Proceedings of the 2015 Winter Simulation Conference
L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, eds.

SIMULATION-BASED MULTI-OBJECTIVE BOTTLENECK IMPROVEMENT: TOWARDS AN AUTOMATED TOOLSET FOR INDUSTRY

Jacob Bernedixen[1]
Amos H.C. Ng[1]

[1]Department of Engineering Science,
Production and Automation Engineering
University of Skövde
Skövde, SE-541 28, SWEDEN

Leif Pehrsson[1,2]
Tobias Antonsson[2]

[2]Volvo Car Corporation
Gothenburg, SE-405 31, SWEDEN

ABSTRACT

Manufacturing companies of today are under pressure to run their production most efficiently in order to sustain their competitiveness. Manufacturing systems usually have bottlenecks that impede their performance, and finding the causes of these constraints, or even identifying their locations, is not a straightforward task. SCORE (Simulation-based COnstraint REmoval) is a promising method for detecting and ranking bottlenecks of production systems, that utilizes simulation-based multi-objective optimization (SMO). However, formulating a real-world, large-scale industrial bottleneck analysis problem into a SMO problem using the SCORE-method manually include tedious and error-prone tasks that may prohibit manufacturing companies to benefit from it. This paper presents how the greater part of the manual tasks can be automated by introducing a new, generic way of defining improvements of production systems and illustrates how the simplified application of SCORE can assist manufacturing companies in identifying their production constraints.

1 INTRODUCTION

Manufacturing companies operate in a very competitive market, be it local or global. Keeping the performance of the production system as high as possible is a key factor in holding a competitive advantage against the competition and in the long run ensuring the survival of the company. Many present production systems have grown fairly large and have unfortunately become very complex in their processes. Usually when such a complex system is not performing optimally, it is not an easy task to identify the cause of this poor performance, i.e. finding the bottleneck of the system. Examples of both analytical and simulation-based techniques for identification of the bottleneck of a production system can be found in literature, including utilization of machines (Hopp and Spearman 2000), blocking and starving patterns (Kuo, Lim, and Meerkov 1996), data-driven approach (Li 2009), shifting bottleneck detection (Roser, Nakano, and Tanaka 2002), multiple bottlenecks (Aneja and Punnen 1999) as well as a method based on inter-departure time and failure cycle data (Sengupta, Das, and VanTil 2008). The simulation-based techniques are widely used in industry but they can be unreliable and at best they are able to pin-point the location of the bottleneck but not the actual cause of it. A new and promising bottleneck detection method called SCORE (Simulation-based COnstraint REmoval) has been proposed by Pehrsson (2013). It utilizes simulation-based multi-objective optimization (SMO) to identify and rank bottlenecks (i.e. not only the primary bottleneck but also the secondary and even lower-order bottlenecks are detected) while at the same time categorizing the causes of the bottlenecks. The use of SMO as a powerful tool for bottleneck detection has been further detailed by Ng, Bernedixen, and Pehrsson (2014)

based on the definition of bottleneck and improvability of production systems found in (Li and Meerkov 2009). The SCORE-method systematically relaxes constraints of the system (i.e. improvements are implemented to eliminate the constraints) while the performance of the system (most often throughput, but other performance measures can be used) is measured with the goal of maximizing the performance (see objective function 1 below) with as few improvements as possible (objective function 2). According to theory of constraints (TOC) (Goldratt 1997) the highest throughput improvement will be achieved by removing the most significant constraint. In the simplest form of SCORE-analysis, processing times, availabilities and repair times are considered as potential causes for bottlenecks. However, the method is applicable to a wider range of causes of bottlenecks that may include, e.g. quality deficiencies, lack of resources, and buffer issues. These parameters are added as optimization input variables with two levels, see (3), the *original value* and an *improved value* where the constraint is removed. By transforming these parameters to binary 0/1-variables as in (4), it is possible to calculate the value of the improvement objective (2). Thus the following optimization problem is at the heart of the SCORE-method.

$$max \quad Performance\ (e.g. Throughput) \tag{1}$$

$$min \quad \sum_{i=1}^{N} I_i \tag{2}$$

$$subject\ to \quad x_i \in \{original\_value_i, improved\_value_i\} \tag{3}$$

$$I_i \in \{0,1\},\ where \begin{cases} I_i = 0\ iff\ x_i = original\_value_i \\ I_i = 1\ iff\ x_i = improved\_value_i \end{cases} \tag{4}$$

$$i \in \{1, \dots, N\},\ where\ N\ is\ the\ total\ number\ of\ possible\ improvements$$

This optimization problem quickly grows with the size of the manufacturing system, for instance even in its simplest form (i.e. with only process times, availabilities and repair times as potential causes of bottlenecks) six variables are added per potential bottleneck location (workstation) in the manufacturing system. Setting up such an optimization problem manually might be feasible for small models but not for large and complex production systems found in industry. Not only would it be a very time-consuming and tedious task to set it up manually, it would also be very error-prone. The SCORE-method, as presented by Pehrsson (2013), lacks details on how the formulation of this optimization problem can be automated, or how a software should be designed to support it so that it can be widely adopted within industry.

