the dynamic system capacity. Capacity means the maximum throughput of an item e.g. machine (Hopp and Spearman 2008). Figure 1 shows an example for such a situation. This figure shows that the capacity of the three different machines varies and hence the bottleneck shifts along the process and with time. Here a process is a sequence of tasks which must be done on products at different machines. In Figure 1, the numbered rectangles show the bottlenecks at different times. At the very beginning machine 2 is the bottleneck, but later machine 1 becomes the constraint. However, machine 1 is limiting the system for the longest uninterrupted time (see bottleneck 4 in Figure 1). Since the capacities change, no clear bottleneck can be identified.

Based on these interferences and stochastic events, bottleneck detection is challenging. As a result, static analysis is unable to detect bottlenecks and hence simulation must be used to identify the most significant bottlenecks.

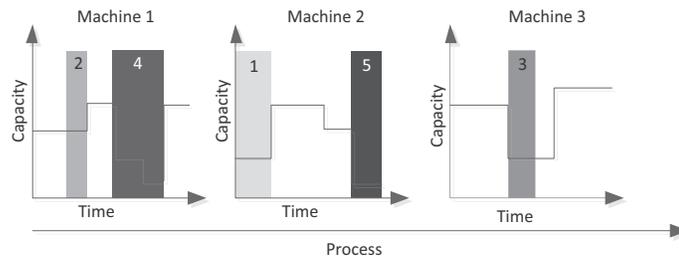


Figure 1: Dynamic Bottleneck Behavior

### 3 EXISTING SIMULATION-BASED BOTTLENECK DETECTION METHODS

#### 3.1 Overview

Analytical bottleneck detection methods model systems in a deterministic and often static way. This level of detail is suitable for long-term bottleneck prediction, but can be inadequate in the short term when a higher level of detail is needed (Leporis and Králová 2012). Improving complex and dynamic manufacturing processes by bottleneck detection requires a high level of detail (down to the technicians and machines) to predict the bottleneck in an accurate way. This enhanced level of detail can be provided by simulation. Moreover, most simulators can generate standard information about system's performance also related to bottleneck detection, such as utilization, break downs and waiting times (Leporis and Králová 2012).

The existing simulation-based bottleneck detection methods can be grouped into three categories: 1. static-calculation methods, 2. coefficient-based methods, and 3. scenario-based methods. The first category analyzes simulation input data to identify the bottleneck, the second category analyzes simulation output data based on coefficients, and the third category performs simulation scenarios to evaluate the sensitivity of different model items. A scenario is defined as a modification to an existing initial simulation model.

#### 3.2 Static-Calculation Method

One simple way to detect bottlenecks is static and deterministic calculations by using input data provided for simulation. This is a form of input-data analysis and offers an overview about the overall system

configuration (Law and Kelton 2000). Also, such calculations are often used for validation and verification purposes (Wenzel et al. 2007). Based on these calculations, critical system items can be predicted.

Given the known demand a manufacturing system must meet during a time period (e.g., 1000 produced items during 10 weeks) the system's takt time can be calculated. By further simplifying system's items into single servers, the system process time can be calculated and compared to takt. If the takt time is greater than the process time, the item is not the bottleneck. However, if the process time exceeds the takt time, an item is unable to keep pace with the takt and limits the overall system. Such simple calculations can identify possible bottlenecks. If no item has a process time greater than the takt time, process times approaching the takt time can be identified as critical based on system's variation.

### 3.3 Coefficient-Based Method

Unlike static calculations, coefficient-based simulation bottleneck detection is based on simulation output data. Simulation calculates coefficients to evaluate manufacturing system's performance. Different coefficients have been utilized in the past.

#### 3.3.1 Utilization-Method

One of the most common coefficients to find bottlenecks is the utilization of items, such as machines and technicians. This method *measures the percentage of time a machine is active* (Roser, Nakano and Tanaka 2003) to detect bottlenecks. The item with the highest percentage in the overall system is assumed to be the limiting factor in the system (Lima, Chwif and Barreto 2008). The utilization-method needs to identify the time when an item's state changes (Faget, Eriksson and Herrmann 2005): (1) Inactive state, when an item is waiting for new tasks or it is blocked due to downstream blockage; (2) Active state, when an item is performing a function. Moreover (Faget, Eriksson and Herrmann 2005) highlighted in detail that *the machine with the longest average active period is considered to be the bottleneck, as this machine is least likely to be interrupted by other machines, and in turn is most likely to dictate the overall system throughput.*

