Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

A PRACTICAL BOTTLENECK DETECTION METHOD

Christoph Roser
Masaru Nakano
Minoru Tanaka

Software Science Laboratory
Toyota Central Research and Development Laboratories
Nagakute, Aichi, 480-1192, JAPAN

ABSTRACT

This paper describes a novel method for detecting the bottleneck in a discrete event system by examining the average duration of a machine being active for all machines. The machine with the longest average uninterrupted active period is considered the bottleneck. The method is widely applicable and also capable of analyzing complex and sophisticated systems. The results are highly accurate, distinguishing between bottleneck machines and non-bottleneck machines with a high level of confidence. This approach is very easy to use and can be implemented into existing simulation tools with little effort, requiring only an analysis of the log file which is readily available by almost all simulation tools. This method satisfies not only academic requirements with respect to accuracy but also industry requirements with respect to usability.

1 INTRODUCTION

This paper describes a method to detect the bottleneck in a discrete event system. The system consists of different entities, which can be generally grouped into two groups, which for the scope of this paper will be described as machines and parts. A machine is an entity, which performs work of any kind. A part is an entity which work is performed on. Machines may be for example processing machines, automated guided vehicles (AGV's), workers, salespeople, microprocessors, or phone operators. Parts may be for example the parts processed by a manufacturing system, data within a computer or network, customers, or phone calls.

The throughput of all systems is limited by the capacity of the different machines of the systems. Depending on the nature of the system, some machines may affect the overall throughput more than other machines. Usually, the limitations of the system can be traced to the limitations of one or two machines, commonly called constraints or bottlenecks. As one of the goals of a system is to process a large number of products in a given time, the throughput is of significant economic concern. In order to improve the throughput of the system, the throughput of the bottlenecks has to be improved (Goldratt 1992).

However, in order to improve the throughput of the bottlenecks, it is necessary to first find the bottlenecks. Finding the bottleneck is not always a trivial. For manufacturing systems, (Cox and Spencer 1997) for example simply recommend that "… the best approach is often to go to the production floor and ask knowledgeable employees …". Yet, this approach is clearly difficult to automate or to apply to a simulation. This paper describes a new method to detect the bottleneck in a discrete events system, which is fast, easy and accurate.

Currently, there are two frequently used methods to detect the bottleneck in a system by measuring either the waiting time in front of a machine or the workload represented by the percentage of the time a machine is active (Law and Kelton 1991). Both approaches have several drawbacks.

When measuring the average waiting time, the machine with the longest waiting time is considered to be the bottleneck. However, the accuracy of this approach is compromised if the system contains buffers of limited size. Furthermore, this approach analyzes only the processing machines of the manufacturing system. Other elements, as for example AGV's, supply and demand, or human workers do not have a buffer in the classical sense and require additional consideration or may not be considered at all. And finally, the waiting time of all buffers before the bottleneck tend to approach infinitely whereas all buffers after the bottleneck tend approach zero.

When measuring the workload, the machine with the largest workload is considered the bottleneck. Yet, as more than one machine may have a similar percentage of being active, the difference between the workloads of the machines may be very small. Subsequently due to the result-

ing percentages having errors due to random variation of the data, it often cannot be said with confidence which entity is the bottleneck. While this method is easy to automate, the results are not always accurate. (Luthi and Haring 1997) describe an approach to determine the likelihood of multiple bottlenecks based on the percentage of the time the machines are active using a bottleneck probability matrix. The bottleneck detection method from (Berger, Bregman and Kogan 1999) also investigates all possible combinations of bottlenecks, which rapidly becomes more complicated for larger systems. (Blake, and Breese 1995) describes an automatic bottleneck detection method for computer networks also using a measurement of the workload in combination with decision theory.

Besides methods described above, there is also the possibility to detect the bottleneck by analyzing the structure of the system. (Cox and Spencer 1997) for example describes a V-A-T logical structure analysis to find the bottleneck by investigating the structure of the system. However, this is a complex manual task, difficult to automate, and applicable only to simple systems. Adding a product mix or additional elements, as for example AGV's or human workers will greatly complicate the approach.