In this paper we address this problem. We will show how the setup of the SCORE optimization problem can be automated to a large extent by introducing a generic way of defining improvements (SCORE Groups) in Section 2. In Section 3 these SCORE Groups will be used to perform a SCORE analysis on a small academic test model to further illustrate how these SCORE Groups can help with the formulation of the SCORE optimization problem. Section 4 will describe a successful application of this automation within industry and at the same time provide some motivation for the need of this type of analysis within industry. Conclusions and direction of future research are given in Section 5.

2 AUTOMATING THE SETUP OF THE SCORE OPTIMIZATION PROBLEM

SCORE optimization problems tend to, as mentioned earlier, have a vast amount of input variables and are not easy to set up manually. This section will present a way in which this task can be automated to a large extent and at the same time eliminate or at least significantly reduce the errors commonly introduced when performing this task manually. First a generic way of defining the input variables, see (3), is introduced and then we present how these inputs are transformed to another set of binary variables (4) that are used to formulate the improvement objective (2) of the SCORE optimization problem.

2.1 A Generic Way of Defining Improvements

Removal of system constraints is at the core of the SCORE analysis. Constraints are removed through actions that improve the values of the constraining parameters of the production system. Such an improvement action can be described as changing a system parameter from its *original value* to some *improved value*. The type of system parameter will determine whether an increase (e.g. the case for availability) or a decrease (e.g. the case for processing time) of the parameter value represents an improvement. The information needed to define a group of improvement actions (called a SCORE Group) is shown in Figure 1 and described in more detail in Table 1.

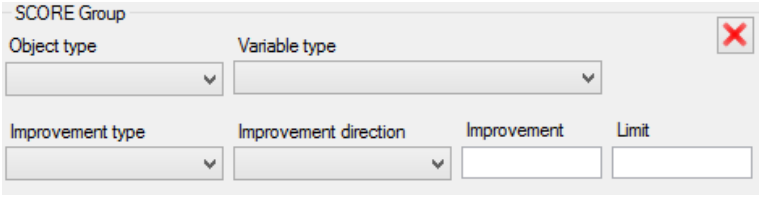

Figure 1: Dialog used for defining a SCORE Group.

Table 1: Detdetailed information about how groups of improvement actions (SCORE Groups) are defined using the dialog in Figure 1. The type of system parameter is determined by the two fields object type and variable type.

Field(-s)	Description
Object type	List of all object types of the modeled production system. Used to select what type of object this improvement applies to, e.g. an Operation, an Assembly station etc.
Variable type	List of all available variables of the selected object type. Used to select what type of system parameter this improvement applies to.
Improvement type	This will determine the type of improvement as one out of three types: • ABSOLUTE - The *improved value* is calculated as an absolute offset from the *original value*. • RELATIVE - The *improved value* is calculated as a relative offset from the *original value*. • FIXED - The *improved value* is fixed and independent of the *original value*.
Improvement direction	Determines what direction is considered an improvement for the selected type of system parameter. • POSITIVE - An increase of the *original value* is considered an improvement. • NEGATIVE - An decrease of the *original value* is considered an improvement.
Improvement	Parameter used to calculate the *improved value*. It's meaning is determined by the selected Improvement type: • Improvement type = ABSOLUTE - This parameter is the absolute offset used to calculate the *improved value*. • Improvement type = RELATIVE - This parameter is a percentage value used to calculate the size of the improvement as a percentage of the *original value*. • Improvement type = FIXED - This parameter is the actual *improved value*.
Limit	Optional parameter used to limit the improvement and possibly exclude system parameters from improvement. If the *improved value* is better than this limit it will be capped to this limit parameter. On the other hand if the *original value* is already better than this limit the system parameter will not be added as an improvement.

With the information specified in a SCORE Group it is possible to automatically retrieve a set of system parameters and add them as binary multiple-choice set (MCS) (Bernedixen and Ng 2014) input variables to a SCORE optimization problem. These input variables will take either the *original value* or the *improved value* and the corresponding MCS notation is {*original value*, *improved value*}. As explained in (Bernedixen and Ng 2014), failure to handle MCS and equality constraint effectively is one reason why some simulation-based optimization packages fall short to solve large-scale industrial system design problems. The paper also provides the details on how MCS can be effectively embedded into meta-heuristic evolutionary algorithms.

Combining several SCORE Groups provides a very powerful and generic way of setting up large and custom SCORE optimization problems with little effort compared to the tedious and error-prone way of manually setting up the same problem. This will be demonstrated in the following sections.

2.2 The Improvement Objective

The second objective (2) of the SCORE optimization problem is to minimize the total number of improvements that are actually implemented. In order to do this we need to transform the input variables (3) of the problem to binary 0/1-variables (4), where 0 corresponds to the *original value* (no improvement) and 1 corresponds to the *improved value* (one improvement). Taking the sum of these variables we are able to calculate the total number of improvements that are actually implemented in each solution, i.e. giving us the desired objective (2).

3 APPLICATION ON A SIMPLE ACADEMIC MODEL

This section will give a detailed view of how SCORE Groups can help with the formulation of the SCORE optimization problem, starting with a brief description of the model used and how the optimization problem is formulated using SCORE Groups. We then conclude with the results from the SCORE-analysis.