#### 3.3.2 Waiting-Time-Method

Another method considers the cumulative time products must wait for the availability of a specific item (e.g., a machine). The item, which has the highest cumulative waiting time of all items in the manufacturing system, is assumed to be the overall system constraint (Roser, Nakano and Tanaka 2003, Lima, Chwif and Barreto 2008). When using this method two assumptions must be made (Roser, Nakano and Tanaka 2003). First, this coefficient is only valid for linear-connected items, without multiple product types, since multiple product types can lead to occasions *where a machine with a few parts being processed slowly constrain the system more than a machine with a lot of parts being processed quickly* (Roser, Nakano and Tanaka 2003). Second, this method only works when the queues ahead of the items have an infinite capacity; otherwise, one item could lead to a blockade of another item upstream, when its queue is full (Roser, Nakano and Tanaka 2003).

#### 3.3.3 Inter-Departure-Time-Method

The inter-departure-time-method is similar to utilization-method, however, it analyzes inter-departure data. The method defines four different states of an item to find the bottleneck (e.g., a machine) (Sengupta, Kanchan and Van Til 2008): 1. Cycle (item is performing), 2. Blocked-Down (finished product cannot leave item), 3. Blocked-Up (item is waiting for products), 4. Fail (item is broken).

The method identifies the bottleneck as an item which is least affected by other items in the system. When analyzing the different states it is obvious that a fast item fed by a slow item is often in blocked-up

state. This is also true when a fast item feeds a slow item; in this case the item is often in the blocked-down state. Hence (Sengupta, Kanchan and Van Til 2008) suggested calling an item a bottleneck when it has the minimum combined percentage of time being in the blocked-up and blocked-down state. In this case, the item is not frequently influenced by other items, but it most likely influences other items.

### 3.3.4 Shifting-Bottleneck-Method

Also well-known is the shifting-bottleneck-method for bottleneck detection (Roser, Nakano and Tanaka 2002, Sengupta, Kanchan and Van Til 2008, Lima, Chwif and Barreto 2008). As a first step, system items must be grouped into two different states as in the utilization-method. Two different types of bottlenecks are found here (Roser, Nakano and Tanaka 2001):

- **Momentary bottleneck:** The item with the longest uninterrupted active period at the specific time. Furthermore (Roser, Nakano and Tanaka 2002) explained that *the overlap of the active period of a bottleneck with the previous or subsequent bottleneck represents the shifting of the bottleneck from one machine to another machine*. Hence, the longer the uninterrupted time of an item, the more likely it is that the specific item constraints and influences other items.
- **Average bottleneck** (Roser, Nakano and Tanaka 2002): The sum of sole-bottleneck and shifting-bottleneck time is calculated to identify the bottleneck. The item with the highest sum is assumed to be the constraint.

### 3.4 Scenario-Based Method

Scenario-based bottleneck detection methods use simulation's capability to answer *What-If* questions and test different model scenarios (Pawlewski and Fertsch 2010). The concept can be defined as follows: *The item with largest sensitivity of system's performance index to this item's production rate in isolation is defined as the bottleneck item* (Sengupta, Kanchan and Van Til 2008). So this method changes the simulation model to find the bottleneck. According to (Li et al. 2007) the first step of this method is to identify critical model items, which can influence and constrain the overall performance. Items in this case can be, for example, the number of technicians or the number of machines.

Next, different sets of scenarios, for each critical item, are defined where the capacity of the different model items is changed. For example the number of technicians on a specific machine could be increased. After the definition, the different scenarios are simulated and the resulting performances are compared. The scenario which provides the biggest increase in performance is selected and the underlying change is called the bottleneck.

When using this method it must be considered whether a bottleneck can be found depends exclusively on the pre-defined scenarios. It is important to find all possible model items which can constrain the system. Another consideration is that the scenarios can only be compared if the changes in the scenarios are comparable. For instance, if scenario A increases the number of technicians by one unit and scenario B increases the number of machines by one hundred units, it is likely that B raises production more than A, but that does not mean that the number of machines is the bottleneck. This is because the increases are not comparable. Besides that it must be also considered how many model items are changed in one scenario, which relates to design of scenarios and sensitivity analysis. More information on that can be found in (Law and Kelton 2000).

