QÜESTIIÓ, Vol 18, 2 pp. 209-228, 1994

SENSITIVITY EXAMINATION OF THE SIMULATION RESULT OF DISCRETE EVENT DYNAMIC SYSTEMS WITH PERTURBATION ANALYSIS*

TAMAS KOLTAI†, JUAN LARRAÑETA, LUIS ONIEVA and SEBASTIÁN LOZANO‡

Universitat de Sevilla

Simulation completed with perturbation analysis provides a new approach for the optimal control of queuing network type systems. The objective of this paper is to calculate the sensitivity range of finite zero-order perturbation, that is, to determine the maximum and minimum size of perturbation within which zero-order propagation rules can be applied. By the introduction of the concept of virtual queue and first and second level no-input and full-output matrices, an algorithm is provided which can solve this task efficiently in transfer lines and in relatively small general networks when short simulation run is required and the sensitivity of the individual sample path is in question. The implementation of the algorithm with the help of conventional simulation languages is also discussed and presented in an example.

Key words: Simulation, perturbation analysis, queuing networks.

*This research is part of the research project 'Expert Systems for the Scheduling and Dynamic Production Control in a Steel making Factory BOF/Secondary Metallurgy/Continuous Casting', sponsored by the European Coal and Steel Community, and carried out at the University of Seville, Department of Industrial Organization.

† Tamas Koltai. Associate Professor at The Technical University of Budapest, Visiting Professor at The University of Seville.

‡ Juan Larrañeta. Professor.
Luis Onieva. Associate Professor.
Sebastián Lozano. Associate Professor.
} University of Seville, School of Engineering, Department of Industrial Organization.

–Article rebut el desembre de 1992.
–Acceptat el desembre de 1993.

1. INTRODUCTION

The complex nature of design, planning and control of modern, highly automated production systems raises various challenging problems in the field of operations management. The difficulty of their examination comes from the fact that these are generally discrete event dynamic systems (DEDS) in the majority of the cases with a stochastic nature. Queuing network representation seems to provide appropriate modelling framework but their exact analytical examination can be performed just in very limited cases. Usually the combined application of simulation and exact optimization algorithm is required.

One of the most challenging solution of this problem seems to be the recently developed perturbation analysis (PA) which can provide gradient information from a single simulation experiment [4]. The idea is to perform a simulation experiment, and via an algorithm an estimate can be derived about the derivative of a performance measure of the system with respect to one of its parameters [6]. This gradient information can be used for iterative improvement of system performance [8],[11].

Various intriguing problems have been solved since the first presentation of the method. Propagation rules for infinitesimal and finite perturbations [5], examination of multi-class networks [1], various suggestions for avoiding or at least smoothing the effect of discontinuities are extending the applicability of the method [7]. Researchers of this field, however, have mostly concentrated on generating and/or propagating perturbations, but have avoided the examination of validity range within which the gradient information is correct. The infinitesimal approach deals with this problem by simply saying that the size of the perturbation is small enough not to hurt the deterministic similarity. The finite approach calculates accurately the effect on the performance measures or on other activities with higher order propagation rules, but it also fails to provide information about the validity [5]. The effect of a specific perturbation is calculated correctly but if the perturbation changes the calculation has to be performed again.

The objective of this paper is to calculate the sensitivity range of zero-order perturbation, that is, to determine the maximum and minimum size of perturbation within which zero-order propagation rules can be applied. The practical significance of this problem is, that within the determined sensitivity limits a linear extrapolation of the change of the examined performance measure based on the gradient information is accurate [3].

An intuitive illustration of this problem can be given by linear programming (LP). The basic idea of the optimization process in LP is that an initial basis solution is generated and an iterative improvement is performed with the help of the shadow prices. The shadow price is actually the gradient of the objective function with respect to the right hand side parameters. The validity range of the shadow price imposes limits on the improvement and also provides important information about the limiting constraints [2]. With PA we may try to construct a similar process. The simulation results can be considered as initial solutions. The gradient information can be gained by the propagation rules. The objective of this paper is to derive the validity range of the gradient.

2. PROBLEM DEFINITION

Suppose we have a transfer line type queuing network consisting of M resources and the same number of queues, illustrated in Fig. 1. Every resource (R_j) is preceded by a queue (Q_j) with $c(j)$ capacity. ($c(j)$ includes the place in R_j as well.) At the queues the first-in-first-out (FIFO) rule is applied.

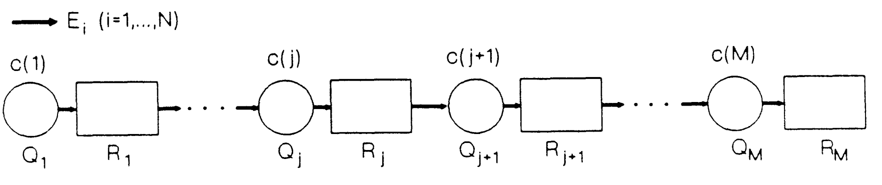

Figure 1.
Transfer line type queuing network.

N entities (E_i) enter the network and follow a flow-shop type process. The order of entities at every resource is the same. The operation time of E_i at R_j in a specific sample path is $t^{\omega}_{i,j}$ (shortly $t_{i,j}$)[1].

A simulation experiment is performed. As a result a time schedule of the activities is gained and summarized in a matrix B where $b_{i,2j-1}$ denotes the beginning time and $b_{i,2j}$ the ending time of the operation of E_i at R_j.