3.1 Model and SCORE Optimization Problem

The model used is a simple production line (Figure 2) with one variant and only 5 serial machines without any buffers. The line is never starved (i.e. infinite supply of the only variant in Source1) and never blocked (i.e. there is an infinite demand at Sink1). The settings of the machines are detailed in Table 2.

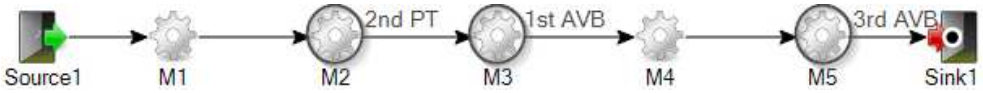

Figure 2: Simple 5-machine serial production line with mixed issues. Top 3 bottlenecks according to the SCORE-analysis are circled along with their cause (PT – process time, AVB – availability).

Table 2: Machine settings of the mixed issues line.

Parameter	M1	M2	M3	M4	M5
Process time [seconds]	32	35	30	33	31
Availability [%]	95	92	86	94	90
Mean time to repair [seconds]	600	600	600	600	600

The processing times of the machines are constant and the availabilities are modeled with exponential distribution for time between failures and Erlang distribution for repair times. The simulation model is run for 6 days with 1 day of warm-up time and with 30 replications.

Before running a SCORE-analysis on this model we need to determine some reasonable improvements. Assuming that a SCORE-analysis is to be run with the following improvements: Process times are improved by 30 % but are not allowed to be shorter than 22 seconds (e.g. due to technical restrictions), Availabilities are improved to 99 % no matter their current value and Mean time to repair are all reduced by 540 seconds. These improvements are easily defined with the three SCORE Groups shown in Figure 3.

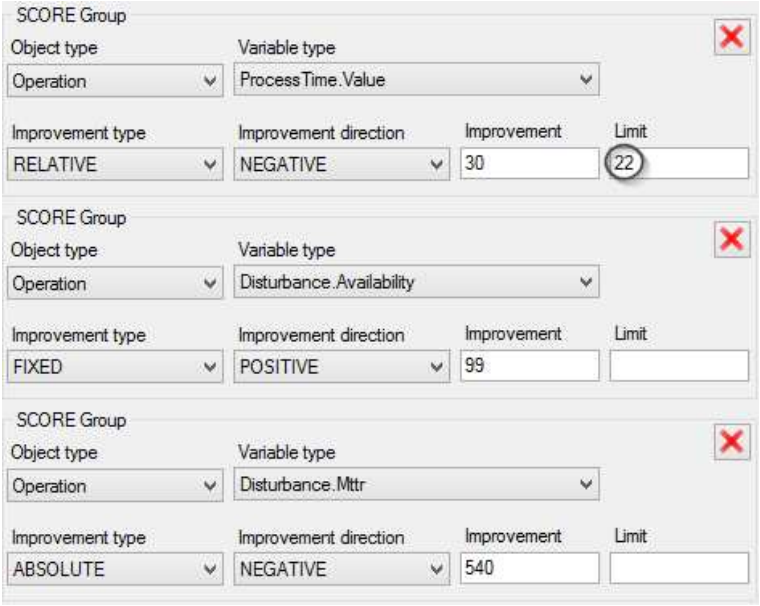

Figure 3: SCORE Groups for process time, availability and mean time to repair. Observe the limit set for process time (circled).

In this example the defined SCORE Groups results in an optimization problem with 15 input variables and 15 corresponding improvement variables. These are listed in Figure 4 along with the two objectives, maximize throughput and minimize the number of improvements. In the list of input variables there are two process time variables (circled in Figure 4) that have been capped at 22 seconds by the limit set in the process time SCORE Group, i.e. a 30 % reduction of these process times would have ended up below 22 seconds. If some improvements need individual tailoring (other than what was defined in the SCORE Group) it is possible to manually adjust these variables directly in these lists.

The total of 30 variables is not an insurmountable problem to set up manually, but still it serves as a good example to illustrate the different functionalities of the SCORE Groups. It is worth noting that even for such a simple example the SCORE Groups can avoid the error-prone, manual task significantly.

Name	Formula	Goal
minImp	imp_M1_ProcessTime_Value + imp_M...	Minimize
maxTP	Throughput	Maximize

Name	Data Type	Set
M1_ProcessTime_Value	EXTENDEDTIMETYPE	{32,22.4}
M2_ProcessTime_Value	EXTENDEDTIMETYPE	{35,24.5}
M3_ProcessTime_Value	EXTENDEDTIMETYPE	{30,22}
M4_ProcessTime_Value	EXTENDEDTIMETYPE	{33,23.1}
M5_ProcessTime_Value	EXTENDEDTIMETYPE	{31,22}
M1_Disturbance_Availability	REAL	{95,99}
M2_Disturbance_Availability	REAL	{92,99}
M3_Disturbance_Availability	REAL	{86,99}
M4_Disturbance_Availability	REAL	{94,99}
M5_Disturbance_Availability	REAL	{90,99}
M1_Disturbance_Mttr	EXTENDEDTIMETYPE	{600,60}
M2_Disturbance_Mttr	EXTENDEDTIMETYPE	{600,60}
M3_Disturbance_Mttr	EXTENDEDTIMETYPE	{600,60}
M4_Disturbance_Mttr	EXTENDEDTIMETYPE	{600,60}
M5_Disturbance_Mttr	EXTENDEDTIMETYPE	{600,60}

