

# Integration of Real-Time System Prototyping with Qualitative and Quantitative Reasoning-Based Parameter Tuning Methods

Kiyoshi Itoh

Laboratory of Information and Systems Engineering, Faculty of Science and Technology,  
Sophia University, Kioi-cho 7-1, Chiyoda-ku, Tokyo 102, Japan  
itohkiyo@hoffman.cc.sophia.ac.jp, tel.+81-3-3238-4143, fax.+81-3-3238-3885

<KEYWORDS>integration paradigm, prototyping, parameter tuning, qualitative reasoning, real-time systems, synchronized queueing network

<ABSTRACT>There are a number of parameter tuning plans for improving a real-time system prototype performance. This paper describes the integration of real-time system prototyping with qualitative and quantitative methods by Bottleneck Diagnosis / Improvement Expert Systems for Synchronized queueing network (BDES-S and BIES-S). By the qualitative reasoning, BDES-S diagnoses or identifies bottlenecks and their sources, and generates qualitative improvement plan. BIES-S quantitatively estimates the effects of the improvement for bottleneck. BDES-S and BIES-S assume a real-time transaction oriented concurrent software system (TCSS) as a synchronized queueing network (SQN).

## 1. Effective Performance Improvement for Real-Time System Prototype

Software prototyping is a software development method by which a prototype is constructed in a rapid and iterated fashion on the basis of careful verifications and evaluations of the prototype in order to clarify the requirements for the target software system to be developed or assure its feasibility. Prototyping cycle consists of prototype construction, execution and evaluation steps in order to satisfy the purpose of the software prototyping. The purpose of software prototyping in requirements analysis is to clarify, correct or modify ambiguous requirements of users or customers by reviewing the behavior of the prototype by repeating the cycle. The purpose in software design is to assure the feasibility of the target system by determining the implementation mechanism, algorithms or data structures to be adopted in the target system.

Real-time system prototyping methodology has been recognized to be very important as mentioned in [Luq88]. We developed a software prototyping environment called TransObj (TRANSACTION and OBJECT oriented prototyping environment) for designing real-time Transaction oriented Concurrent Software Systems (TCSS) ([ITO84, 92abc]). TCSS is a system in which one or more concurrent functional objects (abbreviated to FObjs) process a lot of transactions (abbreviated to Xacts) simultaneously in real-time. Examples of TCSSs include online reservation systems, database inquiry system, etc.. The Xact is an entity which generates on the external, is processed by the TCSS, and transmits the response to the external. FObj is an entity which is a processing component of the TCSS. One or more FObjs exist and run concurrently in one TCSS. The FObjs process the Xacts cooperatively. The TCSS has a synchronized queueing network (SQN) structure which consists of connected, Xact-driven FObjs. As a model for representing the behavior of complex TCSS, we devised the novel model, i.e., a synchronized queueing network (SQN) ([ITO91],[SHI91]). SQN is different from ordinary queueing networks (QN) ([KLE75], [GLE80]) in that SQN provides synchronized server objects.

TCSS applications require strict performance requirements. For example, a TCSS requires much more precise timing than a business application system. The response time is defined as the elapsed time between the generation of the Xact on the external and the output of the required response to the external. If the response time exceeds the permissible limit, malfunctions would occur. If the actual performance is discovered to be out of the permissible range after installation, labor will inevitably be spent in maintenance activities, such as reconfigurations and modifications. It is very important to predict the target TCSS performance precisely during the design phase, in order to find the optimal TCSS configurations.

The TransObj's methodology is called "Stepwise Prototyping Method (SPM)" by which a designer can identify and design concurrent FObjs clearly. TransObj aims at providing an integrated set of prototyping tools. In addition, TransObj has the ability for qualitative reasoning based parameter tuning for object networks in [ITO89bc, 90ab, 91ab]. For TCSS to be designed, any bottlenecks should be detected rapidly. Appropriate performance improvement plans for the bottlenecks should be generated. In particular, the reduction of prototyping cost depends on the ability to produce improvement plans. The prototyping evaluation step should accomplish appropriate parameter tuning to reduce prototyping cost. In performance design, there are a large number of parameter tuning plans for the bottleneck of a TCSS prototype.