The presented bottleneck detection method describes a fast, easy and reliable way to detect the bottleneck in a discrete event system. The method is applicable to all active elements of a system, as for example processing machines, AGV's, human operators, data processors, or supply and demand. The following section describes the method in detail, followed by an industry example and validation.

2 BOTTLENECK DETECTION METHOD

The bottleneck detection method applies to a discrete event system consisting of one or more machines as described in the introduction. At least one of these machines is the bottleneck of the system. As the presented method is developed for discrete event systems, it is required that all of these machines are at any given time in one out of a list of possible discrete states. A processing machine, for example, may be working, waiting, in repair, changing tools, or being blocked. An AGV may be waiting, moving to a pickup location, moving to a drop off location, moving to a waiting area, recharging, or being repaired.

To apply the method only a log of the system activities is needed, i.e. which machine changed its status at what time. This is a standard output of most simulations and many system-monitoring processes and therefore readily available. No additional information about the system structure is needed. Subsequently the method is very easy to implement into most software tools.

As a first step, it is necessary to group all possible states into two groups, being either **active** states or **inactive** states. A state is inactive if the associated machine in this state is waiting for the arrival of a part or service, or for the removal of a part. A state is active whenever it is not inactive, and the machine activity is aimed towards improving the system throughput, including repair and service states. Table 1 shows a possible grouping of selected states for different machines into active and inactive.

Table 1: Active – Inactive States for Different Machines

Machine	Active	Inactive
Processing Machine	Working, in repair, changing tools, serviced	Waiting for part, waiting for service, blocked
AGV	Moving to a pickup location, moving to a drop off location, recharging, being repaired	Waiting, moving to a waiting area
Human Worker	Working, recovering	Waiting
Supply	Obtaining new part,	Blocked
Output	Removing a part form the system	Waiting
Computer	Calculating	Idle
Phone Operator	Servicing Customer	Waiting

In the conventional approach, the above grouping would be applied to the system data in order to determine the workload of a machine. However, in the presented method, not the percentage is measured, but the **duration of a machines being active**, i.e. the duration a machine is in an active state without interruption by an inactive state. Consecutive active states are considered to be one active state. For example a machine working on a part, then being repaired, and then working on another part without interruption, is considered 1 active state, with the duration of the active state being the sum of the durations of the individual active states. Figure 1 shows an example of the active (work, repair, tool change) and inactive (waiting) states of a machine during a brief period of a simulation, including the duration of the active period based on consecutive active states.

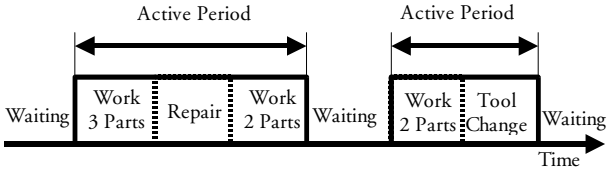

Figure 1: Active Periods of Machine during Simulation

The duration a_i of all active periods is measured for all machines i throughout the simulation data. This results in a set of durations A_i for each machine i as shown in Equation (1). The calculation of the average duration $\overline{a}_i$ for a machine i is very straightforward as shown in Equation (2).

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. The simulation example in the next section will show that this measurement usually has a very clear difference between the bottleneck machine and the non-bottleneck machines.

$$A_i = \{a_{i,1}, a_{i,2}, ...a_{i,j}, ...a_{i,n}\} \tag{1}$$

$$\bar{a}_i = \frac{\sum_{j=1}^{n} a_{i,j}}{n} \tag{2}$$

In addition, this approach has additional benefits due to the nature of the bottleneck detection method. Usually, simulation data cannot be assumed to be independent of each other, and subsequently it is difficult to calculate a confidence interval of a simulation measurement. Also, for example the percentages of time a machine is active can be measured only once per simulation. Subsequently additional techniques as for example batching have to be used to establish a valid confidence interval.