Name	Formula
imp_M1_ProcessTime_Value	abs(M1_ProcessTime_Value - 32) > 1E-10
imp_M2_ProcessTime_Value	abs(M2_ProcessTime_Value - 35) > 1E-10
imp_M3_ProcessTime_Value	abs(M3_ProcessTime_Value - 30) > 1E-10
imp_M4_ProcessTime_Value	abs(M4_ProcessTime_Value - 33) > 1E-10
imp_M5_ProcessTime_Value	abs(M5_ProcessTime_Value - 31) > 1E-10
imp_M1_Disturbance_Availability	abs(M1_Disturbance_Availability - 95) > 1E-10
imp_M2_Disturbance_Availability	abs(M2_Disturbance_Availability - 92) > 1E-10
imp_M3_Disturbance_Availability	abs(M3_Disturbance_Availability - 86) > 1E-10
imp_M4_Disturbance_Availability	abs(M4_Disturbance_Availability - 94) > 1E-10
imp_M5_Disturbance_Availability	abs(M5_Disturbance_Availability - 90) > 1E-10
imp_M1_Disturbance_Mttr	abs(M1_Disturbance_Mttr - 600) > 1E-10
imp_M2_Disturbance_Mttr	abs(M2_Disturbance_Mttr - 600) > 1E-10
imp_M3_Disturbance_Mttr	abs(M3_Disturbance_Mttr - 600) > 1E-10
imp_M4_Disturbance_Mttr	abs(M4_Disturbance_Mttr - 600) > 1E-10
imp_M5_Disturbance_Mttr	abs(M5_Disturbance_Mttr - 600) > 1E-10

Figure 4: Generate optimization problem; Objectives (2) & (1) at the top, Input variables (3) to the left and Improvement variables to the right.

3.2 Results

The results from the 5000 evaluations that was run with the NSGA-II algorithm (Deb et al. 2000) on the SCORE optimization problem are shown in Figure 5.

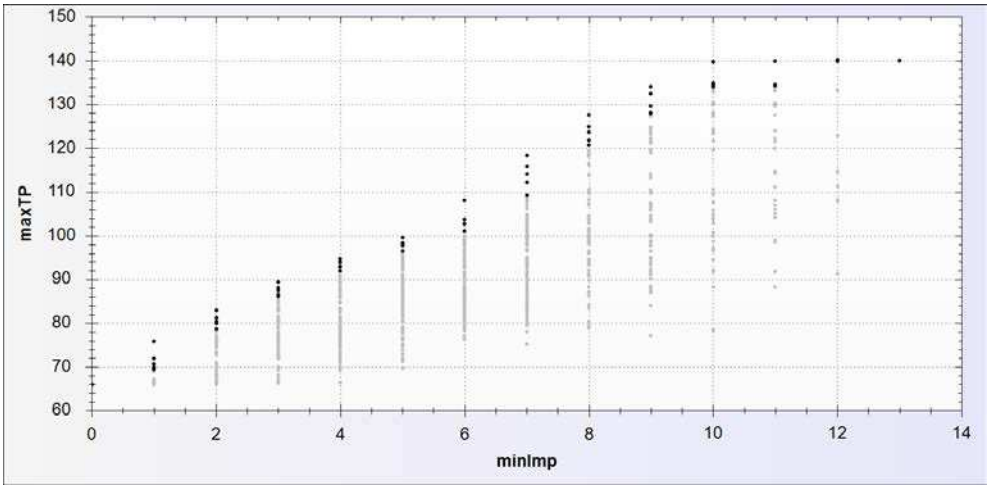

Figure 5: Optimization results with solutions of non-domination rank (Deb et al. 2000) 1 through 5 highlighted with black.

A frequency analysis is used for innovization (Deb 2003), i.e. to extract knowledge from optimizations with multiple conflicting objectives. Here, the causes of bottlenecks in the system are ranked in accordance with their severity. The best solutions, black solutions in Figure 5, are the result of applying different promising combinations of improvements to the system. Using the improvement variables of these solutions for the frequency analysis (Figure 6) we are able to rank the causes of bottlenecks in the system.

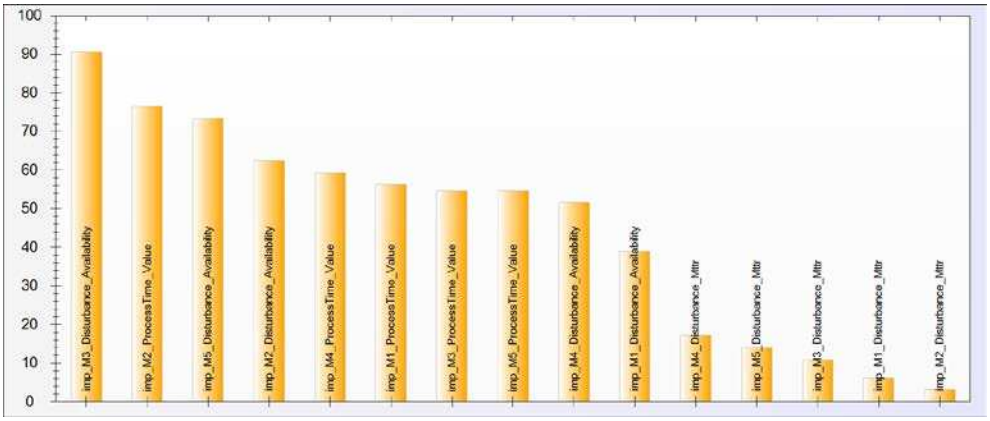

Figure 6: The 15 causes of bottlenecks ranked using the constraint frequency analysis for unique solutions of non-domination rank 1 through 5.