For non-experts, in order to select appropriate plans, We developed two knowledge-based expert systems, BDES (Bottleneck Diagnosis Expert System) and BIES (Bottleneck Improvement Expert System). BDES qualitatively diagnoses or identifies bottlenecks and their sources, and generates qualitative improvement plan. BIES quantitatively estimates the effects of the improvement for bottleneck and their sources on the whole queueing network [ITO89b, 90]. BDES and BIES are based on "qualitative and quantitative reasonings," respectively. The application range of BDES and BIES is beyond the range of analytical methods which are based on "queueing theory" because it is not easy for analytical methods to produce distinct improvement plans for bottlenecks. As further extension, we have developed BDES-S and BIES-S [ITO92c]. BDES-S and BIES-S assume a real-time TCSS as a synchronized queueing network (SQN). In SQN, an object is called a server. BDES-S and BIES-S call it a server object. Xacts are processed by TCSS, i.e., Xacts are flowing on a SQN.

To determine a tuning plan, it is effective to arrange experts' knowledge. The knowledge can be well formulated within a framework of qualitative expression. At the first step for a parameter tuning, a qualitative reasoning method determines more than one tuning plans, each indicating the increases or decreases of parameters valid for tuning. All that such a tuning plan indicates is the judgment of increase or decrease of parameters; it does not indicate the specific amounts of increases or decreases of parameters. At the second step for the parameter tuning, a quantitative tuning is performed based on a quantitative reasoning method for each qualitative tuning plan proposed at the first step.

Qualitative reasoning is usually used to provides a model to express typical qualitative behaviors of some economic systems, electric circuits, and etc. ([APT86], [BOB85], [DEK85]). Our approach, i.e., applying a qualitative reasoning method to the QN, is quite a new approach ([ITO89ab,90,91a], [SAW90]).

## 2. SQN and Bottleneck

The open-type QN is dealt with in which all Xacts to be processed arrive / go out from / to the external. We assume that only 1 type of Xacts exists in the QN. A Synchronized QN (SQN) contains not only ordinary FCFS (First Come First Served) server objects in ordinary QN but also 3 types of synchronized server objects. Performance Parameters are as follows: LAMBDA: average arrival rate of Xacts for a server object,  $\mu$  : average servicing rate of a server object for Xacts,  $T$  : average throughput of Xacts by a server object,  $\rho$ : average utilization rate of a server object,  $Q$ : average queue length of Xacts in front of a server object, and  $R$ : branching probability of Xacts at a branching point.

A SQN is composed of the following 4 types of server objects. These 4 types of server objects can be used to represent various aspects of the behavior of actual systems.

(1)Normal Server object (1st entry of Table 1): A Xact arrived enters into the server object By the FCFS rule and leaves the server object after a finite time period.

(2)SPLIT Server object (2nd entry of Table 1): A Xact arrived enters into the server object By the FCFS rule and produces a clone after a finite time period. The 2 Xacts leave the server object through separate output paths simultaneously. Thus, the 2 outputs always have the same quantity.

(3)MERGE Server object (3rd entry of Table 1): Xacts arrive at 2 different queues through 2 different

paths. By the FCFS rule, 2 Xacts, from the 1st and 2nd queues, enter into the server object simultaneously and are merged into 1 Xact after a finite time period. If there is no Xact in another queue, the Xact waits for another Xact to arrive at another queue. Xact B in 3-b-2) and 3-c) of Table 1 is waiting for Xact A.

(4)MATCH Server object (4th entry of Table 1): Xacts arrive at 2 different queues through 2 different paths. By the FCFS rule, 2 Xacts, from the 1st and 2nd queues, enter into the server object simultaneously and, after a finite time interval, each Xact leaves the server object through separate output paths simultaneously. Thus, the 2 outputs always have the same quantity.

In an ordinary QN, a bottleneck exists at the server object whose RHO is very close to 1. The queue at the bottleneck server object may grow into an infinite length. If such a bottleneck server object exists, the QN is in an unstable (overloaded) state.

An expert's heuristics is that RHO of 0.7 is a bottleneck landmark (BL). If  $RHO \geq BL$ , experts judges, from the failure-to-safety aspect, that the possibility of bottleneck exists. Q of 1 is another bottleneck landmark (QBL). If  $Q \geq QBL$ , there is a risk that excessive Xacts will arrive, causing a further increase in the RHO. The excessive Q might be gradually reduced later, but experts judges, from the failure-to-safety aspect, that the possibility of bottleneck exists.

Failure-to-safety aspect means, in narrow sense, that system has the mechanism by which it fails with not hazardous state but safety state. In broad sense, it means that hazardous conditions are avoided in advance if the possibility of hazardous state exists. In the SQN, the bottleneck is considered as one of hazardous states. In M/M/1 (Poisson arrival / exponential service) type of server object, an equation,  $Q = RHO^2 / (1 - RHO)$ , holds at the stable state. Our objective QN may consist of non-M/M/1 type server objects, may be in the unstable state, or may not be measured in long time period. Therefore, distinct bottleneck landmarks, both  $RHO \geq 0.7$  and  $Q \geq 1$ , are provided on the failure-to-safety aspect.