A single finite perturbation is introduced at E_x on $R_y(\delta_{x,y})$ and its effect is examined on the throughput time (T), that is, on the total operation time

[1] Since all the calculations introduced here, refer to a specific sample path we abandon ω when it is not misleading.

necessary to finish the manufacturing of N entities. We would like to determine the gradient of the throughput time with respect to the change of the operation time $t_{x,y}$ and the validity range of this gradient.

Our analysis is based on the **event sequence table** introduced by Ho and Cassandras [6]. The principle of this table is that every resource participate in one of three mutually exclusive events, that is,

- performing an operation on an entity (OP),
- being idle and waiting for the arrival of an entity (NI),
- being blocked by the proceeding part of the process (FO).

The event sequence table contains the order of these events at the resources. The event sequence table changes if any of its event disappears or new one appears as a consequence of any change of the system control variables. **Deterministic similarity** means that the event sequence table of the original and perturbed sample path are equal.

Based on these definitions our problem can be expressed on the following way:

We suppose that $\delta_{x,y}$ is not infinitesimal, but small enough not to hurt the deterministic similarity. We want to determine the set of $\delta_{x,y}$ which satisfies this requirement.

3. EVENT SEQUENCE AND VIRTUAL QUEUE

We will assign numeric values to the activities of the event sequence table and group NI (no-input) and FO (full-output) activities in different matrices. First, however, we have to introduce the concept of the 'virtual queue'.

Entities may wait for a resource either in a queue or, in case of blocking, in an other resource. Let's suppose that E_i is blocked in R_j (Fig. 2.). This practically means that E_i joins to the $c(v)$ entities waiting for the release of R_v, by this way transforming R_j into a part of the queue of R_v. If in the meantime entities are entering in Q_j then they will join a line which now consists of the entities waiting in Q_v, R_j and Q_j for the release of R_v. If Q_j is full and blocks R_p then the entities in R_p and in Q_p join the same line, transforming R_p and Q_p as well into part of the virtual queue of R_v.

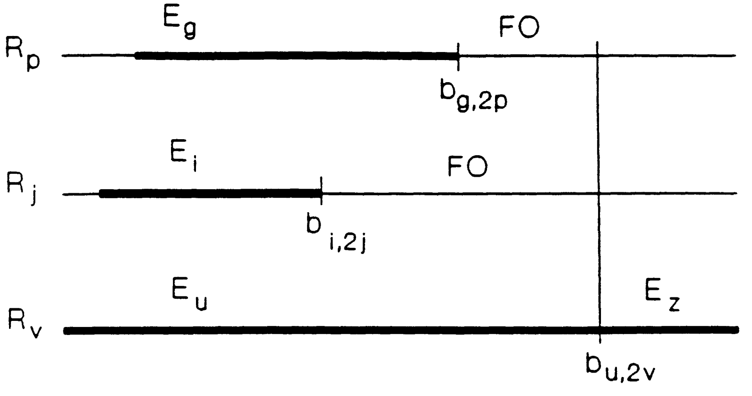

Figure 2.
Illustration of the virtual queue.

Definition 1:

The virtual queue consists of all those resources and queues in which the entities are waiting at the same time for the release of the same resource.

In further discussions, let $\boldsymbol{V(t)} \subset \{\mathbf{1}, \ldots, \boldsymbol{M}\}$ denote the index set of the resources belonging to a virtual queue of $\boldsymbol{R_v}$ $\boldsymbol{v} \in \boldsymbol{V(t)}$, as $\boldsymbol{E_u}$ causes blocking in it when $b_{u,2v-1} \leq t \leq b_{u,2v}$. If we move upstream in this virtual queue, then the utmost resource which is blocked (by $\boldsymbol{E_g}$) will be $\boldsymbol{R_p}$ $\boldsymbol{p} \in \boldsymbol{V(t)}$.

A state of a queue changes if the number of entities in the queue changes [6]. This will be true for the virtual queue as well. The only difference is that even if the capacity of all the resources is one, the state of the virtual queue may change by more than one. It may occur when a queue joins the virtual queue and contains more than one entity. For example in Fig. 2. all the entities in Q_p join the virtual queue of R_v at $b_{g,2p}$. It may also occur that more than one resource and their queues join or leave the virtual queue at the same time.

The capacity of the virtual queue is the sum of the capacity of all the queues participating in it (In Fig. 2. $c'(v) = c(v) + c(j) + c(p)$). An extreme case of the virtual queue is a transfer line where the last workstation is the bottleneck of the system and blocks all the other machines.

The pseudo code of the algorithm to find the resource and entity (R_v, E_u) which are responsible for the appearance of the virtual queue containing R_j, E_i is represented in Fig. 3.

```
fin:= 0
WHILE j + 1 ≤ M AND i − c(j + 1) > 0 AND fin:= 0 DO
        BEGIN
        IF b_{i−c(j+1),2(j+1)} > b_{i,2j} THEN
            BEGIN
            j := j + 1
            i := i − c(j)
            END
        ELSE
            BEGIN
            u := i + c(j)
            v := j − 1
            fin:= 1
            END
        END
store (v, u)
```

Figure 3. Pseudo code of the algorithm for searching E_u, R_v.

