Roser, Christoph, Masaru Nakano, and Minoru Tanaka. "Time Shifting Bottlenecks in Manufacturing." In International Conference on Advanced Mechatronics. Asahikawa, Hokkaido, Japan, 2004.

Time Shifting Bottlenecks in Manufacturing

Christoph Roser
Masaru Nakano
Minoru Tanaka
Toyota Central Research and Development Laboratories
Nagakute, Aichi, 480-1192, JAPAN
E-mail: croser@robotics.tytlabs.co.jp

Abstract

The throughput of all manufacturing systems is con-strained by one or more bottlenecks. We recently developed a novel method to find the bottlenecks in a manufacturing system and to return a quantitative measure of the bottleneck constraint. This shifting bottleneck detection method has now been expanded to monitor the bottlenecks for non-steady state and highly variable discrete event systems, allowing the user to monitor these running bottlenecks over time as the underlying system changes. This paper presents the method for monitoring these shifting bottlenecks over time. The method is an extension of the highly reliable shifting bottleneck detection method. This method uses a novel holistic approach to the analysis of discrete event system, analyzing not only the entities by itself but also the interactions between the entities. The paper includes an example to demonstrate the method.

1. Introduction

This paper describes a method to detect and monitor the running bottlenecks in highly variable and non-steady-state discrete event systems subject to random variation. The method is based on the previously developed shifting bottleneck detection. There are numerous definitions as to what constitutes a bottleneck [1]. Within this paper, we define a bottleneck as a stage in a system that has the largest effect on slowing down or stopping the entire system. The presented method is an extension of the shifting bottleneck detection method [2-5].

While the shifting bottleneck approach is based on the theory of constraints [6, 7], the method is not limited to long-term average bottlenecks. Although most discrete event systems usually have one main bottleneck, in all but the simplest applications bottlenecks are not static but rather shift between different parts of the system [8, 9]. These shifts may for example be due to the sequence of random events or due to a gradual change in the system. A non-bottleneck entity may become a bottleneck, for example due to a breakdown, and similarly a bottleneck may become a non-bottleneck.

Finding the bottleneck is no trivial task, and [10] for example simply recommends that '… the best approach is often to go to the production floor and ask knowledgeable employees …'. A number of systematic methods are available to find the bottleneck for discrete event systems, although most methods are flawed. One approach measures the utilization of the different entities of the system [11], and the entity with the highest utilization is considered the bottleneck. Yet, this method is not very accurate, finds sometimes an incorrect bottleneck, does not work for shifting bottlenecks and does not return a quantitative measure of the bottleneck constraint.

Another frequently used method analyses the queue lengths of the entities in the production systems. In this method, either the queue length or the waiting time is determined, and the entity with the longest queue length or waiting time is considered to be the bottleneck. While this method is able to follow shifting bottlenecks, it is also not very accurate, finds sometimes an incorrect bottleneck, and does not return a quantitative measure of the bottleneck constraint.

Other methods are for example a very rigorous mathematical approach developed by [12-14] analyzing the interaction between the entities in order to determine the effect of the entities onto the bottleneck. [15] uses disjunctive graphs to detect the bottleneck in order to optimize the scheduling in a shifting bottleneck procedure. [16] compared the shifting bottleneck procedure to the theory of constraints.

Recently we developed a novel method to measure the likelihood of an entity being the bottleneck, resulting in a quantitative measure of the constraint. This shifting bottleneck detection method was documented extensively in earlier publications [2-5]. This paper expands the shifting bottleneck detection method for highly variable and non-steady state systems by continuously monitoring the bottleneck probability over a short periods of time. This running bottleneck shows the changes in the constraints overtime as the underlying system changes, allowing the user to adjust the system to improve the performance of the momentary constraints in order to optimize the overall system performance.

The following section reviews the shifting bottleneck detection method, followed by a detailed description of the new running bottleneck method. An example demonstrating the running bottleneck method is also presented before the paper closes with the conclusions. The following section reviews the shifting bottleneck detection method, followed by a detailed description of the new running bottleneck method. An example demonstrating the running bottleneck method is also presented before a description of the implementation and the summary.

2. Shifting Bottleneck detection

The shifting bottleneck detection method detects and monitors the shifting bottleneck of a discrete event system, and also determines the average bottleneck over a selected period of time based on the duration the system entities are active without interruption. This method expands the theory of constraints into momentary and shifting bottlenecks [8], [9].