On MERGE or MATCH server object, if the 2 queues have different LAMBDAs, the quantity processed by the server object is equal to the smaller LAMBDA. The difference of the 2 LAMBDAs will be accumulated as the system is running. From the failure-to-safety aspect, it can be judged that MERGE and MATCH server objects with 2 different LAMBDAs have the possibility of bottlenecks.

### 3. QLBE and QL-BIE Adopted in BDES-S

In the qualitative reasoning notation, [RHO] and [Q] are qualitative values evaluated with BL and QBL to be the origin, respectively.

[RHO] = +: Possibility of bottleneck exists. This means  $RHO > BL$ . From failure- to-safety aspect, we assume that  $dRHO = +$ . [Q] = +: Possibility of bottleneck exists. This means  $Q > QBL$ . We assume that  $dQ = +$ .

Qualitative expressions describe the behavior of each server object. These expressions are called qualitative behavior expressions (QLBE) for each server object. The expressions used in the case of bottleneck possibility are specifically called qualitative bottleneck improvement expressions (QL-BIE).

Table 2 shows QLBE and QL-BIE for each server object. The 1st item [RHO] = - indicates QLBE and the 2nd item [RHO] = + indicates QL-BIE. The 1st and 2nd items of Table 2(a) correspond to 1-a) and 1-b), respectively, of the 1st entry of table 1. For example, the 1st item of Table 2(a) indicates that RHO and t follows an increase and decrease of LAMBDA when [RHO] = - to a normal server object, because equation " $RHO = LAMBDA / MU$ " holds as shown in 1-a) of Table 1. The 2nd item of Table 2(a) indicates that increasing RHO ( $dRHO = +$ ) is inhibited. In the same way, the 1st row in the 3rd item of Table 2(c) indicates that a decrease of LAMBDA-B decreases Q-B but has no effect on RHO, T and Q-A when [RHO] = -, [Q-A] = - and [Q-B] = + to a MERGE server object.

In order to change LAMBDA of a server object, t from a server object placed in its upstream should be changed. An increase of t of a server object may increase LAMBDA of a server object placed in its downstream. Therefore, expressions for all server objects should be resolved simultaneously.

Fig.1 shows an example of a SQN referred to as "SQN5" which represents a real-time system prototype. The performance is measured by a simulation package. The total number of expressions for SQN5 is about 350. With additional information on the linkage information between server objects, solving these expressions simultaneously based on the qualitative behavior reasoning method is not efficient because of an explosion of the number of states.

#### 4. Substructure-Based QL-BIE adopted in BDES-S

Instead of solving all the expressions, we formulate expressions representing the correlations among internal performance parameters for each substructure, to reduce the overall number of expressions.

The following heuristics can be used to identify bottlenecks: 1)The server object whose  $RHO \geq 0.7$  may be the bottleneck. 2)The server object whose  $RHO < 0.7$  but  $Q \geq 1$  may be the bottleneck. 3)The MERGE / MATCH server object in which one LAMBDA is greater than another LAMBDA (i.e., one  $q$  is greater than another  $q$ ) in the 2 input paths may be the bottleneck.

Table 3 lists 14 knowledges for bottleneck improvement using substructures and their internal parameter values. To improving bottleneck at tandem server object s2, knowledge 3 tells us that  $t$  in at the just upstream server object be reduced. Also knowledge 2 indicates that, when increasing MU, we must reduce excessive RHO downstream.

The knowledges in Table 3 and QL-BIEs reveal that, if we focus our attention to substructures of the QN, the number of performance parameters is considerably reduced from the case where we discuss all server object parameters. For example, if server object s3 that receives throughput from a loop has the possibility of bottleneck, we can improve it by reducing only the amount of arrival at the loop, instead of reducing the performance parameters within the loop (see knowledge 7 in Table 3). We can use only the following qualitative improvement expressions to improve the bottleneck:  $dRHO_3 = -$   $dT_{in} = -$

For MERGE and MATCH server objects before transformation, LAMBDA of the normal server objects after transformation may be increased to prevent queues from growing too large. To increase LAMBDA of a server object,  $t$  from a server object that is placed in the upstream of the server object should be increased. Thus, an increase of LAMBDA results in increases of RHOs of server objects which are in its upstream, and can cause bottlenecks of these server objects. To improve these bottlenecks, expressions for these conditions should be solved using the knowledge from 10 to 14 in Table 3.