The last resource element of the virtual queue and the entity residing in it (R_p, E_g) can be found by the algorithm given in Fig. 4. It is assumed that R_v and E_u is already known and the search is started at E_i in R_j $j \in V(t)$.

```
fin:= 0
WHILE j − 1 > 0 AND i + c(j) ≤ N AND fin:= 0 DO
        BEGIN
        IF b_{i,2j} < b_{u,2v} THEN
            BEGIN
            i := i + c(j)
            j := j − 1
            END
        ELSE
            BEGIN
            g := i − c(j + 1)
            p := j + 1
            fin:= 1
            END
        END
store (p, g)
```

Figure 4. Pseudo code of the algorithm for searching E_g, R_p.

4. QUANTITATIVE ANALYSIS OF THE FO AND NI EVENTS

We will define two FO and two NI matrices based on whether the event (FO or NI) in question is **experienced at** or **caused by** E_i in R_j.

4.1. Full-output matrices

Definición 2

The first level full-output matrix (FO(1)) expresses the duration of the FO R_j has to **endure**, while it contains E_i.

A FO event exists when an entity can not leave the resource because the queue (either real or virtual) it has to enter is full. On Fig. 5(a). and 5(b). it can be seen that E_i wants to leave R_j and go to R_{j+1} but Q_{j+1} is full. Since the capacity of Q_{j+1} is $c(j+1)$, R_j will be blocked until $E_{i-c(j+1)}$ can not leave R_{j+1} and free a place in Q_{j+1}. This event may occur at $b_{i-c(j+1),2(j+1)}$ if $j+1 \notin V(t)$ or at $b_{u,2v}$ if $j+1 \in V(t)$. The first level full-output matrix ($fo_{i,j,1}$) denotes the length of the FO event and is calculated as follows.

$$fo_{i,j,1} = \begin{cases} b_{i-c(j+1),2(j+1)} - b_{i,2j} & \text{if } j+1 \not\subset V(t) \\ b_{u,2v} - b_{i,2j} & \text{otherwise} \end{cases} \tag{1}$$

If $fo_{i,j,1} > 0$ then FO really exists when E_i wants to leave R_j and the blocking of R_j lasts $fo_{i,j,1}$. If $fo_{i,j,1} \leq 0$ then there is no FO and $fo_{i,j,1}$ expresses the available time to avoid it.

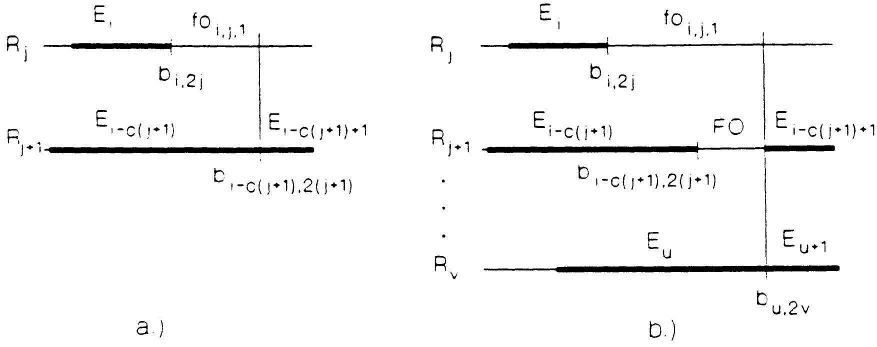

Figure 5.
Calculation of the FO(1) matrix.

Definition 3

The second level full-output matrix (FO(2)) expresses the duration of the FO R_j **causes**, while it contains E_i. The calculation of $fo_{i,j,2}$ is as follows (fig 6(a), 6(b)):

If, $(j-1) \notin V(t)$ then,

(2) $$fo_{i,j,2} = b_{i+c(j),2(j-1)} - b_{1,2j}$$

If, $(j-1) \in V(t)$ then,

(3) $$fo_{i,j,2} = \begin{cases} b_{g,2p} - b_{i,2j} & \text{if, } b_{g+1,2(p-1)} \leq b_{u,2i} \\ b_{g+1,2(p-1)} - b_{i,2j} & \text{otherwise} \end{cases}$$

where (R_p, E_g) is the last in this virtual queue.

If $fo_{i,j,2} < 0$, then FO really exists when E_g wants to leave R_p. If $fo_{i,j,2} \geq 0$, then there is no FO and $fo_{i,j,2}$ expresses the available time to avoid it.

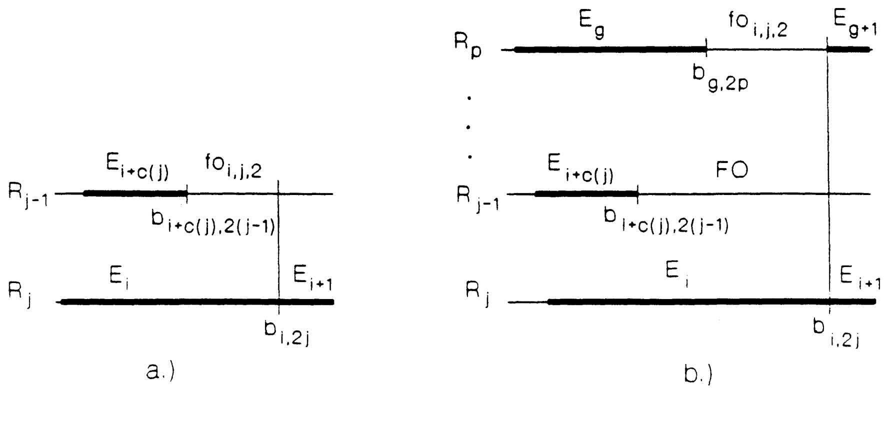

Figure 6. Calculation of the FO(2) matrix.