2.1. The Momentary Bottleneck

The shifting bottleneck detection method is based on the active times of the system entities, as for example the times a entity is working, or a processor is busy, or a operator is receiving calls. While this is the same data used by the utilization bottleneck detection method, the analysis is fundamentally different. The utilization method merely sums up the active times to determine the entity with the largest active time, or largest percentage of the time being active. The shifting bottleneck detection method, however, determines how long an entity was active without interruption. The shifting bottleneck detection method compares the durations of the active periods of the different entities.

The bottleneck entity for any given time is the entity with the longest uninterrupted active period for this time.

The underlying idea of the method is that at any given time the entity with the longest uninterrupted active period is the momentary bottleneck at this time. The overlap of the active period of a bottleneck with the previous or subsequent bottleneck represents the shifting of the bottleneck from one entity to another entity.

Figure 1 visualizes the method using a simple example consisting of only two entities, E1 and E2. The figure shows the active periods of the entities over a short period of time. At the beginning, both entities E1 and M2 are active. Yet, as E1 has the longer active period, E1 is the bottleneck. However, at the end of the bottleneck period, E2 is active and has the longest active period. Therefore, the subsequent bottleneck entity is E2. During the overlap between the current bottleneck period and the subsequent bottleneck period the bottleneck shifts from E1 to E2. Processing all available data using this method shows at what time which entity is the bottleneck, when the bottleneck is shifting, and when there is no bottleneck at all. Therefore, it is possible to detect and monitor the bottleneck at all times.

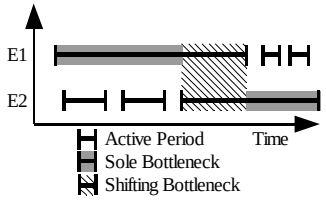

Figure 1: Shifting Bottlenecks

2.2. The Bottleneck Probability

The above method detects and monitors the momentary bottleneck at any instant of time. However, in many cases it may be of interest not to investigate an instant of time but rather for the entire simulation. This section describes how to compare different entities with respect to the bottleneck for the entire simulation. To determine the bottleneck for the entire simulation the available data is analyzed and the momentary bottlenecks are determined. Next, the percentage of time a entity is the sole bottleneck entity and the percentage of the time a entity is part of a shifting bottleneck is measured by adding up the individual bottleneck times for the entities and dividing by the duration of the simulation. This percentage is the likelihood of a machine being the bottleneck, i.e. the bottleneck probability. The bottleneck probability also represents the magnitude of an entity constraining the system, i.e. it determines not only the bottleneck but also measures the quantity of the constraint. Subsequently, the method can also be used to make further predictions about the effect of change sin the system [17, 18].

Figure 2 visualizes this method using the example with two entities as shown in Figure 1. The percentages of the entities being the sole bottleneck or the shifting bottleneck have been measured for the short data shown in Figure 1. The larger

the percentages, the larger is the effect of the respective entity onto slowing down or stopping the system. In this example, both entities are very similar, with E1 being a slightly larger bottleneck than E2.

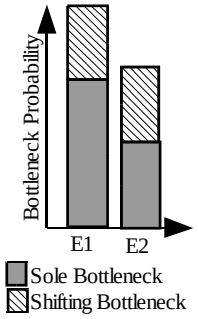

Figure 2: Average Bottleneck

The reliability and accuracy of the shifting bottleneck detection method has been proven in numerous articles [2-5], and the method was also reviewed by companies of the Toyota group and found to be very accurate.

3. Running Bottleneck Probability

The above bottleneck detection method determined either the momentary bottleneck entity at any instance in time or the bottleneck for a the entire simulation. However, practitioners are frequently interested in the bottleneck probability for shorter periods of time. Calculating the bottleneck probability for a shorter period of time is no difficult task. The running bottleneck probability takes this idea one step further and calculates a continuous bottleneck probability for a fixed period of time as shown in Figure 3 for the example of Figure 1.

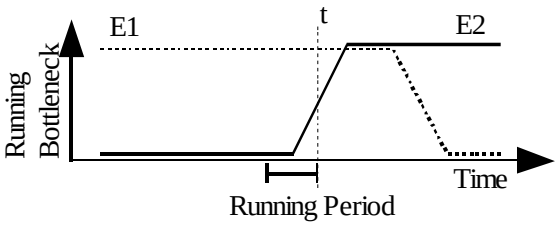

Figure 3: Shifting Bottlenecks

Figure 3 shows the bottleneck probability of the two entities E1 and E2 over time, always calculating the bottleneck probability for the previous running period. For example, during the period preceding the time *t*, entity E1 was a bottleneck for the entire period, i.e. the bottleneck probability was 100%. Entity E2 just started to be a bottleneck, and was a bottleneck for 50% of the time period preceding the time *t*, i.e. the bottleneck probability is 50%. Using this approach and selecting a suitable running period, it is easy to monitor changes in non-steady-state systems over time. The usefulness of this method will be demonstrated using a practical example.