In 2 input paths of MERGE / MATCH server object, if one LAMBDA is greater than another LAMBDA (i.e., one  $q$  is greater than another  $q$ ), there exist 4 types of bottleneck improvement, i.e.,

- the large LAMBDA may be decreased,                      - the small LAMBDA may be increased,
- both may be decreased in different rates, and       - both may be increased in different rates.

#### 5. Example of Qualitative Parameter Tuning by BDES-S

BDES-S can enumerate qualitative improvement plans using heuristics as shown in Table 3. For bottleneck object s19 (MERGE server object) in SQN5 as shown in Fig.1, Fig.2 shows a qualitative diagnosis example by BDES-S. Item (1) enumerates bottleneck server objects by heuristics on initial diagnosis. Item (2) shows the structural diagnosis on bottleneck server object s19. Item (3) shows the rough sketch on improvement for the bottleneck s19. Item (4) shows qualitative improvement plans for decreasing the large LAMBDA. There are 4 alternative plans. In Item (5), some combination method shows that "\*" and "+" represent "and" and "or", respectively. There are 4 plans, i.e., 1, 2, 3\*4, 3\*5 for increasing small LAMBDA. Items (6) and (7) show that both upper and lower parts of dashed line should be required for decreasing or increasing both LAMBDAs.

#### 6. Quantitative Parameter Tuning by BIES-S

The automatic production of quantitative equations is based on the heuristics about so-called flow balance meaning that, if the bottlenecks can be improved, the input quantity is equal to the output quantity at any part of the SQN. For example, LAMBDA for the server object is equal to its T, and LAMBDA for the loop is equal to its T, if there is no bottleneck server object. By the flow balance, RHO is forced to decrease to 0.7. Only if the bottleneck server object can be improved: new MU = original LAMBDA / 0.7 or new LAMBDA = original MU \* 0.7. For a server object whose LAMBDA may be increased after tuning: new MU = new LAMBDA / 0.7. For SQN with NORMAL type server objects and SPLIT type server objects, quantitative parameter tuning can be performed using the above equations. For MERGE type server objects and MATCH type server objects with unbalanced LAMBDA's or which may get unbalanced after tuning, new LAMBDA-B = new LAMBDA-B, new MU = new LAMBDA-A / 0.7 or new LAMBDA-B / 0.7

Table 4 shows the example of quantitative parameter tuning for S19.

## 7. Concluding Remarks

BDES-S and BIES-S have been implemented in Prolog on a PC.

## References

- [APT86]Apte, C. et al. Using qualitative reasoning to understand financial arithmetic, Proc.AAAI, (1986).
- [BOB85]Bobrow,D.G.,et al.:Qualitative reasoning about physical systems, MIT Press, (1985).
- [DEK85]De Kleer, J.:How circuits work, in [BOB85].
- [GEL80]Gelenbe,E. et al.: Analysis and Synthesis of Computer Systems, Academic Press,(1980).
- [ITO84]Itoh,K. et al.: Software Design Process: Chrysalis Stage under the Control of Designers, J.Inf.Process., Vol.7, No.1, pp.5-15 (Mar. 1984).
- [ITO89a]Itoh,K. et al.: Tools for Prototyping for Developing Software, Journal of IPSJ, Vol.30, No.4, pp.387-395, (Apr. 1989), in Japanese.
- [ITO89b]Itoh, K., et al.: Knowledge-based parameter tuning for queueing network type system - A new application of qualitative reasoning,IFIP CAPE'89, pp.209- 216, (Oct.1989).
- [ITO90]Itoh,K., et al.: Role of Qualitative and Quantitative Reasoning in Diagnosis and Improvement for Queueing Network Bottleneck, InfoJapan'90,Vol.2, pp.171-178, (Oct.1990).
- [ITO91a]Itoh,K. et al. : Application of Qualitative Reasoning to Parameter Tuning, Journal of IPSJ, Vol.32, No.2, pp.171-178, (Feb., 1991), in Japanese.
- [ITO91b]Itoh,K. et al. : Qualitative Reasoning Based Parameter Tuning on Bottleneck of Synchronized Queueing Network, IEEE CompSoc '91, pp.307-314, (Sep. 1991)
- [ITO92a]Itoh,K., et al.: TransObj: Software prototyping environment for real- time transaction-based software system applications, International Journal of SE and KE, Vol.2, No.1, pp.5-30, (Mar. 1992).
- [ITO92b]Itoh,K.: Systematic integration of qualitative and quantitative parameter tuning methods for improving real-time system prototypes, 2nd IEEE ICSI, pp.54-65, (June, 1992).
- [ITO92c]Itoh,K. et al.: An integrated method for parameter tuning on synchronized queueing network bottlenecks by qualitative and quantitative reasoning, IEICE Trans.Information and Systems, Vol.E75D, pp.635-647, (Sep. 1992).
- [KLE75]Kleinrock,L.: Queueing Systems, John Wiley & Sons, Inc., (1975).
- [LUQ88]Luqi et al.: Rapidly Prototyping Real-Time Systems, IEEE Software, pp.25-36, (Sep., 1988).
- [SAW90]Sawamura,J., Itoh,K. et al.: A Method for Diagnosis and Improvement on Bottleneck of Queueing Network by Qualitative and Quantitative Reasoning, Trans. JSAI, Vol.5, No.1, pp.92-105, (Jan.,1990), in Japanese.
- [SHI91]Shida,K., Itoh,K. et al. : A Method for Qualitative Parameter Tuning of Synchronized Queueing Network Bottleneck, Trans.JSAI, Vol.6, No.6, pp.891-903, (Nov., 1991).

