An Enhanced Data-Driven Algorithm for Shifting Bottleneck Detection

An Enhanced Data-Driven Algorithm for Shifting Bottleneck Detection

Chapter · August 2021

DOI: 10.1007/978-3-030-85874-2_74

4 authors:

Christoph Roser
Karlsruhe University of Applied Sciences
94 PUBLICATIONS **672** CITATIONS
SEE PROFILE

Mukund Subramaniyan
Chalmers University of Technology
15 PUBLICATIONS **173** CITATIONS
SEE PROFILE

Anders Skoogh
Chalmers University of Technology
75 PUBLICATIONS **1,051** CITATIONS
SEE PROFILE

Björn Johansson
Chalmers University of Technology
138 PUBLICATIONS **1,683** CITATIONS
SEE PROFILE

Roser, Christoph, Mukund Subramaniyan, Anders Skoogh, and Björn Johansson. “An Enhanced Data-Driven Algorithm for Shifting Bottleneck Detection.” In *Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems*. Springer, 2021. (PREPRINT)

An Enhanced Data-Driven Algorithm for Shifting Bottleneck Detection

Christoph Roser[1], Mukund Subramaniyan[2], Anders Skoogh[2], Björn Johansson[2]

[1]Karlsruhe University of Applied Sciences, Moltkestrasse 30, Karlsruhe, Germany
[2]Department of Industrial and Materials Science, Chalmers University of Technology, Gothenburg 41296, Sweden

Abstract: Bottleneck detection is vital for improving production capacity or reducing production time. Many different methods exist, although only a few of them can detect shifting bottlenecks. The active period method is based on the longest uninterrupted active time of a process, but the analytical algorithm is difficult to program requiring different self-iterating loops. Hence a simpler matrix-based algorithm was developed. This paper presents an improvement over the original algorithm with respect to accuracy.

Keywords: shifting bottleneck detection, active period method, load balancing, throughput bottlenecks, production system

1. Introduction

Finding a bottleneck is a critical task in modern industry for improving production output within a given time or reducing the time needed to produce a given quantity [1]. There are numerous bottleneck detection methods. These could be based on cycle times [2], utilization [3], waiting times [2], average waiting time [4], length of the queue [5], or combinations thereof [6]. The arrow method looks at blocking and starving [7], as does the turning point method [8]. The bottleneck walk does direct observations of waiting times and inventory levels on the shop floor [9]. Other methods are data-driven [10] or based on process mining [11]. See [12] for a more detailed discussion and comparison of these methods. However, many of these methods do not work well with shifting bottlenecks, even though in industry shifting bottlenecks are very common. In this paper, the bottleneck is defined as the machine to which the overall system throughput has the largest sensitivity [8]. This paper analyzes the active period method. While the method has high accuracy [12], it is a challenge to program the analysis code. [13] improved this method using a matrix approach to analyze the bottleneck. This paper further improves this algorithm to enhance accuracy and ease of use.

2. Active Period Method

The active period method was developed by [14], [15]. In this method, a process is considered active whenever the process is not waiting for parts or material. At any given time, the process with the longest active period is the momentary bottleneck. The overlap between the longest active periods are times of shifting bottlenecks. Periods with no overlaps are sole bottlenecks. The total bottleneck probability is the likelihood of a process being a sole or a shifting bottleneck. This method gives very accurate results even for shifting bottlenecks when compared with other methods [12].

3. An Enhanced Data-Driven Algorithm

The enhanced data-driven algorithm starts similarly as [13] with a matrix, where the number of rows corresponds to the number of processes i = 0 … m-1 and the number of columns corresponds to the number of time instants j = 0 … n-1. This generates a matrix of size m x n. The entries in the matrix take the value 1 if the process is active at the given time instant, and the value 0 if the process is inactive at the given time instant.

Please note that this matrix will usually have many more columns than rows. If the data processing is done in typical desktop software like Microsoft Excel, the software may provide more rows than columns. In this case, it may be easier to work with a transposed matrix and adjust all subsequent algorithms accordingly.

3.1. Preparation of the Raw Data

The first step is the collection and preparation of the raw data. While this is not detailed in [13], it is usually required for practical applications This is visualized in **Fig. 1**. The raw data a) is usually a table with one entry for every change in the system. Please note that due to the data collection algorithms often there are redundant entries

for the same process and time slot, resulting in multiple events with a duration of zero. In this case, only the last one is valid, which will be corrected in step d). This raw data a) is converted into a matrix b). The missing entries are filled with the data from the previous entry as shown in c). Finally, if there are multiple entries for the same time, all but the last one according to the sequence in a) is deleted, otherwise, the subsequent algorithm will provide incorrect results.