4. Computational example

4.1. Example System

The running bottleneck probability will be demonstrated on a manufacturing system example consisting of 9 machines as shown in Figure 4. The system produces two different part types. Part type A is processed by the machines M2, M3, M4, M6, M7, M8 and M9, and part type B is processed by machines M1, M4, M5, M8 and M9. All parts are processed in the order they are received, and the buffers are using a first-in first-out (FIFO) logic. The initial system does not contain any parts, and all buffers and machines are empty. Table 1 shows the mean values of the exponentially distributed cycle times of the example system and the buffer capacities immediately before and after the machines. The buffers before M1 and M2

are large to act as an In-buffer for arriving parts, especially if they arrive in batches. The interarrival times of the parts were exponentially distributed, with the mean interarrival times being 6 minutes for part A and 20 minutes for part B.

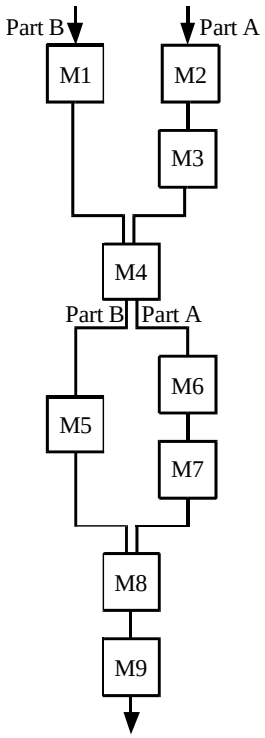

Figure 4: Example System

Table 1: Setting of Example System

Machine	Cycle Time (min)	Buffer Before	Buffer After
M1	1.5	135	1
M2	1.5	54	1
M3	4	0	1
M4	1.2	0	1
M5	6	1	1
M6	4	1	1
M7	2	0	1
M8	1	1	1
M9	1.6	0	1

4.2. Steady State System

The described system was simulated for 10,000 minutes, representing 10 days with two 8 hour shifts each. A warming up period of 1000 minutes, or one day of work, has been removed. The obtained system data has been analyzed and the bottleneck probabilities and the utilizations have been determined. Figure 5 shows the bottleneck probabilities for the different machines. Due to the high variability of the exponentially distributed machine cycle times, every machine was a bottleneck at some time. However, machines M3 and M5 had the highest bottleneck probability and were the primary bottlenecks. Improving machines M3 and M5 would be most beneficial for this system. Table 2 shows the utilizations of the different machines.

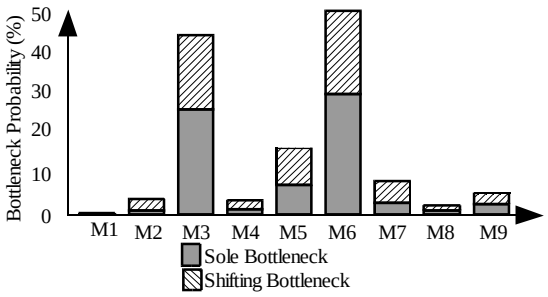

Figure 5: Steady State Bottleneck Probabilities

Table 2: Steady State Machine Utilizations

Machine	Working	Idle	Blocked
M1	7.4%	92.8%	0.5%
M2	25.8%	45.6%	28.8%
M3	65.6%	28.6%	6.0%
M4	25.9%	59.4%	14.8%
M5	30.1%	70.2%	0.0%
M6	65.7%	32.0%	2.4%
M7	31.5%	68.3%	0.3%
M8	21.6%	74.5%	3.8%
M9	34.8%	65.2%	0.0%

4.3. Non Steady State System

The above situation describes the steady state system. To create a non-steady state system, the arrivals of parts of type B are modified. In addition to the exponentially distributed arrival of a part with a mean interarrival time of 20 minutes, a batch of 130 parts arrives 2000 minutes and 6000 minutes after the start of the simulation, representing the beginning of the first shift of day 3 and day 7. The modified system has also been simulated for 10,000 minutes and a warming up period of 1000 minutes has been removed. The obtained system data has been analyzed and the bottleneck probabilities and the utilizations have been determined. Figure 6 shows the bottleneck probabilities of the non steady state system. Machines M3 and M6 are still the main bottlenecks, but now machine M5 is also a main bottleneck.