## 4 SEMI-AUTOMATIC SIMULATION-BASED BOTTLENECK DETECTION APPROACH

An overview of the approach can be found in Figure 3. It is a cyclic approach and in each cycle two sequential phases are performed. The first phase is called *analysis-phase* and provides an overview about the system's behavior. The second phase is called *detection-phase* and identifies possible bottlenecks based on performed scenarios.

### 4.1 Analysis Phase and Detection Phase

The analysis-phase starts with simulating the initial model. Different coefficients, similar to the ones presented in Section 3.3, are calculated based on simulation outputs.

Next the detection-phase automatically generates the scenarios based on different coefficients. For example, if the utilization coefficient of an item (e.g., a machine) is greater than a pre-defined value, the detection-phase will generate a scenario in which the item’s capacity is increased. Equation 1 shows how, for each and every machine, the utilization is calculated and compared to a pre-defined value. If the utilization is greater than or equal to the pre-defined value, a scenario is generated. That means for each and every critical item, which could be a bottleneck (i.e., any machine or technician), a set of different scenarios is built. As soon as all scenarios are generated, the scenarios are simulated and their performances compared. The scenario which provides the highest performance increase is chosen and its underlying change is declared as bottleneck (as shown in Equation 1 for machine utilization). In addition to that, the user has also the possibility to build scenarios on his/her own (semi-automatic approach). Then the underlying change is used to modify the initial simulation model and the approach starts again with the analysis-phase. The approach ends as soon as no scenario can create an additional performance increase. In this case, the bottleneck is out of the scope of the defined scenarios.

$$Machine\ Utilization_j = \frac{Working\ Time}{Time\ Machine\ Switched\ On} \geq Value, \forall j = 1 \dots m \tag{1}$$

### 4.2 Balancing Algorithm

The balancing algorithm applies only if all the scenarios are comparable, as explained in Section 3.4. This algorithm tries to find the maximum possible performance increase a specific scenario can generate. For example, if one scenario considers a specific machine, the algorithm tries to find the machine’s capacity which provides the maximum system performance increase. In this example, the algorithm changes the machine’s capacity multiple times (consequently multiple simulations are executed) until it finds the “best” capacity. Hence a scenario is more than just one modification to the initial model, but what change to its item provides maximum performance increase is identified. This is done for all scenarios comparing the different performances. In this case, scenarios’ performances represent the potential of scenarios’ underlying change to improve the system.

Use of the balancing algorithm is shown in Figure 4. The algorithm assumes that the maximum system performance, provided a specific scenario item, can be found by increasing item’s capacity (e.g., machine capacity) multiple times and analyzing when an increase of the capacity does not lead to an increase to system’s performance anymore. In this case it is assumed that the maximum system performance of the scenario is found. However, the number of changes is limited to avoid long algorithm execution times.

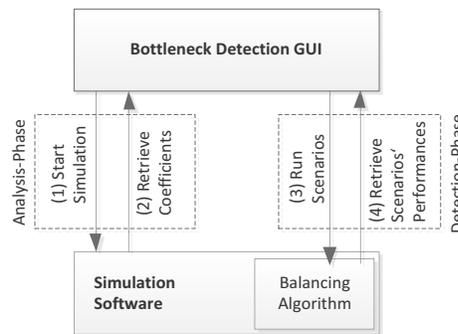


Figure 2: Approach Architecture

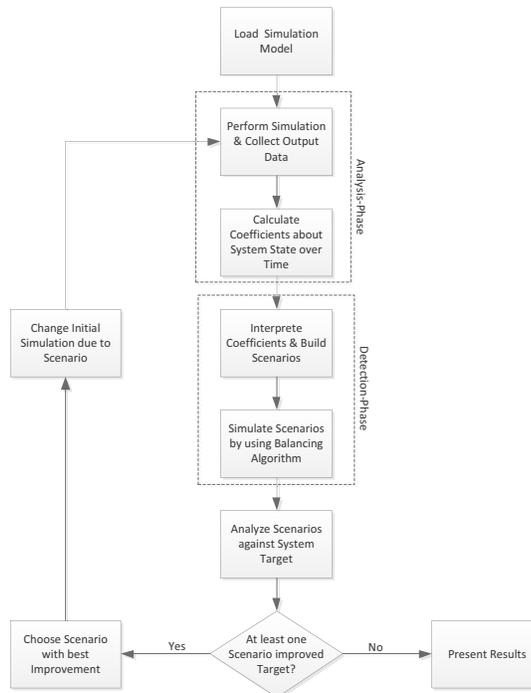


Figure 3: Bottleneck Detection Approach

Three different pieces of information are passed to the balancing algorithm: 1. The item which shall be changed. 2. The number of changes done to the item. 3. The increase to item in each change.