4.2. No-input matrices

Definition 4

The first level no-input matrix (NO(1)) expresses the duration of the NI R_j has to **endure** after completing the operation on E_i.

A no-input event exists when a resource is empty and waits for an incoming entity. It may occur just after the termination of an operation (Fig. 7(a).) or after the release of the resource from blocking (Fig. 7(b).).

If. $(j-1) \notin V(t)$ and $j \notin V(t)$ then.

$$ni_{i,j,1} = b_{i+1,2(j-1)} - b_{i,2j} \tag{4}$$

If. $(j-1) \notin V(t)$ and $j \in V(t)$ then.

$$ni_{i,j,1} = b_{i+1,2(j-1)} - b_{i,2j} - fo_{i,j,1} \tag{5}$$

If $ni_{i,j,1} > 0$, then NI really exists when E_i leaves R_j and $ni_{i,j,1}$ expresses the time R_j has to wait for E_{i+1}. If $ni_{i,j,1} \leq 0$, then there is no NI and $ni_{i,j,1}$ expresses the available time to avoid it. When both R_j and R_{j+1} belong to a virtual queue, then $ni_{i,j,1}$ has no meaning because E_{i+1} can never cause NI at R_j.

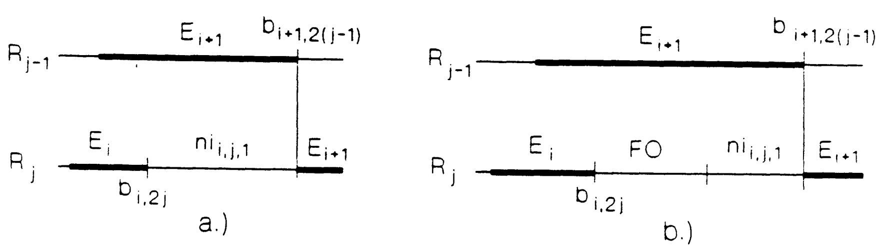

Figure 7.
Calculation of the NI(1) matrix.

Definition 5

The second level no-input matrix (NO(2)) expresses the duration of the NI **caused** by R_j while performing operation on E_i.

The calculation of $ni_{i,j,2}$ is as follows (fig 8(a). 8(b)):

$$ni_{i,j,2} = \begin{cases} b_{i-1,2(j+1)} - b_{i,2j} & \text{if } j+1 \notin V(t) \\ b_{u,2i} - b_{i,2j} & \text{otherwise} \end{cases} \tag{6}$$

If $ni_{i,j,2} < 0$, then NI really exists when E_{i-1} (E_u) leaves R_{j+1} (R_i) and $ni_{i,j,2}$ expresses the time R_{j+1} (R_i) has to wait for E_i (E_{u+1}). If $ni_{i,j,2} \geq 0$, then there is no NI and $ni_{i,j,2}$ expresses the time we have to avoid it.

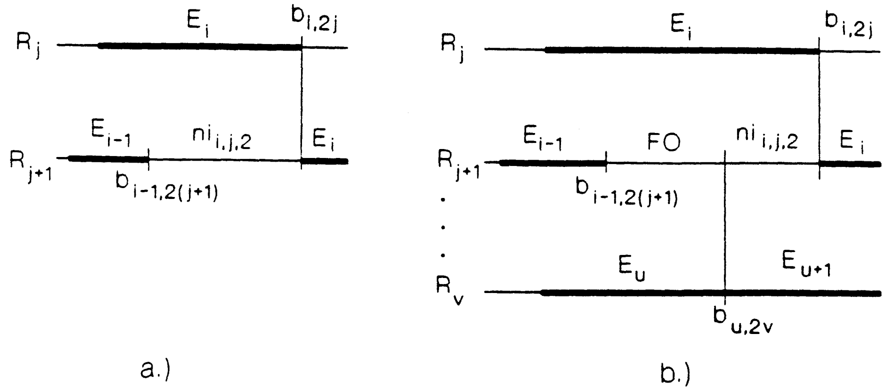

Figure 8.
Calculation of the NI(2) matrix.

5. CALCULATION OF THE VALIDITY RANGE

In case of zero-order perturbation the size of $fo_{i,j,k}$ and $ni_{i,j,k}$ may change if the perturbation appears only at one of the two entities which participate in the calculation of $fo_{i,j,k}$ and $ni_{i,j,k}$. If it appears at both entities then the size of $fo_{i,j,k}$ and $ni_{i,j,k}$ will not change.

Let $\mathbf{Z}$ denote a set of the (i, j) index-pairs belonging to those E_i at R_j which $fo_{i,j,k}$ and $ni_{i,j,k}$ $(k = 1, 2)$ change due to the effect of the introduced perturbation.