Table 2 QLBE and QL-BIE for each type of servers.

(a) Normal server

	$d\rho$	$dt$	$d\lambda$	$d\mu$
$[\rho] = -$	$\pm$	$\pm$	$\pm$	
	$\mp$	0		$\pm$
$[\rho] = +$	$-$	0	$-$	
	$-$	$+$		$+$

(b) Split server

	$d\rho$	$dt$	$d\lambda$	$d\mu$
$[\rho] = -$	$\pm$	$\pm$	$\pm$	$\pm$
	$\mp$	0	0	$\pm$
$[\rho] = +$	$-$	0	0	$-$
	$-$	$+$	$+$	$+$

(c) Merge server

	$d\rho$	$dt$	$dq_A$	$dq_B$	$d\lambda_A$	$d\lambda_B$	$d\mu$
$[\rho] = -$	$\pm$	$\pm$	0	0	$\pm$	$\pm$	
$[q_A] = -$	0	0	$+$	0	$+$	0	
$[q_B] = -$	$-$	$-$	0	$+$	$-$	0	
	$\mp$	0	0	0			$\pm$
$[\rho] = +$	$-$	0	$-$	$-$	$-$	$-$	
$[q_A] = +$	$-$	$+$	$-$	$-$			$+$
$[q_B] = +$	$-$	$+$	$-$	$-$	$-$	$+$	
$\times$	$-$	$+$	$-$	$-$	$-$	$+$	
$[\rho] = -$	0	0	0	0	$-$	$-$	
$[q_A] = -$	$-$	$-$	0	$-$	$-$	$-$	
$[q_B] = +$	$+$	$+$	0	$-$	$+$	$+$	

(d) Match server

	$d\rho$	$dt$	$d\lambda_A$	$d\lambda_B$	$dq_A$	$dq_B$	$d\lambda_A$	$d\lambda_B$	$d\mu$
$[\rho] = -$	$\pm$	$\pm$	$\pm$	0	0		$\pm$	$\pm$	
$[q_A] = -$	0	0	0	$+$	0	$+$	0		
$[q_B] = -$	$-$	$-$	0	0	$+$	$-$	0		
	$\mp$	0	0	0	0				$\pm$
$[\rho] = +$	$-$	0	0	$-$	$-$	$-$	$-$		
$[q_A] = +$	$-$	$+$	$+$	$-$	$-$			$+$	
$[q_B] = +$	$-$	$+$	$+$	$-$	$-$	$-$	$-$		
$\times$	$-$	$+$	$+$	$-$	$-$			$-$	$+$
$[\rho] = -$	0	0	0	0	0	$-$	$-$		
$[q_A] = -$	$-$	$-$	$-$	0	$-$	$-$	$-$		
$[q_B] = +$	$+$	$+$	$+$	0	$-$	$+$	$+$		

$\times$  denotes the expression to be used in case of  $q_A < q_B$

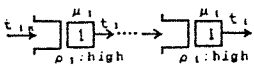
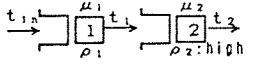
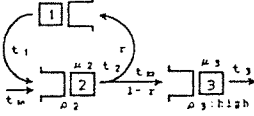
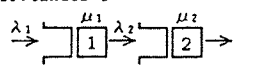
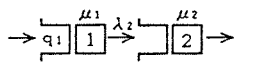
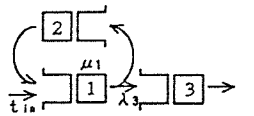
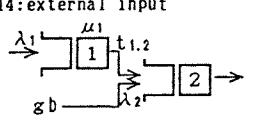
Table 1 Four types of servers and their performance parameters.