When all information is provided the algorithm works as shown in Figure 4. For each iteration, the algorithm changes the item's capacity by the provided increase. The scenario is simulated and the performance of the scenario in the specific change is saved. Once all changes are executed, the algorithm returns the maximum possible performance increase by this scenario.

The performance of a simple balancing algorithm depends on the increase value used as well as number of changes. More sophisticated balancing algorithms could be used to increase the overall efficiency of the approach. Moreover, it is assumed that the bottleneck can be found by dealing with one change and ignoring multiple changes to different model items in one scenario.

As described, the analysis and detection-phase are performed automatically, referring to calculating system coefficients and also building and executing scenarios. Nonetheless, manual preparations are needed to configure the approach for a specific field of application (e.g., painting, machining or assembling). The preparation must be done once per field and can be reused in a generic way. The following manual steps must be performed:

- *Find System Target*: The overall target of the analysis must be defined and a main coefficient of the system performance must be found (e.g., throughput of parts).
- *Find possible Bottlenecks*: Analyze what system items could generate bottlenecks, such as any machine or technician.
- *Find representative Coefficients*: Based on the possible bottlenecks, coefficients must be found to detect such bottlenecks (e.g., high machine utilizations).
- *Define generic Scenarios*: Map discovered coefficients and possible bottlenecks and generate model changes to build a scenario. Generic means general scenarios must be defined hypothetically, for example a machine or a technician in the system could be the bottleneck, and the performance could improve if the number of items is increased. During the execution of the approach the generic scenarios become more specific, for example machine 1 or technician 3 could be the bottleneck and their capacity gets extended. The mapping of possible bottlenecks

and coefficients also relates to defining bounds for the coefficients. The bounds indicate whether a scenario shall be generated (e.g. when machine’s utilization is greater than a defined bound).

- *Balance Scenarios*: Make sure that changes of the scenarios are comparable to each other with regard to the system target. This is done by the explained balancing algorithm. However, such an algorithm must be defined for each and every possible scenario.

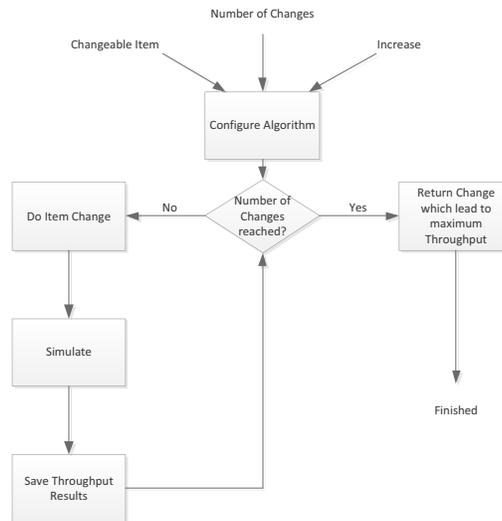


Figure 4: Balancing Algorithm

The presented approach combines multiple aspects. In reference to the bottleneck definitions it considers both ways of defining a bottleneck: The first way, which considers the additional performance a bottleneck can provide (see Section 2), is used within the scenario concept. The second way, which focuses on the limitations a bottleneck causes to the system (see Section 2), is taken into account by calculating the different coefficients.

Finally, the approach also combines the different categories of simulation-based bottleneck detection methods. The coefficient-based analysis is combined with the scenario-method. As a result, the approach attains a higher performance compared to just performing scenarios, without deep system knowledge. It also proves if a possible bottleneck really constrains the system, unlike the coefficient-based methods.

The new approach is simulator-independent. However, the simulation system used must have specific capabilities. First, the simulator must be able to generate the needed output coefficients. Most simulators provide simple coefficient outputs by default. However, coefficients like “longest active machine at a time” used in the shifting-bottleneck-method could be more challenging. Second, the simulator must be able to handle multiple scenarios and change the model automatically; otherwise scenario-usage is not possible. Third, the simulation speed is significant, since all scenarios are simulated with regard to confidence intervals. Hence, each change in one scenario must be simulated in multiple runs.