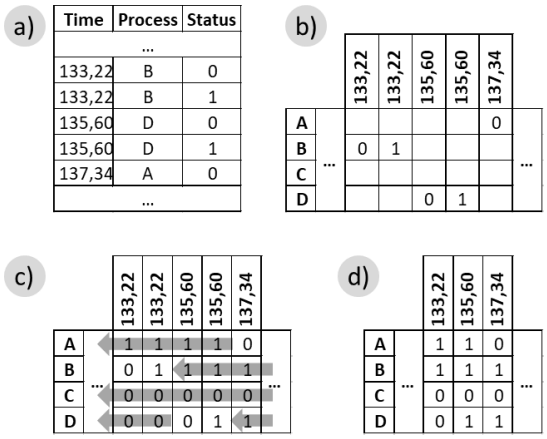

Fig. 1. Preparing of the raw data matrix

3.2. Consideration of Equidistant Interval Matrix

The original paper by [13] uses an interval of 1 second. However, different equidistant intervals may be used. If this interval is too small, the computation requirements would go up, as would the file size of the database. If the interval is set to large, the accuracy would go down, eventually giving incorrect results. In any case, any interval larger than zero has there is the risk of missing out on smaller gaps between active periods.

However, using the times of the entries of the raw data in the data matrix also brings its own risk of a flawed analysis if the algorithm merely counts the number of entries in the data matrix. These are not equidistant but depend on the random behavior of the system or the (pseudo-)random behavior of the simulation. **Fig.2** provides an illustrative example of the problem with the data columns arranged on a real-time axis. Initially, process A is the bottleneck. When process A becomes idle, a simple count of the number of active states in the data matrix shows process B with 5 successive active states in sequence, and process C only with 4. Hence, process B would be the next bottleneck based on the number of active states. However, since the columns are not equally distributed along the time axis, process C has a much longer active period and should be the next bottleneck.

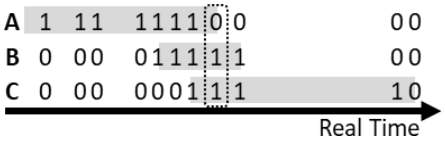

Fig.2. Illustrative example of using real-time data.

An explorative analysis showed that the error when using timestamped non-equidistant data is non-zero but usually small. Hence, it is possible to use time-stamped data for the matrix without significant loss of accuracy.

Further explorative analysis for equidistant data entries showed that the error, too, is small if the time interval is chosen small enough. The time interval depends on the cycle times of the processes and their fluctuations as well as on the buffer sizes between the processes. As a rule of thumb based on these explorative analyses, the time interval should be smaller than the cycle time for reasonable accuracy results.

Within this paper, however, we follow academic rigor and analyze the data under consideration of the actual timestamp of the data. This differs from the original method by [13]. **Fig.3** shows an example data matrix where the entries maintain their timestamps with non-equidistant intervals.

	122,17	122,92	122,94	123,73	123,96	124,92	125,70	126,64	128,72	129,02	129,82	131,18	132,05	132,36	133,22	135,60	137,34	138,79	139,16	139,99	140,26	141,21	142,67	142,77	144,44	145,98	146,75	148,65	149,39	150,22	150,66
A	0	0	1	0	1	1	1	0	1	0	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
B	1	1	1	0	1	1	0	0	1	0	0	1	1	1	1	1	1	0	1	1	0	1	1	0	1	1	1	0	1	0	0
C	0	1	1	1	0	1	0	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
D	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	0	1	1	0	1	1	0	1	1	0	1	0

Fig.3. Data Matrix with non-equidistant entries including the time stamp

3.3. State Accumulation Transformation

The state accumulation transformation is also modified from [13] to include the actual duration between the entries. In the first step, the actual duration between one entry and the next entry is entered in all matrix locations where the status from the matrix in Fig.3 was not zero (i.e. the process was active) as shown in Fig.4.

	122,17	122,92	122,94	123,73	123,96	124,92	125,70	126,64	128,72	129,02	129,82	131,18	132,05	132,36	133,22	135,60	137,34	138,79	139,16	139,99	140,26	141,21	142,67	142,77	144,44	145,98	146,75	148,65	149,39	150,22	150,66
A	0	0	0,8	0	1	0,8	0,9	0	0,3	0	1,4	0,9	0	0,9	2,4	0	1,5	0,4	0,8	0,3	0,9	1,5	0,1	1,7	1,5	0,8	1,9	0,7	0,8	0,4	2,1
B	0,8	0	0,8	0	1	0,8	0	0	0,3	0	0	0,9	0,3	0,9	2,4	1,7	1,5	0	0,8	0,3	0	1,5	0,1	0	1,5	0,8	1,9	0	0,8	0	0
C	0	0	0,8	0,2	0	0,8	0	2,1	0,3	0	1,4	0,9	0	0,9	2,4	1,7	1,5	0,4	0,8	0,3	0,9	1,5	0,1	1,7	1,5	0,8	1,9	0	0	0	0
D	0,8	0	0,8	0,2	1	0,8	0,9	2,1	0,3	0,8	1,4	0,9	0,3	0,9	2,4	0	0	0,4	0,8	0	0,9	1,5	0	1,7	1,5	0	1,9	0,7	0	0,4	0