<p>1) normal server</p> <p>1-a) case of <math>\lambda &lt; \mu</math></p>	<p>1-b) case of <math>\lambda \geq \mu</math></p>
<p>2) split server</p> <p>2-a) case of <math>\lambda &lt; \mu</math></p>	<p>2-b) case of <math>\lambda \geq \mu</math></p>
<p>3) merge server</p> <p>3-a) case of <math>\lambda_A = \lambda_B &lt; \mu</math></p>	<p>3-b) case of <math>\lambda_A \geq \mu</math> and <math>\lambda_B \geq \mu</math></p> <p>3-b-1) case of <math>\lambda_A = \lambda_B \geq \mu</math></p>
<p>3-b-2) case of <math>\mu \leq \lambda_A &lt; \lambda_B</math></p>	<p>3-c) case of <math>\lambda_A &lt; \lambda_B</math> and <math>\lambda_A &lt; \mu</math></p>
<p>4) match server</p> <p>4-a) case of <math>\lambda_A = \lambda_B &lt; \mu</math></p>	<p>4-b) case of <math>\lambda_A \geq \mu</math> and <math>\lambda_B \geq \mu</math></p> <p>4-b-1) case of <math>\lambda_A = \lambda_B \geq \mu</math></p>
<p>4-b-2) case of <math>\mu \leq \lambda_A &lt; \lambda_B</math></p>	<p>4-c) case of <math>\lambda_A &lt; \lambda_B</math> and <math>\lambda_A &lt; \mu</math></p>

Table 4 Qualitative improvement plans and quantitative improvement equations & plans.

server	plan name plan type	qualitative improvement plan	quantitative improvement equation	quantitative improvement plan
s19	S19LD1 large $\lambda \rightarrow$	d r7,13 = -	$X = r7,13 (0.0 < X < 0.8)$ $0.098 \times X = 0.054$	r7,13 0.8 $\rightarrow$ 0.563
	S19LD4 large $\lambda \rightarrow$	d ga = -	$X = ga (0.0 < X < 0.166)$ $X \times 0.7 \times 0.8 \times 0.8$ $= X \times 0.7 \times 0.2 + X \times 0.3$	ga 0.166 $\rightarrow$ 0.0
	S19S11 small $\lambda \rightarrow$	d r3,8 = +	$X = r3,8 (0.2 < X < 1.0)$ $0.117 \times (1-X) \times 0.8$ $= 0.117 \times X + 0.033$	r3,8 0.2 $\rightarrow$ 0.2877
	S19S13 small $\lambda \rightarrow$	d ga = + d r2,5 = +	$X = ga (0.166 < X < 1.0)$ $Y = r2,5 (0.2 < Y < 1.0)$ (s9) $X \times 0.3 = 0.167 \times Y$ (s19) $X \times 0.7 \times 0.8 \times 0.8$ $= X \times 0.7 \times 0.2 + X \times 0.3$	ga 0.166 $\rightarrow$ 0.0 r2,5 0.2 $\rightarrow$ 0.0
	S19BD1 both $\lambda \rightarrow$	d r3,8 = - d r7,13 = -	$X = r3,8 (0.0 < X < 0.2)$ $Y = r7,13 (0.0 < Y < 0.8)$ $0.117 \times (1-X) \times Y$ $= 0.117 \times X + 0.033$	r3,8 0.2 $\rightarrow$ 0.15 r7,13 0.8 $\rightarrow$ 0.504
	S19BD2 both $\lambda \rightarrow$	d r3,8 = - d r1,3 = -	$X = r3,8 (0.0 < X < 0.2)$ $Y = r1,3 (0.0 < Y < 0.7)$ $0.166 \times Y \times (1-X) \times 0.8$ $= 0.166 \times Y \times X + 0.033$	r3,8 0.2 $\rightarrow$ 0.1 r1,3 0.7 $\rightarrow$ 0.3206
	S19BD6 both $\lambda \rightarrow$	d ga = - d r2,5 = - d r7,13 = - d $\mu 22$ = +	$X = ga (0.0 < X < 0.166)$ $Y = r2,5 (0.0 < Y < 0.2)$ $Z = r7,13 (0.0 < Z < 0.8)$ $W = \mu 22 (0.1 < W < 1.0)$ (s9) $X \times 0.3 = 0.167 \times Y$ (s19) $X \times 0.7 \times 0.8 \times Z$ $= X \times 0.7 \times 0.2 + X \times 0.3$ (s22) $0.167 \times (1-Y) \times 0.4 \times Z / W = 0.7$	ga 0.166 $\rightarrow$ 0.10 r2,5 0.2 $\rightarrow$ 0.1796 r7,13 0.8 $\rightarrow$ 0.7857 $\mu 22$ 0.1 $\rightarrow$ 0.15657
	S19B11 both $\lambda \rightarrow$	d r3,8 = + d r7,13 = +	$X = r3,8 (0.2 < X < 1.0)$ $Y = r7,13 (0.8 < Y < 1.0)$ $0.117 \times (1-X) \times Y = 0.117 \times X + 0.033$	r3,8 0.2 $\rightarrow$ 0.32 r7,13 0.8 $\rightarrow$ 0.885
	S19B18 both $\lambda \rightarrow$	d r1,3 = + d r3,8 = +	$X = r1,3 (0.7 < X < 1.0)$ $Y = r3,8 (0.2 < Y < 1.0)$ (s9) $0.166 \times (1-X) = 0.033$ (s19) $0.166 \times X \times (1-Y) \times 0.8$ $= 0.166 \times X \times Y + 0.033$	r1,3 0.7 $\rightarrow$ 0.8012 r3,8 0.2 $\rightarrow$ 0.3066