The architecture for the implementation is shown in Figure 2. An external bottleneck detection GUI is used to coordinate all needed approach-steps and informs the user about bottleneck detections. The GUI starts approach’s analysis-phase by launching the simulation and also retrieving the coefficients. Based on the coefficients the GUI builds scenarios and starts the simulation again. In this case the detection-phase is started and the balancing algorithm within the simulation finds for each scenario its maximum performance. Upon completion of the scenario tests, the GUI retrieves the scenarios’ performances and compares those to identify the bottleneck.

## 5 APPLICATION EXAMPLE

The bottleneck detection approach is demonstrated using a paint shop application. 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 generic simulation model consists of a flow graph with directional links and nodes. 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 inter-arrival times).

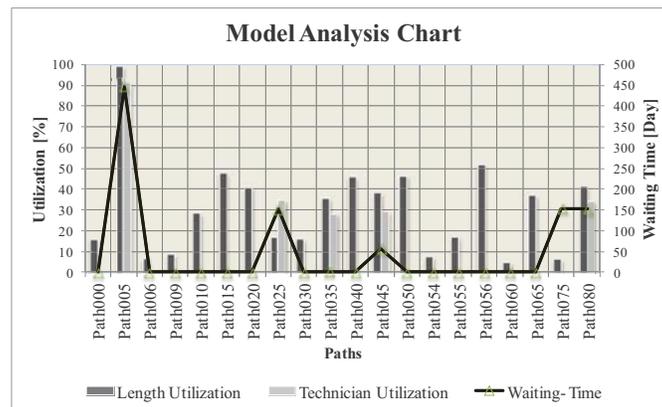


Figure 5: Results for Analysis-Phase

Nodes in the network are referred to as “stops” and links as “paths”. A conveyor consists of a number of paths and moves carriers at a specific speed. At any stop a station can be located where different tasks (e.g., loading or unloading parts on carriers) or specific value-added processes (e.g., heating parts) can be executed. The tasks are executed by technicians. Carriers move along paths, where equipment (e.g., ovens or washers) might be located.

As mentioned in Section 4, manual preparation is needed to use the approach. The following preparation steps are performed to use the approach in the field of paint shops. As soon as these steps are performed they can be reused on any paint shop type.

- *Find System Target:* The performance of the simulation model is measured in throughput: unloaded carriers during a week.
- *Find possible Bottlenecks:* Manual system analysis provides the following possible bottlenecks: (1) The number of carriers in the system is too few, (2) The capacity of a station in terms of number of technicians is too small, (3) Length of a path between two different stations is too small to support the necessary system flow rate, (4) The conveyor speed is too slow.
- *Find representative Coefficients:* The following coefficients can be used to predict if any of the four bottleneck types exist: (1) The number of carriers in the system is increased to determine if a carrier shortage exist, (2) High technician utilization indicates that more technicians may be needed, (3) High path utilization indicates either the path length/capacity is too small, or the following paths are a constraint, (4) The slowest conveyor speed in the system is a potential bottleneck. In addition, also (5) the waiting time (see Section 3.3.2) of a path indicates a too small queue between two stations.
- *Define generic Scenarios:* Possible station bottlenecks are tested if an increase in capacity leads to more performance. Possible path bottlenecks are tested if an increase of length generates more performance. Conveyor speeds are analyzed if an increase leads to a higher performance, and

similarly, if an increase of the number of carriers leads to more performance. The scenarios are generated if either path or technician utilization is higher than a pre-defined value. The number of carriers and the conveyor with the slowest speed is always considered. Similarly, the station with the highest joining time creates another potential scenario.

- **Balance Scenarios:** The scenarios are balanced by the algorithm shown in Figure 4. For the four different scenarios the specific changeable item is used as one input (e.g., a specific path). Moreover the increase of a specific item is set to its smallest possible increase (such as increasing carriers by one unit). Station capacity can be increased by decreasing its work times in decrements of 10%, similarly the speed of the slowest conveyor is increased by 10%. The number of changes is set to 10 changes for each scenario.