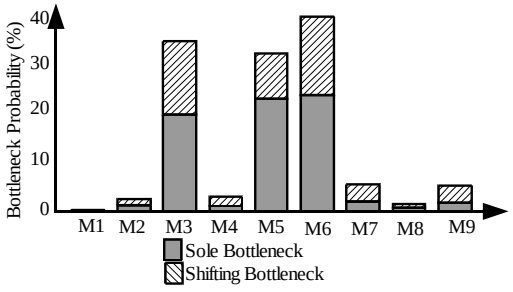

Figure 6: Non Steady State Bottleneck Probabilities

A running bottleneck analysis shows a clearer picture of the situation. Figure 7 shows the running bottleneck probabilities of the system for the prior 600 minutes during the entire simulation. Usually machines M3 and M6 are the main bottlenecks, both having a roughly equal bottleneck probability. This result is similar to the results obtained in the steady state system shown in Figure 5. However, whenever a large batch of part B arrives, machine M5 becomes the main bottleneck for the subsequent 1000~2000 minutes. At the same time, the bottleneck probabilities of machines M3 and M6 is greatly reduced, and the machines M3 and M6 are only very minor bottlenecks. After the batch of part type B is processed, the sys-

tem returns to its normal steady state until the next batch arrives. Note, that the running bottleneck probabilities of the other machines are not shown in Figure 7. The running bottleneck probabilities of the other machines all vary between 0 and 15%, and are not significant for the understanding of the system.

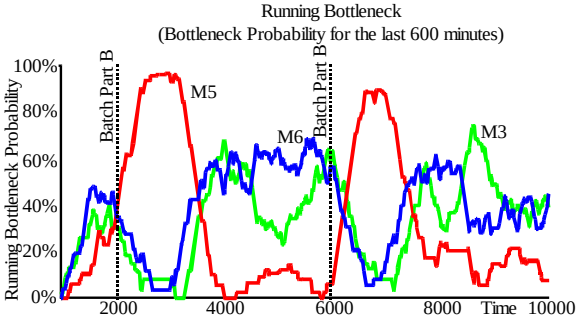

Figure 7: Running Bottleneck

Overall, the running bottleneck probability shown in Figure 7 indicates that the critical machines are machines M3 and M6, except when a large batch of part type B arrived, in which case the critical constraint is machine M5 for the subsequent 1000~2000 minutes, or 1-2 days. The subsequent actions depend if the system has short term resources available that can be allocated to improve the performance of a bottleneck machine, as for example additional workers on the bottleneck machines, or by increasing the speed of the machines at the cost of a reduced tool lifetime. For the presented example, it would improve the system throughput if machine M5 is improved by increasing the speed or allocation additional workers after a batch of parts B arrived, and to improve machines M6 and M3 at all other times. Similarly, the speed of machine M5 can be reduced to increase the tool life or additional workers can be removed if machine M5 is not a bottleneck, because improving M5 will only affect the system if M5 is a bottleneck, but will have little or no effect at other times. Similar is true for the machines M3 and M6, which should only be improved during bottleneck periods, and speeds can be reduced at other times.

5. Implementation

The running bottleneck detection method has been implemented in an automated analysis tool called TopQ Analyzer. This software automatically analyzes the history files of discrete event systems and returns a wealth of data in an easy to read MS Excel file. Besides the common performance measures as for example the utilization and the working times, the software also provides the bottleneck probabilities, the shifting bottlenecks, and the running bottlenecks both numerically and graphically. Most of the graphs in this paper are copied with small formatting changes from the output of the analysis software. A sensitivity analysis is also included.

Furthermore, the status of the buffers is also presented both as mean values and as histograms. The prediction of the effect of the buffers using a mathematical model based on a detailed system analysis is also included and can be used to optimize the buffer allocation [17].

The TopQ Analyzer is bilingual English and Japanese and automatically detects the correct language. A screenshot of the English main window is shown in Figure 8. The software is used by selected companies in the Toyota group, who also independently verified the accuracy of the shifting bottleneck detection method.

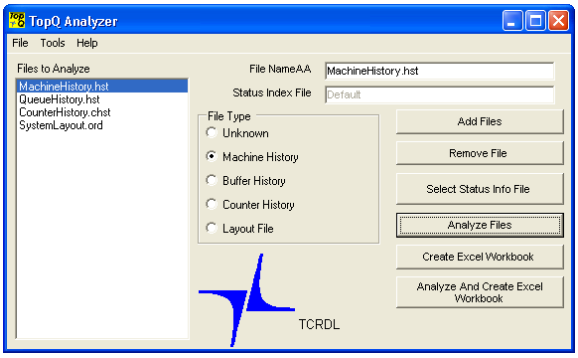

Figure 8: TopQ Analyzer Screenshot

6. Summary

The shifting bottleneck detection method has been proven to be very reliable for detecting the bottlenecks of a discrete event system and determining the magnitude of the constraints. The running bottleneck expands this reliable method for non steady state systems, allowing the study of the changes of the bottlenecks over time. Subsequently, the system can be optimized to improve the performance subject to the current bottlenecks, where effort may be spent to increase the speed of temporary bottlenecks and effort may be saved to reduce the speed of temporary non-bottlenecks. This enables the system at a better performance to cost ratio, providing a better return of the investment.