The frequency analysis pinpoints the availability of machine M3 as the cause of the most severe bottleneck in the system. Recall that in the initial settings of the system (Table 2), this is the machine with the worst availability (86 %). Moving on to the second cause in the bar chart, the processing time of machine M2, which is not difficult to understand since it is the machine with the longest processing time (35 seconds). While locating these two attributes as the key causes of the system constraint is not surprising, the interestingness of this analysis result is that SCORE has clearly identified that the availability loss of M3 contributes higher capacity loss than the longer processing time of M2.

For comparison, analysis results on the same production line model using the utilization method and the shifting bottleneck detection method are shown in Figure 7. Both of these two methods have identified M2 as the bottleneck machine, without providing the details on which attribute of M2 has to be improved, in contrast to the SCORE analysis which pinpoints the availability of M3 as the major cause of the constraint. The best way to evaluate the accuracy of these bottleneck methods is to simply make the improvements to the simulation model accordingly and then compare the respective throughput gains. Based on utilization and shifting bottleneck detection (Figure 7), only processing time of M2 is reduced to 24.5 seconds (all other attributes retain their original values), which gives a throughput of 69.8 parts/hr. - an improvement of 5.9%. On the other hand, based on SCORE (Figure 6), only availability of M3 is improved to 99%, which increases the throughput to 75.9 parts/hr., i.e., a gain of 15%, significantly higher than 5.9%. Albeit the model and the validation are simple, it sufficiently illustrates the adverse effect of not pinpointing the correct order of the key attributes that restrain the performance of the system.

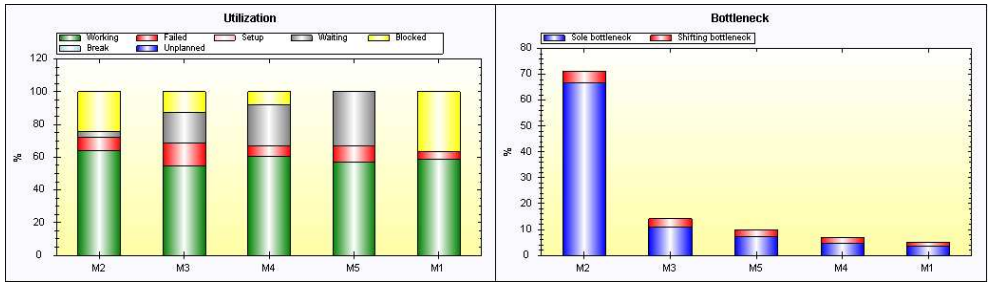

Figure 7: Bottlenecks in order of importance according to utilization (working + failed) analysis to the left and shifting bottleneck detection to the right.

It should be noted that without buffers in the system, different repair times (with the same availability) will have a limited effect on the system. As a matter of fact, it is possible to include the

increase of inter-station buffer capacities as a type of improvement action, which is also directly supported by SCORE through the addition of a SCORE Group for the parameter buffer capacity. If the simple example is extended to include inter-workstation buffers as shown in Figure 8 and a SCORE optimization is run to include Bi={1,10}, i.e. buffer capacities increase from 1 to 10. Then the SCORE analysis results shown in Figure 9 is obtained, showing that SCORE can rank buffer capacities as causes of bottlenecks.

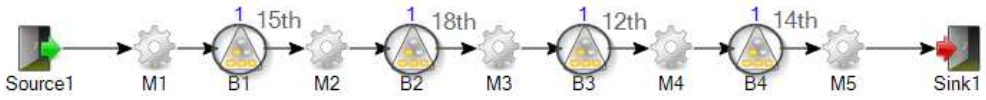

Figure 8: The simple 5-machine serial production line extended with inter-workstation buffers with their bottleneck rank highlighted.

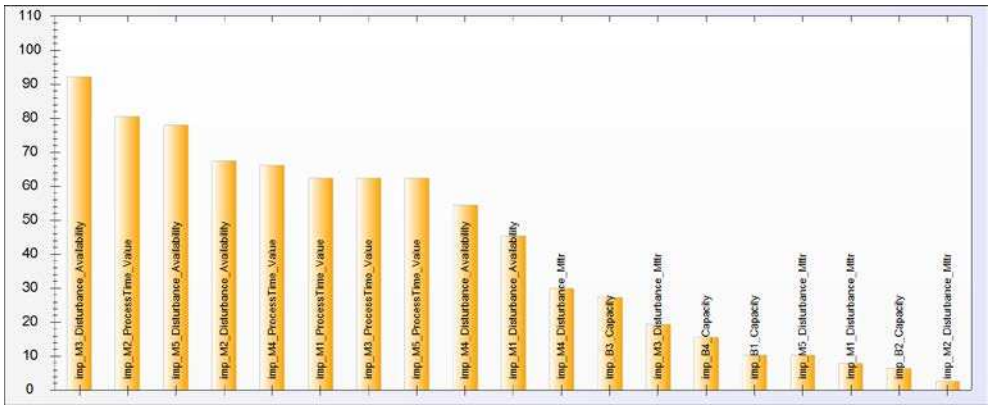

Figure 9: The 19 causes of bottlenecks (including buffer capacities) ranked using the constraint frequency analysis for unique solutions of non-domination rank 1 through 5.