Fig.4. Entering the duration to the next column for all non-zero entries

The next step is similar to [13], where the cumulative durations are entered for each successive entry in the matrix for the active periods. The last entry for each active period is the duration of the active period as shown in Fig.5. For this matrix, the timestamp is no longer necessary, but shown for your convenience.

	122,17	122,92	122,94	123,73	123,96	124,92	125,70	126,64	128,72	129,02	129,82	131,18	132,05	132,36	133,22	135,60	137,34	138,79	139,16	139,99	140,26	141,21	142,67	142,77	144,44	145,98	146,75	148,65	149,39	150,22	150,66
A	0	0	0,8	0	1	1,7	2,7	0	0,3	0	1,4	2,2	0	0,9	3,2	0	1,5	1,8	2,7	2,9	3,9	5,3	5,4	7,1	8,6	9,4	11	12	13	13	15
B	0,8	0,8	1,6	0	1	1,7	0	0	0,3	0	0	0,9	1,2	2	4,4	6,2	7,6	0	0,8	1,1	0	1,5	1,6	0	1,5	2,3	4,2	0	0,8	0	0
C	0	0	0,8	1	0	0,8	0	2,1	2,4	0	1,4	2,2	0	0,9	3,2	5	6,4	6,8	7,6	7,9	8,9	10	10	12	14	14	16	0	0	0	0
D	0,8	0,8	1,6	1,8	2,8	3,5	4,5	6,6	6,9	7,7	9	9,9	10	11	13	0	0	0,4	1,2	0	0,9	2,4	0	1,7	3,2	0	1,9	2,6	0	0,4	0

Fig.5. Cumulative duration of active periods

As a next step, it is necessary to add the maximum value of the active period to all entries of the active period as shown in Fig.6. This matrix can then be used for subsequent bottleneck analysis. The bottlenecks are already highlighted for an easier understanding of the data.

	122,17	122,92	122,94	123,73	123,96	124,92	125,70	126,64	128,72	129,02	129,82	131,18	132,05	132,36	133,22	135,60	137,34	138,79	139,16	139,99	140,26	141,21	142,67	142,77	144,44	145,98	146,75	148,65	149,39	150,22	150,66
A	0	0	0,8	0	2,7	2,7	2,7	0	0,3	0	2,2	2,2	0	3,2	3,2	0	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
B	1,6	1,6	1,6	0	1,7	1,7	0	0	0,3	0	0	7,6	7,6	7,6	7,6	7,6	7,6	0	1,1	1,1	0	1,6	1,6	0	4,2	4,2	4,2	0	0,8	0	0
C	0	1	1	1	0	0,8	0	2,4	2,4	0	2,2	2,2	0	16	16	16	16	16	16	16	16	16	16	16	16	16	16	0	0	0	0
D	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	0	0	1,2	1,2	0	2,4	2,4	0	3,2	3,2	0	2,6	2,6	0	0,4	0

Fig.6. Total duration of active periods in all cells of the corresponding active period.

3.4. Potential Bottleneck Detection

Finally, the bottleneck can be detected. For every column in the matrix, the largest value is a bottleneck. Similar to [13] an 1 is added in a new matrix for every active period of the previous matrix that is at least once the longest active period or shares this longest duration with one or more other active periods. The resulting matrix is shown in Fig.7. If two processes happen to have the same duration, then they are both considered to be shifting bottlenecks. If no process is active, then there is no bottleneck detected during this time.

	122,17	122,92	122,94	123,73	123,96	124,92	125,70	126,64	128,72	129,02	129,82	131,18	132,05	132,36	133,22	135,60	137,34	138,79	139,16	139,99	140,26	141,21	142,67	142,77	144,44	145,98	146,75	148,65	149,39	150,22	150,66
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
D	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig.7 . Resulting active period bottlenecks.

Further algorithms similar to [13] allow the distinction between sole and shifting bottlenecks as shown in Fig.8. A simple statistic can then give the percentage of each process being a sole bottleneck, or a shifting bottleneck, or jointly sole or shifting bottleneck.