The running bottleneck has been demonstrated for a manufacturing example, but may be applied to any discrete event system, as for example computer networks, traffic systems, bank tellers, hospitals, or call centers.

Further research aims to expand the shifting bottleneck detection method for logistic systems. While logistic systems are discrete event systems, the definitions of entities working or idle as needed for the shifting bottleneck detection method can be seen as continuous. For example while a machine is either working or idle, a factory may be anywhere between completely idle or 100% working. This requires special consideration for the use of the shifting bottleneck detection method.

Nevertheless, the shifting bottleneck detection method and its variants are a well rounded method, returning practical and useful quantitative results to be used for further calculations. Therefore, the method has great value for industry, and the method is under consideration or currently adapted in industry worldwide.

7. References

[1] S. R. Lawrence and A. H. Buss, "Economic Analysis of Production Bottlenecks," *Mathematical Problems in Engineering*, vol. 1, pp. 341-369, 1995.

[2] C. Roser, M. Nakano, and M. Tanaka, "Detecting Shifting Bottlenecks," presented at International Symposium on Scheduling, Hamamatsu, Japan, 2002.

[3] C. Roser, M. Nakano, and M. Tanaka, "Tracking Shifting Bottlenecks," presented at Japan-USA Symposium on Flexible Automation, Hiroshima, Japan, 2002.

[4] C. Roser, M. Nakano, and M. Tanaka, "Shifting Bottleneck Detection," presented at Winter Simulation Conference, San Diego, CA, USA, 2002.

[5] C. Roser, M. Nakano, and M. Tanaka, "Constraint Management in Manufacturing Systems," *International Journal of the Japan Society of Mechanical Engineering, Series C, Special Issue on Advanced Scheduling*, vol. 46, pp. 73-80, 2003.

[6] E. M. Goldratt, *The Goal: A Process of Ongoing Improvement*, 2nd ed: North River Press, 1992.

[7] J. H. Blackstone, "Theory of constraints - a status report," *International Journal of Production Research*, vol. 39, pp. 1053-1080, 2001.

[8] S. R. Lawrence and A. H. Buss, "Shifting Production Bottlenecks: Causes, Cures, and Conundrums," *Journal of Production and Operations Management*, vol. 3, pp. 21-37, 1994.

[9] H. K. Moss and W. B. Yu, "Toward the Estimation of Bottleneck Shiftiness in a Manufacturing Operation," *Production and Inventory Management Journal*, vol. 40, pp. 53-58, 1999.

[10] J. F. I. Cox and M. S. Spencer, *The Constraints Management Handbook*. Boca Raton, Florida: CRC Press - St. Lucie Press, 1997.

[11] A. M. Law and D. W. Kelton, *Simulation Modeling & Analysis*, 3 ed: McGraw Hill, 2000.

[12] S.-Y. Chiang, C.-T. Kuo, and S. M. Meerkov, "Bottlenecks in Markovian Production Lines: A Systems Approach," *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 352-359, 1998.

[13] S.-Y. Chiang, C.-T. Kuo, and S. M. Meerkov, "c-Bottlenecks in Serial Production Lines: Identification and Application," *Mathematical Problems in Engineering*, 2002.

[14] C.-T. Kuo, J.-T. Lim, and S. M. Meerkov, "Bottlenecks in Serial Production Lines: A System-Theoretic Approach," *Mathematical Problems in Engineering*, vol. 2, pp. 233-276, 1996.

[15] J. Adams, E. Balas, and D. Zawack, "The Shifting Bottleneck Procedure for Job-Shop Scheduling," *Management Science*, vol. 34, pp. 391-401, 1988.

[16] R. Uzsoy and C.-S. Wang, "Performance of decomposition procedures for job shop scheduling problems with bottleneck machines," *International Journal of Production Research*, vol. 38, pp. 1271-1286, 2000.

[17] C. Roser, M. Nakano, and M. Tanaka, "Buffer Allocation Model based on a Single Simulation," presented at Winter Simulation Conference, New Orleans, Louisiana, USA, 2003.

[18] C. Roser, M. Nakano, and M. Tanaka, "Throughput Sensitivity Analysis using a single simulation," presented at Winter Simulation Conference, San Diego, CA, USA, 2002.