4 APPLICATION WITHIN INDUSTRY

In this section, a successful application of the automated SCORE-analysis on a model of a real-world production system is presented. We will start by giving a background to this application of the SCORE-analysis and while doing so motivating the need for such an analysis. Following the background, we will present the model and the SCORE optimization problem before we conclude with some results from the analysis.

4.1 Background

Redesigning or building completely new production systems is not an easy task and sometimes the resulting system may not perform fully according to the original specifications. For instance, the system design specifies technical availabilities that are usually higher than what can be achieved when the system is up and running. Finding out why the system is not performing as planned is not an easy task. An example is a re-engineered line at a partner company (Figure 10), not performing according to the plan, despite being modeled and simulated from the project start. Relentless efforts had been made to identify and improve the bottlenecks in order to elevate the system performance, using traditional methods. However, it was not an easy task to agree on the bottlenecks and the actions required to really remedy the system performance issues. The pure utilization statistics and the more advanced shifting bottleneck detection method (Roser, Nakano, and Tanaka 2002) had already been used in several attempts to identify the bottlenecks when the automated SCORE method was applied in parallel as described in the following sections.

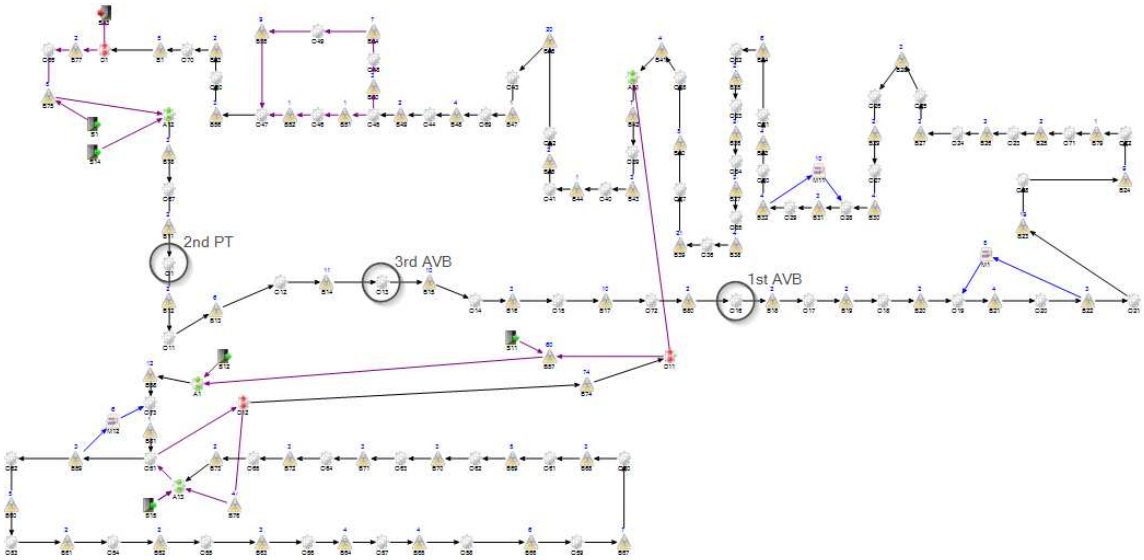

Figure 10: Model of real-world production system, illustrating its complex nature. Top 3 bottlenecks according to the SCORE-analysis are circled along with their cause (PT – process time, AVB – availability).

4.2 Model and SCORE Optimization Problem

The modeled production line (Figure 10) is a complex production system. The model contains 71 workstations, 72 buffers, 2 variants, pallets, assemblies and disassembles etc. By using default SCORE-analysis settings (20 % reduction of cycle times, availabilities improved to 98 % and repair times reduced by 50 %) a SCORE optimization problem with $N = 163$ improvement alternatives (89 process time improvements, 24 availability improvements and 50 repair time improvements) was set up. Adding these manually and then transforming them to binary 0/1-variables (4) is far too time-consuming to fit with the rest of the tasks of a production engineer and would prevent this kind of analysis from being performed. However, with this automation and the default SCORE settings, it can be done with a few straightforward steps.

4.3 Results

The results from the 20000 evaluations that were run with the NSGA-II algorithm (Deb et al. 2000) on the SCORE optimization problem are shown in Figure 11. As can be seen in the figure (circled solutions) there is a significant boost in performance with only one improvement and two improvements, indicating some rather severe bottlenecks in the system. Being such severe bottlenecks, these were actually known to the production engineer before this analysis, but knowledge about subsequent bottlenecks was missing. The frequency analysis presented in Figure 12 pinpoints the causes of these top 2 bottlenecks and also ranks subsequent bottlenecks even though they are less obvious than the first and second ones (1^{st} availability in operation O16, 2^{nd} process time in operation O1, 3^{rd} availability in operation O13, 4^{th} process time in A12…).