	122,17	122,92	122,94	123,73	123,96	124,92	125,70	126,64	128,72	129,02	129,82	131,18	132,05	132,36	133,22	135,60	137,34	138,79	139,16	139,99	140,26	141,21	142,67	142,77	144,44	145,98	146,75	148,65	149,39	150,22	150,66
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	122,17	122,92	122,94	123,73	123,96	124,92	125,70	126,64	128,72	129,02	129,82	131,18	132,05	132,36	133,22	135,60	137,34	138,79	139,16	139,99	140,26	141,21	142,67	142,77	144,44	145,98	146,75	148,65	149,39	150,22	150,66
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig.8. Shifting Bottlenecks (top) and sole bottlenecks (bottom).

4. Summary and Discussion

This paper uses the active period bottleneck detection method by [14, 16] in combination with the data-driven shifting bottleneck detection algorithm as proposed by [13] and improves and enhances the latter. The new improved algorithm increases the accuracy of the analysis by using actual-time intervals to determine the longest active period at any given time.

The detailed step-by-step approach of the method allows both academics and practitioners to use the active period method for bottleneck detection. Detection bottlenecks is often a critical step in managing a production system, and crucial for improving the throughput. In a secondary role it can also play a part in reducing cost by improving throughput. Hence, finding the bottleneck is important for many improvement projects. Errors in bottleneck detection can lead to improving a non-bottleneck, which would waste time and resources. The detection of the momentary bottleneck also allows a faster reaction and allows the uses of countermeasures that can be changed on short notice, as for example the work assignment of the operators or the production plan to reduce the impact of short term bottlenecks.

5. Bibliography

1. Goldratt EM, Cox J (1992) The Goal: A Process of Ongoing Improvement, 2nd revised ed. North River Press
2. Law AM, Kelton DW (1991) Simulation Modeling & Analysis, 2nd ed. McGraw Hill
3. Tang H (2019) A new method of bottleneck analysis for manufacturing systems. Manufacturing Letters 19:21–24. https://doi.org/10.1016/j.mfglet.2019.01.003
4. Pollett PK (2000) Modelling congestion in closed queueing networks. International Transactions in Operational Research 7:319–330. https://doi.org/10.1016/S0969-6016(00)00004-6
5. Lawrence SR, Buss AH (1994) Shifting Production Bottlenecks: Causes, Cures, and Conundrums. Journal of Production and Operations Management 3:21–37
6. Elmasry GF, McCann CJ (2003) Bottleneck discovery in large-scale networks based on the expected value of per-hop delay. In: 2003 IEEE Military Communications Conference, 2003. MILCOM ’03. pp 405-410 Vol.1
7. Kuo C-T, Lim J-T, Meerkov SM (1996) Bottlenecks in Serial Production Lines: A System-Theoretic Approach. Mathematical Problems in Engineering 2:233–276
8. Li L, Chang Q, Ni J (2009) Data driven bottleneck detection of manufacturing systems. International Journal of Production Research 47:5019–5036. https://doi.org/10.1080/00207540701881860
9. Roser C, Lorentzen K, Deuse J (2014) Reliable Shop Floor Bottleneck Detection for Flow Lines through Process and Inventory Observations. In: Proceedings of the Robust Manufacturing Conference. Bremen, Germany
10. Yu C, Matta A (2016) A statistical framework of data-driven bottleneck identification in manufacturing systems. International Journal of Production Research 54:6317–6332. https://doi.org/10.1080/00207543.2015.1126681

11. Lorenz R, Senoner J, Sihn W, Netland T (2021) Using process mining to improve productivity in make-to-stock manufacturing. International Journal of Production Research 0:1–12. https://doi.org/10.1080/00207543.2021.1906460
12. Roser C, Nakano M (2015) A Quantitative Comparison of Bottleneck Detection Methods in Manufacturing Systems with Particular Consideration for Shifting Bottlenecks. In: Proceedings of the International Conference on the Advances in Production Management System. Tokyo, Japan
13. Subramaniyan M, Skoogh A, Gopalakrishnan M, et al (2016) An algorithm for data-driven shifting bottleneck detection. Cogent Engineering 3:1239516. https://doi.org/10.1080/23311916.2016.1239516
14. Roser C, Nakano M, Tanaka M (2002) Shifting Bottleneck Detection. In: Yucesan E, Chen C-H, Snowdon JL, Charnes JM (eds) Winter Simulation Conference. San Diego, CA, USA, pp 1079–1086
15. Roser C, Nakano M, Tanaka M (2002) Detecting Shifting Bottlenecks. In: International Symposium on Scheduling. Hamamatsu, Japan, pp 59–62
16. Roser C, Nakano M, Tanaka M (2004) Monitoring Bottlenecks in Dynamic Discrete Event Systems. In: European Simulation Multiconference. Magdeburg, Germany