However, practical results show that the times between inactive periods are approximately independent of each other, and subsequently the average active durations are also approximately independent of each other. This allows a straightforward calculation of a standard deviation as shown in Equation (3) and a confidence interval as shown in Equation (4), estimating the accuracy of the bottleneck measurement. Therefore it is easy to determine the accuracy of the bottleneck detection.

$$\sigma_i = \sqrt{\frac{\sum_{j=1}^{n} \left(a_{i,j} - \bar{a}_i\right)^2}{n-1}} \tag{3}$$

$$CI_i = t_{\alpha/2} \cdot \frac{\sigma_i}{\sqrt{n}} \tag{4}$$

After finding the bottleneck of a system, it is then possible to improve the performance of the bottleneck in order to improve the overall performance of the system. The next section will demonstrate the method using an industry example.

3 SIMULATION EXAMPLE

The presented method was verified using a number of different simulations. The following shows one representative complex simulation example, consisting of 8 sequential machines with buffers of size 3. The simulation was performed using the GAROPS simulation software as shown in (Kubota, Sato and Nakano 1999) and (Nakano et al. 1994). A screenshot of the simulation model is shown in Figure 2. The method was implemented in an automatic software tool GAROPS ANALYZER for analyzing the log files of the GAROPS simulation software and automatically creating a report of the simulation performance data in MS Excel.

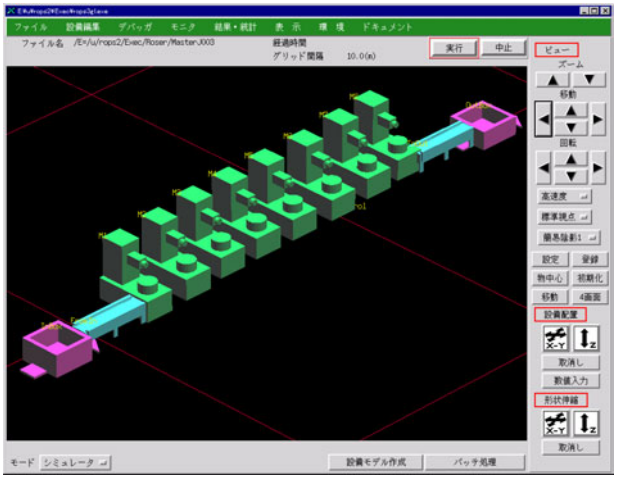

Figure 2: Simulation Model

All of the 8 machines are at any given time in one of 5 possible states. These states are 1: working, 2: waiting, 3: blocked, 4: tool change, and 5: under repair. For the bottleneck analysis, the status 2 and 3 (waiting and blocked) are considered inactive, whereas all other statuses are considered active. The simulation period was approximately 130 hours, of which a warming up period of 1 hour was removed. About 6,000 parts passed through the system during the simulation period.

For comparison purposes, the system is first analyzed using conventional bottleneck detection methods. As the buffer sizes are limited, the waiting time cannot be used to detect the bottleneck. Subsequently, the workload is determined by measuring the percentage of the time the machines are active. Table 2 shows the percentages of the times the machines were active. The width of the confidence interval with a confidence level of 95% is also given, where the confidence interval was calculated using a delta method.

Both machines M2 and M4 have a similar workload with 97% and 99% respectively, and the confidence intervals overlap. Subsequently it cannot be said with certainty which of the two machines is the bottleneck. A more detailed hypothesis test reveals that while M4 is probably the bottleneck, there is a not negligible 13.4% chance that M2 might be the bottleneck. The percentage of the machines being active including the confidence intervals is also shown graphically in Figure 3, with the possible bottlenecks shaded in gray.