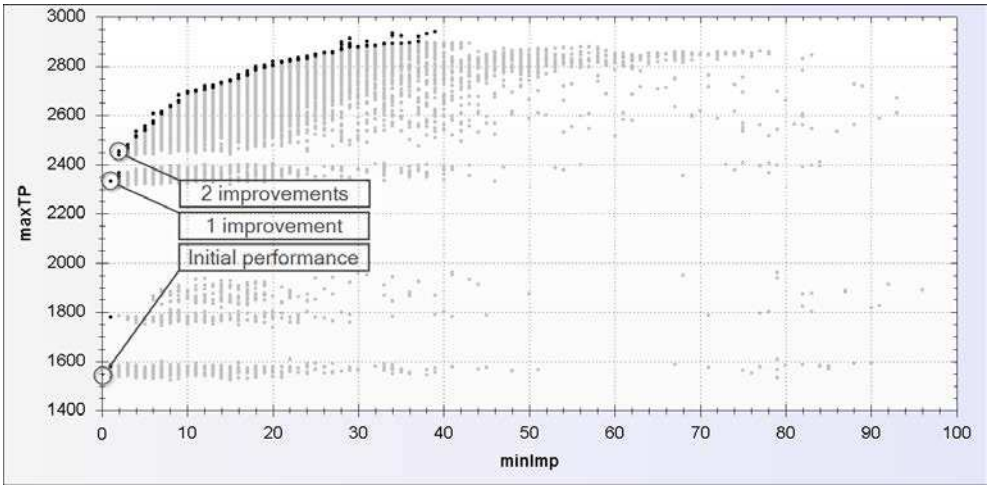

Figure 11: Optimization results with solutions of non-domination rank 1 through 5 highlighted with black. First two improvements have some significant impact on the performance of the system, as can be seen by the circled solutions, indicating the presence of severe bottlenecks.

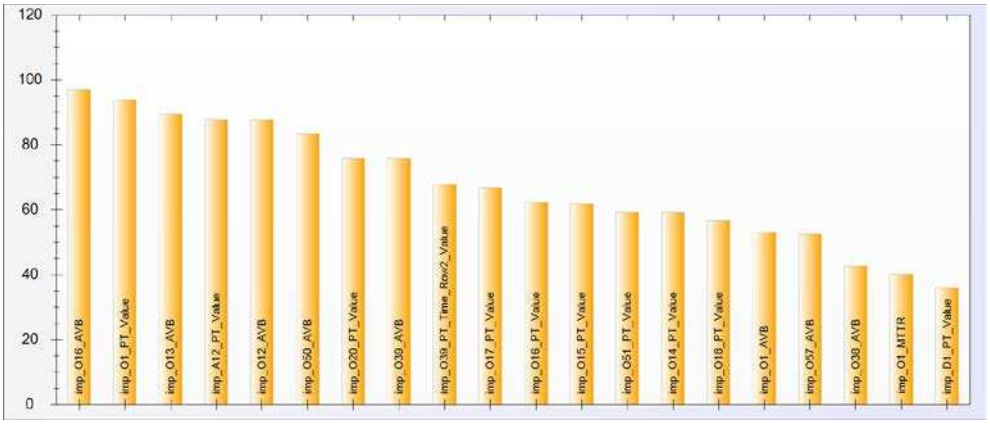

Figure 12: Top 20 causes of bottlenecks ordered using the constraint frequency analysis for unique solutions of non-domination rank 1 through 5.

Using the SCORE-analysis with the constraint frequency analysis as decision support for where to make improvements during a couple of weeks the production system now delivers as it was originally intended to.

For comparison the utilization and shifting bottleneck detection analysis are presented in Figure 13. In this case, both methods agree to the results from the SCORE-analysis for the top bottlenecks (that were known beforehand in this case, see throughput increase in Figure 11). However, they lack details about their impact on the system performance that is included in the SCORE-analysis. In addition, when comparing these three data plots generated from the three different methods, it is apparent that SCORE provides more detailed information about the order of the lower-ranked bottlenecks as well as their causes so that appropriate improvement actions to address them can be planed proactively.

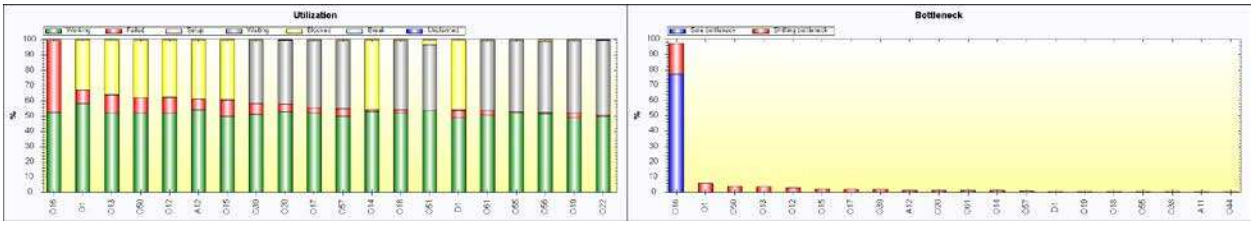

Figure 13: Top 20 bottlenecks according to utilization (working + failed) analysis to the left and shifting bottleneck detection to the right.