Theorem 1

Q_j, R_j $j = 1, \ldots, M$ and E_i $i = 1, \ldots, N$ determine a transfer line type queuing network with $c(j)$ queue capacities and FIFO queuing disciplines. A perturbation $(\delta_{x,y})$ is introduced on E_x at R_y, and has an effect of $\delta_{i,j}$ on E_i at R_j. Deterministic similarity holds as long as

$$(7) \qquad \begin{array}{ccc} \text{MAX}(ni_{i,j,k}, fo_{i,j,k}) \leq \delta_{x,y} \leq & \text{MIN}(ni_{i,j,k}, fo_{i,j,k}) \\ ni_{i,j,k} \leq 0 & ni_{i,j,k} > 0 \\ fo_{i,j,k} \leq 0 & fo_{i,j,k} > 0 \end{array}$$

$$i, x \in \{1, \ldots, N\}; \quad j, y \in \{1, \ldots, M\};$$
$$i, j \in \mathbf{Z}; \quad k = 1, 2;$$

Proof

Due to the propagation of $\delta_{x,y}$ the resulting $\delta_{i,j}$ may have the following effects on the FO and NI events,

If $\delta_{i,j} > 0$, then while R_j is working on E_i

- FO and NI may disappear after R_j,
- FO may appear on the preceding resources,
- NI may appear on the subsequent resources.

If $\delta_{i,j} < 0$, then while R_j is working on E_i

- FO and NI may appear after R_j,
- FO may disappear on the preceding resources,
- NI may disappear on the subsequent resources.

Identifying these cases in the FO(1), FO(2), NI(1) and NI(2) matrices directly comes (7).

■

6. EXTENSION OF THE PROBLEM

The calculation introduced in the previous sections is valid only in transfer line type networks. It is easy, however, to generalize the method to be valid for general networks (GN). Fig. 9. shows the type of network we want to deal with. We kept the notations of Fig. 1., but completed it with the routing parameter $(r_{i,j})$. There are two basic problem we have to cope with,

- entities may follow different routes in the network,
- entities may overtake other entities.

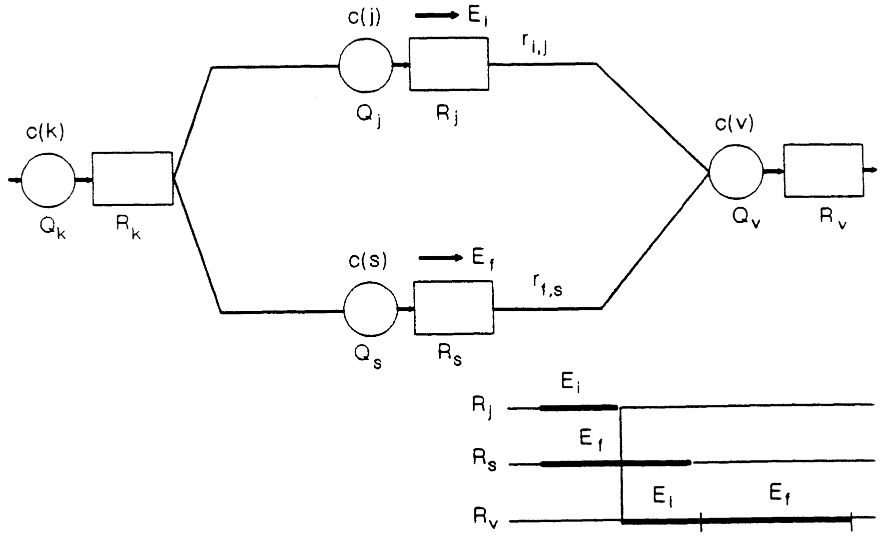

Figure 9.
Illustration of a general queuing network.

6.1. The effect of different routes

The possibility of the different routes makes necessary to revise that simple index identification of entities and resources we introduced at the transfer line type queuing networks.

Since in GN entities may follow different routes, the order of entities at the various resources may be distinct. Let us suppose that E_i is the $i-$th entity on R_j and leaving R_j it goes to R_v. Since R_v can be visited by entities from other resources as well we have no guarantee that E_i will also be the $i-$th at R_v. This phenomena causes the difficulty. In transfer lines we can identify entities by their place in the order at the various resources. At the calculation of the validity range we implicitly used this fact. To cope with GN we have to introduce a general entity identification.

Let $n_{i,j}$ denote the number in the order of E_i at R_j $i = 1, \ldots, N$, $j = 1, \ldots, M$ and $n^*_{k,j}$ $k = 1, \ldots, N$ the identification of the $k-$th entity at R_j.

The $n_{i,j}$ and $n^*_{i,j}$ matrices can be obtained from the simulation output. From the B matrix we can easily derive $n_{i,j}$ if we put in increasing order $b_{i,2j-1}$ or $b_{i,2j}$ $i = 1, \ldots, N$. In a transfer line case $n_{i,j} = i$ and $n^*_{(i,j)}$ identifies E_i.

A similar approach should be applied for the identification of resources. At the transfer line E_i always moves from R_j to R_{j+1}, in GN, however, it may move to any R_s $s = 1, \ldots, M$. The information of the routing in a sample path will be expressed in the routing matrix.

Let $r_{i,j}$ denote the index of the resource E_i enters after completing the service at R_j and $r^*_{i,j}$ the index of the resource E_i left before arriving to R_j.

The $r_{i,j}$ and $r^*_{i,j}$ matrices can be obtained from the simulation output. We can derive $r_{i,j}$ from the B matrix if we put in increasing order $b_{i,2j-1}$ or $b_{i,2j}$ $j = 1, \ldots, M$. The serial of j determines the route of E_i. In the transfer line case $r_{i,j} = j+1$ and $r^*_{i,j} = j-1$.

Applying these notations (1), (2), (3), (4), (5) and (6) can be transformed to be valid for general queuing networks. The transformation can be seen in the appendix.