Table 2: Conventional Bottleneck Detection Approach: Machine Workload

Machine	Percent Active	Confidence Interval	Bottleneck
M1	18.82%	0.18%	
M2	**97.38%**	**1.40%**	**Possible**
M3	55.58%	0.42%	
M4	**99.08%**	**2.68%**	**Possible**
M5	88.36%	0.49%	
M6	69.83%	0.39%	
M7	80.69%	0.58%	
M8	87.14%	0.60%	

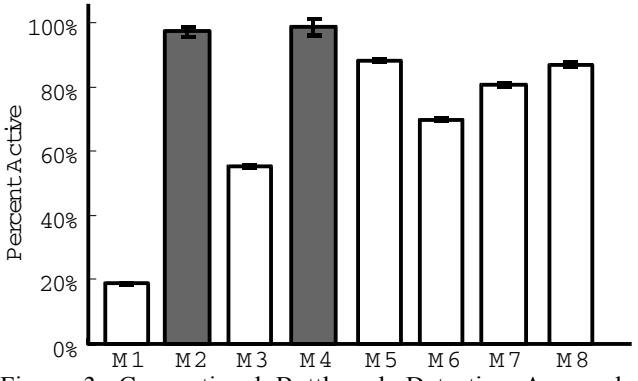

Figure 3: Conventional Bottleneck Detection Approach: Machine Workload

Subsequently, the conventional methods fail to reveal the bottleneck for the given simulation example. Next, the same simulation data is analyzed using the presented method of measuring the active periods. The measured average duration of the active period is shown in Table 3, including the confidence intervals. The duration of the active period of M4 is by far the largest duration of the system, and almost 90 times larger than machine M2, which was considered a potential bottleneck in the conventional analysis above.

Hypothesis testing reveals that it is by all practical accounts certain that M4 is the bottleneck, as there is only a 0.0044% likelihood of M2 being the bottleneck, or a joint probability of only 0.027% of any of the remaining 7 machines being the bottleneck. The average active duration and the related confidence intervals are also shown graphically in Figure 4. Please note that the active duration of M4 by far exceeds all other machines, and the durations for the non-bottleneck machines are so small that they do not show on the graph.

This finding was confirmed in two ways. A confirmation simulation of 300 hours duration also determined M4 to be the bottleneck, using the conventional approach of measuring the workloads of the machines. Furthermore, a structure analysis was performed on the simulation model, which also determined M4 to be the bottleneck. Subsequently it can be said that M4 is the true system bottleneck, and the presented method was able to detect the bottleneck fast and accurate.

Table 3: New Bottleneck Detection Approach: Average Duration of Active Period

Machine	Active Period	Confidence Interval	Bottleneck
M1	13.2	0.0	
M2	168.0	42.0	
M3	39.0	0.0	
M4	**14885.2**	**7376.9**	**Yes**
M5	62.0	0.0	
M6	49.0	0.0	
M7	59.0	1.0	
M8	65.5	1.4	

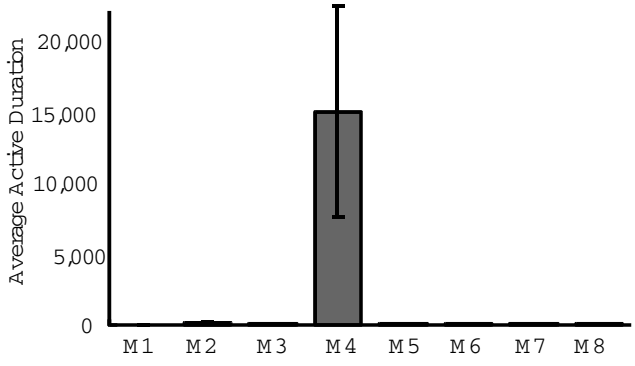

Figure 4: New Bottleneck Detection Approach: Average Duration of Active Period

In comparison, the conventional method was unable to detect the bottleneck with sufficient accuracy. The algorithm could not determine which of two machines, M2 and M4, is the bottleneck. Subsequently, it is not certain which machine should be improved to improve the overall system performance. However, the new method has a high level of confidence in determining one machine, M4, as the bottleneck. Due to the high confidence, it is literally certain that this machine M4 is the bottleneck, and that an improvement of the performance of M4 would improve the overall system performance.