5 CONCLUSIONS

Based on the novel SCORE-method, that treats a bottleneck identification and improvement problem as a multi-objective optimization problem for identifying the optimal (minimal) number of changes to maximize the throughput, this paper illustrates how a generic way of defining improvements of the decision variables, in terms of processing times, availabilities, repair times, can largely automate the analysis process. The importance of such an automated assistance to the users, in most cases simulation or production engineers, should not be underestimated, since manual definition of tens, maybe hundreds of variables, is not only tedious but also error-prone. The efficiency of the automated SCORE-analysis process is illustrated through a simple academic study and through the application on a real-world complex industrial improvement project. Additionally the results obtained in both these studies have clearly shown the advantages offered by SCORE when compared to other bottleneck detection methods like machine utilizations measurement and shifting bottleneck detection. Current and future work involves adding some advanced features, e.g. parameter leveling, sensitivity analysis and guided search based on user preference, to SCORE in order to further improve the efficiency of the method.

ACKNOWLEDGMENTS

The SCORE method was first developed during the FFI-HSO project (2009-2012) funded by VINNOVA, Sweden. This work, as one of the extensions of our previous work in SCORE, is jointly funded by KKS, Volvo Car Corporation and the University of Skövde, through the Apply-IT research school. The authors gratefully acknowledge their financial supports over the years.

REFERENCES

Aneja, Y.P., and A.P. Punnen. 1999. “Multiple Bottleneck Assignment Problems.” *European Journal of Operational Research* 112:167-173.

Bernedixen, J., and A.H.C. Ng. 2014. “Practical Production Systems Optimization Using Multiple-Choice Sets and Manhattan Distance Based Constraints Handling.” In *Proceedings of the 12th Industrial Simulation Conference*, edited by A.H.C. Ng, and A. Syberfeldt, 97–103. EUROSIS.

Deb, K. 2003. “Unveiling Innovative Design Principles by Means of Multiple Conflicting Objectives.” *Engineering Optimization* 35:445–470.

Deb, K., S. Agrawal, A. Pratap, and T. Meyarivan. 2000. “A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II.” In *Parallel Problem Solving from Nature PPSN VI*, edited by M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo and H.-P. Schwefel, 849–858. Berlin: Springer Berlin Heidelberg.

Goldratt, E.M. 1997. *Critical Chain.* Great Barrington: North River Press.

Hopp, W.J., and M.L. Spearman. 2000. *Factory Physics: Foundations Of Manufacturing Management.* 2nd ed. Boston: Irwin/McGraw-Hill.

Kuo, C.-T., J.-T. Lim, and S.M. Meerkov. 1996. "Bottlenecks in serial production lines: A systematic approach." *Mathematical Problems in Engineering* 2:233-276.

Li, Lin. 2009. "Bottleneck Detection of Complex Manufacturing Systems Using a Data-Driven Method." *International Journal of Production Research* 47:6929–6940.

Li, J. and S.M. Meerkov. 2009. *Production Systems Engineering*. 1st ed. New York: Springer.

Ng, A.H.C., J. Bernedixen, and L. Pehrsson. 2014. "What Does Multi-Objective Optimization Have To Do With Bottleneck Improvement Of Production Systems?" In *Proceedings of The 6th Swedish Production Symposium*, edited by J. Stahre, B. Johansson, and M. Björkman. The Swedish Production Academy.

Pehrsson, L. 2013. "Manufacturing Management and Decision Support Using Simulation-Based Multi-Objective Optimisation." Ph.D. thesis, De Montfort University, Leicester, United Kingdom. https://dora.dmu.ac.uk/handle/2086/9697 [Accessed March 18, 2015].

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.M. Charnes, J.L. Snowdon, 1079–1086. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Sengupta, S., K. Das, and R.P. VanTil. 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, and J. W. Fowler, 1741-1745. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

AUTHOR BIOGRAPHIES

JACOB BERNEDIXEN is a system developer and PhD student at University of Skövde. He holds a M.Sc. degree in Industrial Engineering and Management from the University of Linköping, Sweden. His research interests include production system improvement using multi-objective optimization and simulation modeling and development. His email address is jacob.bernedixen@his.se.

AMOS H.C. NG is a Professor of Production and Automation Engineering at the University of Skövde, Sweden. He holds a Ph.D. degree in Computing Sciences and Engineering. His main research interest lies in applying multi-objective optimization and data mining techniques for production systems design, analysis and improvement. His email address is amos.ng@his.se.

LEIF PEHRSSON is a Senior Advisor at Volvo Car Group and an Affiliated Senior Lecturer at the University of Skövde, Sweden. He holds a Ph.D. degree in Manufacturing Systems from De Montfort University in the UK. He has been working in various engineering and managerial positions within automotive industry for 20 years. His email address is leif.pehrsson@his.se.

TOBIAS ANTONSSON is a production engineer at Volvo Car Group. He has many years of experience from industrial engineering and production engineering and has during the last five years been involved in the development of simulation and optimization procedures. His email address is tobias.antonsson@volvocars.com.