6.2. The effect of overtake

Overtake of entities may occur as an effect of perturbation at assembly type queues. Let's suppose that both E_i from R_j and E_f from R_s goes to R_v and $b_{i,2j} < b_{f,2s}$ (Fig. 9.). As a consequence of the FIFO rule $n_{i,v} < n_{f,v}$, that is, E_i will get to R_v before E_f. If the perturbation changes the relationship of $b_{i,2j}$ and $b_{f,2s}$ the relationship of $n_{i,v}$ and $n_{f,v}$ will also change, violating this way the deterministic similarity. To avoid this, we have to ensure that the perturbation be small enough not to cause overtake of entities.

Let $\mathbf{A}$ denote the index set of the assembly type queues and $v \in \mathbf{A}$. If Q_v visited by E_i from R_j and E_f from R_s, then $ot_{i,f,v}$ will denote the time of E_f for not to be passed by E_i on R_v as a consequence of perturbation.

If $ot_{i,f,v} < 0$, then $n_{f,v} < n_{i,v}$, that is, E_f can overtake E_i. If $ot_{i,f,v} \geq 0$, then $n_{f,v} > n_{i,v}$, that is, E_i can overtake E_f. The overtake matrix is calculated as follows.

(8) $$ot_{i,f,v} = b_{f,2s} - b_{i,2j}$$

Let $\mathbf{W}$ denote a set of the (i, f) index-pairs belonging to those E_i and E_f which $ot_{i,f,v}$ $v \in \mathbf{A}$ changes due to the effect of the introduced perturbation.

From the point of view of the change of order of entities at the resources, deterministic similarity holds as far as the perturbation is small enough not to cause overtake, that is, if

$$
\begin{array}{ll}
(9) & \begin{array}{ccc} \text{MAX}(ot_{i,f,v}) \leq \delta_{x,y} \leq \text{MIN}(ot_{i,f,v}) \\ ot_{i,f,v} \leq 0 \qquad\qquad ot_{i,f,v} > 0 \end{array}
\end{array}
$$

6.3. Validity range for general queuing networks

The extension of theorem 1 for general queuing networks is as follows.

Theorem 2

Q_j, R_j $j = 1, \ldots, M$ and E_i $i = 1, \ldots, N$ determines a general queuing network with $c(j)$ queue capacities and FIFO queuing disciplines. A perturbation $(\delta_{x,y})$ is introduced on E_x at R_y, and has an effect of $\delta_{i,j}$ on E_i at R_j. Deterministic similarity holds as long as

$$
(10) \quad \begin{array}{c} \text{MAX}(ni_{i,j,k}, fo_{i,j,k}, ot_{i,f,v}) \leq \delta_{x,y} \leq \text{MIN}(ni_{i,j,k}, fo_{i,j,k}, ot_{i,f,v}) \\ \begin{array}{ll} ni_{i,j,k} \leq 0 & ni_{i,j,k} > 0 \\ fo_{i,j,k} \leq 0 & fo_{i,j,k} > 0 \\ ot_{i,f,v} \leq 0 & ot_{i,f,v} > 0 \end{array} \end{array}
$$

$$
i, f, x \in \{1, \ldots, N\}; \quad j, v, y \in \{1, \ldots, M\};
$$
$$
i, j \in \mathbf{Z}; \quad i, f \in \mathbf{W}; \quad k = 1, 2
$$

Proof

Completing the proof of theorem 1 with the limits imposed by the overtake restrictions, we can get (10).

■

7. IMPLEMENTATION OF THE ALGORITHM

Based on theorem 1 the validity range can be calculated by the following algorithm:

1. Various preliminary calculations are performed to obtainthe B^*, R, R^*, N, N^* furthermore the NI, FO, OT matrices and the **A** set.
2. The perturbation is propagated and the **Z** and **W** set is determined.

3. With theorem 1 or 2 the validity range is calculated.

The algorithm is polynomial type but, due to the matrices necessary to the calculation, a careful data organization is required. If N is the number of entities, M is the number of resources and F is the number of assembly type nodes then, the required number of input data is approximately $N(8M + FN) + F$. This formula shows that we have a quadratically increasing data requirement. In case of transfer line the quadratic tag is canceled out ($F = 0$).

As a consequence, in systems where N is high (i.g. in communication systems where the number of arriving entities may easily increase 10^4) data management may slow down the calculation. In manufacturing systems, especially when technology intensive parts are produced in small lots (e.g. FMS), N may not exceed even 10^2 and the algorithm can be organized easily. The computational time is insignificantly small.

The results provided by the algorithm are valid only for a specific sample path. Considering the stochastic similarity, the convergence properties of a single perturbation should be regarded here as well [3].

The algorithm was tested on various examples. We provide the results of a small system similar to the one in Fig. 9. The queue capacities used in the calculation are the following, $c(k) = 1$, $c(j) = 1$, $c(s) = 2$, $c(v) = 1$. 10 entity with identifications from 1 to 10 enter the system at Q_k and may randomly choose between R_j and R_s. They leave the system at R_v. The simulation was performed with the SIMAN simulation language. The identification of entities and the resulting B matrix of a sample path can be seen in Table 1. Note that the change of E_8 and E_7 is not a coincidence. It reflects that the order of entities at the entrance of the system are not the same as at the exit.

Table 1.