Table 3 Substructure-based knowledge and QL-BIE adopted by BDES-S.

substructure	knowledge	QL-BIE
1 The servers whose " $\rho$ "s $\geq 0.7$ or whose " $q$ "s $> 1.0$ are may be bottleneck.		
2 	On improving a bottleneck server by increasing its " $\mu$ ", decrease " $\rho$ "s of its downstream servers whose " $\rho$ "s $\geq 0.7$ or whose " $q$ "s $> 1.0$ .	$d\rho_1 = -$ $\leftarrow d\mu_1 = +$ $d\rho_2 = -$ $\leftarrow d\rho_1 = -$
3: tandem 	Decrease $t_{in}$ for decreasing $\rho_2$ .	$d\rho_2 = -$ $\leftarrow dt_{in} = -$
7: loop 	Decrease $t_{in}$ for decreasing $t_{2,3}$ (output of a loop).	$d\rho_2 = -$ $\leftarrow dt_{in} = -$
10: tandem-1 	Increase $\lambda_1$ for increasing $\lambda_2$ . (max: $\lambda_1 = \mu_1 * BL$ )	$d\lambda_2 = +$ $\leftarrow d\lambda_1 = +$
10: tandem-2 	When $q_1 \geq QL$ , increase $\mu_1$ for increasing $\lambda_2$ .	$d\lambda_2 = +$ $\leftarrow d\mu_1 = +$
13: loop 	Increase $t_{in}$ (input of a loop) for increasing $\lambda_1$ (output of loop).	$d\lambda_1 = +$ $\leftarrow dt_{in} = +$
14: external input 	a) When $g_b$ is modifiable, increase $g_b$ for increasing $\lambda_2$ . b) When $g_b$ is not modifiable, increase $t_{1,2}$ for increasing $\lambda_2$ .	$d\lambda_2 = +$ $\leftarrow dg_b = +$ $d\lambda_2 = +$ $\leftarrow dt_{1,2} = +$