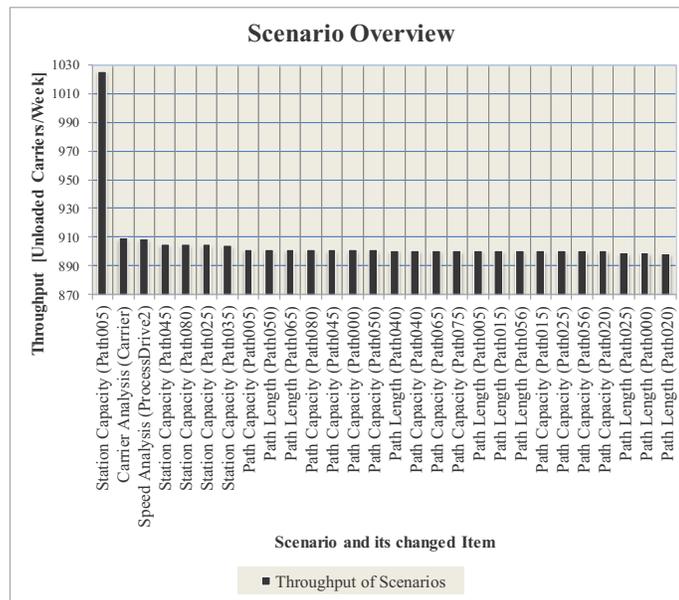


Figure 6: Results for Detection-Phase

The example system consists of a closed loop where different value-added tasks are executed, such as loading and unloading of carriers, pretreatment, and paint processes.

The following results refer to the first cycle through the analysis- and detection-phase of the approach. As explained, the approach executes multiple cycles, but for demonstration purposes just the first one is shown here.

When executing the bottleneck detection approach, the analysis-step presents the following results about the defined system coefficients, without changing the system (see Figure 5). The figure shows on its X-Axis the different paths defined in the model. On the left Y-Axis, the utilization is shown and on the right Y-Axis, the cumulative waiting time of carriers at stations. Three different coefficients are presented: the path utilization; the utilization of the technicians working on the stations; and the waiting time at a station (other coefficients mentioned above are not shown due to clarity reasons). Path005 is identified as a path which has the highest values for all three coefficients. Based on these results it could be guessed that Path005 is the overall bottleneck. However, there is no specific proof for it. That is the reason why different scenarios are created to test multiple system items, as explained in Section 4. The results of the detection-phase and its scenarios can be found in Figure 6, now with a changed system. The figure shows on its X-Axis the different scenarios and its Y-Axis the reached throughput (e.g., the first scenario increased the station’s capacity). It is visible that the change to Path005 led to a significant throughput increase (it has the highest sensitivity to the throughput compared to the other scenarios) and it is possibly the strongest bottleneck inside the system. From a system point of view this result is

interesting, since at the end of Path005 is the loading station for carriers located. By increasing the capacity of this station the model is able to increase throughput. Based on this information the factory is able to increase its throughput by expanding station's capacity. The approach tells how to generate more throughput and focus on the most critical system items.

## 6 CONCLUSIONS AND FUTURE WORK

The paper presents a semi-automatic simulation-based bottleneck detection approach which integrates previously defined bottleneck detection methods into one approach. Unlike existing methods, this approach is able to identify and prove bottlenecks based on the simulation model and ensures a higher data validity. This enables factories to plan and expand their systems in a more precise way. In addition the presented approach can be used in other flow-driven systems, due to its generic design, considering coefficients and scenarios.

However, this approach has some limitations. The first limitation assumes that iterative changes are possible and the changes tested in the scenarios are physically possible in the actual system. Additional research is needed to assure reality. Moreover, the scenarios are based on just one model item, but bottlenecks can refer to multiple combinations of interrelated items. Future research is needed to further develop this consideration. Finally, this approach will be expanded to other manufacturing systems, such as machining or assembly lines.