The simulation output (B matrix)

ID	BEG_k	END_k	BEG_j	END_j	BEG_s	END_s	BEG_v	END_v
E_1	0.0	3.0	3.0	8.0	___	___	8.0	20.0
E_2	3.0	6.0	___	___	6.0	12.0	20.0	24.0
E_3	6.0	10.0	10.0	17.0	___	___	24.0	30.0
E_4	10.0	14.0	___	___	20.0	25.0	30.0	33.0
E_5	14.0	21.0	24.0	27.0	___	___	33.0	35.0
E_6	24.0	29.0	___	___	30.0	38.0	38.0	43.0
E_8	34.0	39.0	39.0	44.0	___	___	44.0	57.0
E_7	29.0	34.0	___	___	38.0	46.0	57.0	61.0
E_9	39.0	42.0	44.0	47.0	___	___	61.0	66.0
E_{10}	44.0	52.0	___	___	57.0	64.0	66.0	71.0

Table 2.

Results of the calculations of perturbation analysis

Introduction		SLOPE	Validity range results				
RES.	IDEN.		LIMIT	IDEN.	ID/R	TYPE	
R_k	E_1	1.0	$-\infty$	—	—	—	LL
			∞	—	—	—	UL
R_k	E_2	0.0	−2.0	E_3	R_k	FO(1)	LL
			3.0	E_5	R_k	FO(1)	UL
R_k	E_3	0.0	−2.0	E_3	R_k	FO(1)	LL
			3.0	E_5	R_k	FO(1)	UL
R_k	E_4	0.0	$-\infty$	—	—	—	LL
			3.0	E_5	R_k	FO(1)	UL
R_k	E_5	0.0	$-\infty$	—	—	—	LL
			3.0	E_5	R_k	FO(1)	UL
R_k	E_6	1.0	−1.0	E_9	E_7	OT(v)	LL
			1.0	E_6	R_k	NI(2)	UL
R_k	E_7	1.0	−1.0	E_9	E_7	OT(v)	LL
			2.0	E_8	E_7	OT(v)	UL
R_k	E_8	1.0	−1.0	E_9	E_7	OT(v)	LL
			2.0	E_8	E_7	OT(v)	UL
R_k	E_9	0.0	$-\infty$	—	—	—	LL
			2.0	E_9	R_k	FO(1)	UL
R_k	E_{10}	0.0	−14.0	E_{10}	R_k	FO(1)	LL
			5.0	E_{10}	R_k	NI(2)	UL

Since SIMAN can directly communicate with FORTRAN [10] the programs for calculating the validity range were written in this programming language. The results can be seen in Table 2. The effect of 10 perturbation was studied. The perturbations were introduced in R_k at every entity independently. The SLOPE column shows the gradient of the through put time with respect to the operation time of the respective entities at R_k. These slopes are certainly valid only within the limits provided by the LIMIT columns. If the perturbation exceeds these limits the deterministic similarity is hurt, that is, higher order propagation rule should be applied. The IDEN. and ID/R columns of the result section identify the place, while the TYPE column identifies the reason of the limits. For example if we introduce a perturbation at E_6 in R_k then the lower limit will be caused by an overtake between E_9 and E_7 while the upper limit will be the result of a NI caused by E_6 when it is in R_k.

The algorithm was implemented in a practical case as well. The gradient of the throughput time and of the waiting time in queues were calculated for production control purposes in a continuous steel manufacturing process. The approximately 40 entities (steel slabs) and four resources (machines) did not cause any memory management problem and the insignificant computation time allowed on line gradient calculation. For further details about this project see Koltai *et al.* [9].

8. CONCLUSIONS

An algorithm was provided to calculate the validity range of deterministic similarity at the simulation of general queuing networks. The calculation provides information about the validity of the zero order gradient of the throughput time with respect to the operation time of entities at resources. The algorithm requires only the time schedule of the resources (B matrix) which can be provided easily by any simulation language in use today. The information gained by the calculation can be attached to the simulation output, providing useful information for parametric analysis of the throughput time.

The results can also provide basis for the calculation of the validity range of higher order perturbations, for the computation of multi variable gradient information, and for the examination of the gradient information concerning other parameters and performance measures, which are all subject of our further research. We expect that any result in these areas might support the improvement of the planning and dynamic control of discrete event dynamic systems.

9. REFERENCES

[1] **Cao. X.R.** (1988)."Realization Probability in Multi-class Closed Queuing Networks". *European Journal of Operations Research*, **36**, 393–401.

[2] **Gál, T.** (1979). *Postoptimal Analyses, Parametric Programming and Related Topics*. MacGraw-Hill Inc.

[3] **Heidelberg, P., Cao, X.R., Zazanis, M.A. & Suri, R.** (1988). "Convergence Properties of Infinitesimal Perturbation Analysis Estimates". *Management Science*, **34**, 1281–1302.

[4] **Ho, Y.C.** (1987). "Performance Evaluation and Perturbation Analysis of Discrete Event Dynamic Systems". *IEEE Transactions on Automatic Control*, **AC-32**, 563–572.

[5] **Ho, Y.C., Cao, X.R. & Cassandras, C.** (1983). "Infinitesimal and Finite Perturbation Analysis for Queuing Networks". *Automatica*, **19**, 439–445.

[6] **Ho, Y.C. & Cassandras, C.** (1983). "A New Approach to the Analysis of Discrete Event Dynamic Systems". *Automatic*, **19**, 149–167.