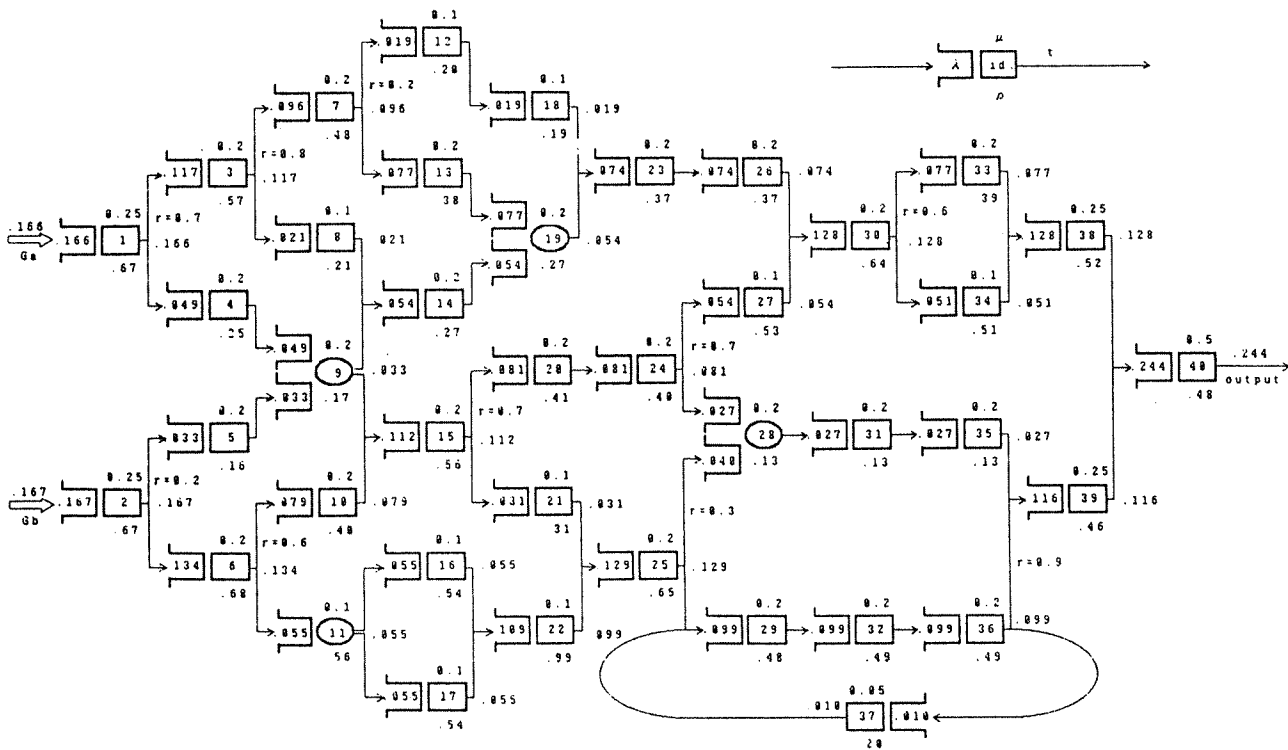


Fig. 1 Measurement for SQN5 (simulation termination time: 10,000).

```

Please input the example number: 5.
Bottleneck diagnosis for sunchronized queueing
network** sqn5 ** starts.
Please input Bottleneck landmark: 0.7.
server with maximum  $\rho$  (server,  $\rho$ , q) (1)
s22 0.99 44.14
server whose  $\rho \geq 0.7$ 
s22 0.99 44.14
Merge / Match server with unbalanced  $\lambda$ s
(server,  $\rho$ , q1, q2)
s19 0.27 111.48 0.02
s28 0.14 65.82 0.00
s9 0.17 71.21 0.07
Please input the server name to be diagnosed
:s19. (2)
s19 is merge-type-server.
s19 is judged to have 2 different  $\lambda$ s.
 $\lambda$  from server s13 is larger than  $\lambda$  from
server s14.
Therefore, q13 is large.
There is no downstream server whose  $\rho(q)$  is
large.
There is no downstream synchronized server
whose 2  $\lambda$ s are different.
For improving the bottleneck server s19 (3)
①decrease both  $\lambda$  (3a)
②increase both  $\lambda$  (3b)
③decrease larger  $\lambda$  (3c)
④increase smaller  $\lambda$  (3d)

**** Diagnosis process is omitted ****

qualitative improvement plans for s19 of
synchronized queueing network** sqn5
(4)decrease larger  $\lambda$ 
1 decrease rs7s13
2 decrease rs3s7
3 decrease rs1s3
4 decrease g[ga]
(5)increase smaller  $\lambda$ 
1 increase rs3s8
2 increase g[ga]
3 increase g[ga]
4 increase rs2s5
5 increase g[gb]
decrease  $\rho$ [s22]
※option (3)*(4+5)
(6)decrease both  $\lambda$ 
1 decrease rs3s8
2 decrease g[ga]
3 decrease g[ga]
4 decrease rs2s5
5 decrease  $\rho$ [s22]
decrease g[gb]
※option (3)*(4+5)
(7)increase both  $\lambda$ 
1 increase rs7s13
2 increase rs3s7
3 increase rs1s3
4 increase g[ga]
1 increase rs3s8
2 increase g[ga]
3 increase g[ga]
4 increase rs2s5
5 increase g[gb]
decrease  $\rho$ [s22]
※option (3)*(4+5)

```

Fig. 2 Parameter tuning on bottleneck server s19 by BDES-S.