## REFERENCES

- Burton-Houle, T. 2001. *The Theory of Constraints and Its Thinking Processes. A Brief Introduction to TOC*. The Goldratt Institute. New Haven, USA: The Goldratt Institute.
- Faget, P., U. Eriksson and F. Herrmann. 2005. "Applying Discrete Event Simulation and an Automated Bottleneck Analysis as an Aid to Detect Running Production Constraints." In *Proceedings of the 2005 Winter Simulation Conference*, Edited by M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J.A. Joines, 1401–1407. Orlando, FL.
- Goldratt, E. M. and J. Cox. 2010. *Das Ziel*. Frankfurt am Main: Campus-Verlag.
- Heinicke, M. U. and A. Hickmann. 2000. "Eliminate Bottlenecks with Integrated Analysis Tools in EM-Plant." In *Proceedings of the 2000 Winter Simulation Conference*, Edited by J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 229–231. Orlando, FL.
- Hopp, W. J. and M. L. Spearman. 2008. *Factory Physics*. Boston, M: Mcgraw-Hill Higher Education.
- Law, A. M. and W. D. Kelton. 2000. *Simulation Modeling and Analysis*. Boston, M.: Mcgraw-Hill Higher Education.
- Leporis, M. and Z. Králová. 2012. "A Simulation Approach to Production Line Bottleneck Analysis." In *International Conference CYBERNETICS AND INFORMATICS, VYŠNÁ BOCA, Slovak Republic*, 1-10. VYŠNÁ BOCA.
- Li, L., Q. Chang, J. Ni, G. Xiao, and S. Biller. 2007. "Bottleneck Detection of Manufacturing Systems Using Data Driven Method." In *Proceedings of the 2007 IEEE, International Symposium on Assembly and Manufacturing*, 76–81. Michigan, SA.
- Lima, E., L. Chwif, and M. Barreto. 2008. "Methodology for Selecting the Best Suitable Bottleneck Detection Method." In *Proceedings of the 2008 Winter Simulation Conference*, Edited by S. J. Mason, R.R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler, 1746–1751. Miami, FL.
- Pawlewsik, P., and M. Fertsch. 2010. "Modeling and Simulation Method to Find and Eliminate Bottlenecks in Production Logistics System." In *Proceedings of the 2010 Winter Simulation Conference*, Edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, 1547–1555. Baltimore, Maryland.
- Roser, C., M. Nakano, and M. Tanaka. 2001. "A Practical Bottleneck Detection Method." In *Proceedings of the 2001 Winter Simulation Conference*, Edited by B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 949–953. Arlington, VA.

- Roser, C., M. Nakano, and M. Tanaka. 2002. "Shifting Bottleneck Detection." In *Proceedings of the 2002 Winter Simulation Conference*, Edited by E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, 1079–1086. San Diego, California.
- Roser, C., M. Nakano, and M. Tanaka. 2003. "Comparison of Bottleneck Detection Methods for AGV Systems." In *Proceedings of the 2003 Winter Simulation Conference*, Edited by S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, 1192–1198. New Orleans, Louisiana.
- Sengupta, S., D. Kanchan, R. Van Til. 2008 "A new Method for Bottleneck Detection." In *Proceedings of the 2008 Winter Simulation Conference*, Edited by S. J. Mason, R.R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler, 1746–1751. Miami, FL.
- Wenzel, S., M. Weiß, S. Collisi-Böhmer, H. Pitsch and O. Rose. 2007. *Qualitätskriterien für die Simulation in Produktion und Logistik. Planung und Durchführung von Simulationsstudien*. Berlin: Springer-Verlag.

## AUTHOR BIOGRAPHIES

**Marco Lemessi**, Ph.D. is the Operations Simulation Manager at Deere & Company and the main developer of Deere-owned generic 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).

**Simeon Rehbein** is part of the Deere & Company Operations Simulation team located in Mannheim, Germany. He develops and enhances Deere-owned generic simulation models with the focus of machining systems. Simeon studied industrial engineering for logistics at Otto-von-Guericke University Magdeburg and holds a Master in Industrial Engineering. His email address is [RehbeinSimeon@JohnDeere.com](mailto:RehbeinSimeon@JohnDeere.com).

**Gordon Rehn** retired from Deere & Company (Moline, IL) in December 2011 with over 38 years of manufacturing experience. He has a Bachelor's degree in Mechanical Engineering from Iowa State University (1976). He pioneered the first Deere applications of discrete event simulation for planning and designing manufacturing systems, and later led the corporate Operations Simulation Group while performing hundreds of simulation analysis projects. Before retiring, he held a Professional Engineering license (State of Illinois), and was a member of SME and IIE. He has authored and co-authored numerous papers and publications on Operations Simulation, and is currently negotiating an affiliation with an established consulting firm. His email address is [gdr\\_sim8\\_con@yahoo.com](mailto:gdr_sim8_con@yahoo.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>.