4 SUMMARY

The presented bottleneck detection method has many advantages compared to conventional bottleneck detection methods. First of all, the method is very easy to use and to implement. The method utilizes only the information contained in the standard simulation log files, describing the change in the status of the different machines with time. No knowledge is needed regarding the structure of the system or the order of the processing.

The results of the bottleneck detection method also have a high level of confidence. This means, either the bottleneck is detected with a higher accuracy compared to conventional methods, or the bottleneck is detected with a shorter simulation time compared to conventional methods.

For real systems, if the bottleneck is detected faster, it is possible to improve the throughput of the bottleneck earlier, improving the overall throughput of the system.

The method is furthermore not only limited to detect bottlenecks in a classical production machine, but can also detect bottlenecks in other supporting machines as for example AGV's, repair teams, human operators, or supply and demand.

Overall, the method is well suited for use in industry and can be readily implemented to standard simulation tools or applied to historical data. Further research includes the modification of this method for non steady state systems in order to detect a shifting bottleneck at any given time within a system.

REFERENCES

Berger, Arthur, Bregman, Lev, and Kogan, Yaakov 1999. Bottleneck Analysis in Multiclass Closed Queueing Networks and Its Application. *Queueing Systems*, 31(3-4), 217-237.

Blake, Russell P., and Breese, John S. 1995. Automatic Bottleneck Detection. Technical Report MSR-TR-95-10, Microsoft Corporation (Redmond, WA), USA.

Cox, James F. III, and Spencer, Michael S. 1997. *The Constraints Management Handbook.* Boca Raton, Florida: CRC Press - St. Lucie Press.

Goldratt, Eliyahu M. 1992. *The Goal: A Process of Ongoing Improvement.* North River Press.

Kubota, Fumiko, Sato, Shuichi, and Nakano, Masaru 1999. Enterprise Modeling and Simulation Platform Integrated Manufacturing System Design and Supply Chain. In *IEEE Conference on Systems, Man, and Cybernetics*, 511-515, Tokyo, Japan.

Law, Averill M., and Kelton, David W. 1991. *Simulation Modeling & Analysis.* McGraw Hill.

Luthi, Johannes, and Haring, Guenter 1997. Bottleneck Analysis for Computer and Communication Systems with Workload Variabilities & Uncertainties. In *Proceedings of 2nd International Symposium on Mathematical Modelling*, I. Troch and F. Breitenecker, 525-534, Vienna, Austria.

Nakano, Masaru, Sugiura, Norio, Tanaka, Minoru, and Kuno, Toshitaka 1994. ROPSII: Agent Oriented Manufacturing Simulator on the basis of Robot Simulator. In *Japan-USA Symposium on Flexible Automation*, 201-208, Kobe, Japan.

AUTHOR BIOGRAPHIES

CHRISTOPH ROSER is an associate researcher at the Toyota Central Research and Development Laboratories, Nagoya, Japan. Current research topics are simulation output analysis, risk and sensitivity analysis. He received his Ph.D. in Mechanical Engineering at the University of Massachusetts, Amherst in 2000, studying flexible and robust design methods. His email address is <croser@ robotics.tytlabs.co.jp>

MASARU NAKANO is a senior researcher at Toyota central Research and Development Laboratories, Inc. (TCRDL) in Japan. He joined TCRDL in 1980 and since then has studied in the field of robotics, computer vision, and manufacturing system design. He is currently research leader and manager of the software science laboratory. He received his B.S. and M.S. in operations research from Kyoto University, and his Dr. Eng. in manufacturing systems from the Nagoya Institute of technology. His email address is <nakano@robotics.tytlabs.co.jp>

MINORU TANAKA is an associate researcher at the Toyota Central Research and Development Laboratories, Nagoya, Japan. He graduated from Matsue Technical College and joined TCRDL in 1985 and specialized in the field of robotics, manufacturing system modeling, and discrete event simulation. He is a member of The Robotics Society of Japan. His email address is <tanaka@ robotics.tytlabs.co.jp>