[7] **Ho, Y.C. & Li, S.** (1988). "Extension of Infinitesimal Perturbation Analysis". *IEEE Transaction on Automatic Control*, **AC-33**, 427–438.

[8] **Ho, Y.C., Suri, R., Cao, X.R., Diehl, G.W., Dille, L.W. & Zazanis, M.** (1984). "Optimization of Large Multi-class (non-product-form) Queuing Networks Using Perturbation Analysis". *Large Scale Systems*, **7**, 165–180.

[9] **Koltai, T., Onieva, L. & Larrañeta, A.** (1993). "Examination of the Sensitivity of an Operation Schedule with Perturbation Analysis". *International Journal of Production Research*, **31**, 2777–2787.

[10] **Pedgen, D.C., Shannon, R.E. & Sadowski, R.P.** (1990). *Introduction to Simulation Using SIMAN*. McGraw-Hill. New York.

[11] **Rubistein, R.Y.** (1986). *Monte Carlo Optimization, Simulation and Sensitivity of Queuing Networks*. John Wiley & Sons. New York.

10. APPENDIX

a) The examination of virtual queue

The transformed version of the algorithm to look for R_v, E_u is presented in Fig. 10, and the algorithm for finding R_p, E_g belonging to R_j, E_i when R_v, E_u is already known, is given in figure 11.

To facilitate further discussion let $b^*_{i,2j-1}$ denote the beginning and $b^*_{i,2j}$ the ending time of the operation of the $i-$th entity that is, entity $n_{i,j}$ on R_j.

```
nr := r_{i,j}                      (next resource)
ec := n*_{n_{i,nr}-c(nr),nr}       (entity for comparison)
fin:= 0
WHILE nr ∈ {1,...,M} AND ec ∈ {1,...,N} AND fin:= 0 DO
        BEGIN
        IF b_{ec,2nr} < b_{i,2j} THEN
            BEGIN
            j := nr
            i := ec
            nr := r_{i,j}
            ec := n*_{n_{i,nr}-c(nr),nr}
            END
        ELSE
            BEGIN
            v := j
            u := i
            fin:= 1
            END
        END
store (v, u)
```

Figure 10.
Pseudo code of the algorithm for searching E_u, R_v in general networks.

```
ec := n*_{n_{i,j}+c(j),j}          (entity for comparison)
pr := r*_{ec,j}                    (previous resorce)
fin:= 0
WHILE pr ∈ {1,...,M} AND ec ∈ {1,...,N} AND fin:= 0 DO
        BEGIN
        IF b_{ec,2pr} < b_{u,2v} THEN
            BEGIN
            i := ec
            j := pr
            ec := n*_{n_{i,j}+c(j),j}
            pr := r*_{ec,j}
            END
        ELSE
            BEGIN
            g := i
            p := j
            fin:= 1
            END
        END
store (p, g)
```

Figure 11.
Pseudo code of the algorithm for searching E_g, R_p in general networks.

b) Calculation of the FO(1) matrix

Recall the elements in figure 5a, 5b. In addition, let's suppose that $r_{i,j} = s$, that is E_i goes to R_s after completing the operation at R_j.

$$(1) \qquad fo_{i,j,1} = \begin{cases} b^*_{n_{i,s}-c(s),2s} - b^*_{n_{i,j},2j} & s \notin V(t) \\ b^*_{n_{u,v},2v} - b^*_{n_{i,j},2j} & \text{otherwise} \end{cases}$$

c) Calculation of the FO(2) matrix

Recall the elements in figure 6a, 6b. In addition, let's suppose that $r^*_{i,j} = s$, that is E_i visited R_s before arriving to R_j.

If, $s \notin V(t)$ then,

$$(2) \qquad fo_{i,j,2} = b^*_{n_{i,s}+c(j),2s} - b^*_{n_{i,j},2j}$$

If, $s \in V(t)$ then

$$(3) \qquad fo_{i,j,2} = \begin{cases} b^*_{n_{g,p},2p} - b^*_{n_{i,j},2j} & \text{if } b^*_{n_{g,p}+1,2(p-1)} \geq b^*_{u,2v} \\ b^*_{n_{g,p}+1,2p} - b^*_{n_{i,j},2j} & \text{otherwise} \end{cases}$$

d) Calculation of the NI(1) matrix

Recall the elements in figure 7a, 7b. In addition, let's suppose that $r^*_{i,j} = s$, that is E_i visited R_s before arriving to R_j.

If, $s \notin V(t)$ and $j \notin V(t)$ then,

$$(4) \qquad ni_{i,j,1} = b^*_{n_{i,s}+1,2s} - b^*_{n_{i,j},2j}$$

If, $s \notin V(t)$ and $j \in V(t)$ then,

$$(5) \qquad ni_{i,j,1} = b^*_{n_{i,s}+1,2s} - b^*_{n_{i,j},2j} - fo_{i,j,1}$$

If, $s \in V(t)$ then $ni_{i,j,1}$ has no meaning.

e) Calculation of the NI(2) matrix

Recall the elements in figure 8a, 8b. In addition, let's suppose that $r_{i,j} = s$, that is E_i goes to R_s after completing the operation at R_j.

$$(6) \qquad ni_{i,j,2} = \begin{cases} b^*_{n_{i,s}-1,2s} - b^*_{n_{i,j},2j} & s \notin V(t) \\ b^*_{n_{u,v},2v} - b^*_{n_{i,j},2j} & \text{otherwise} \end{cases}$$