Proceedings of the 1989 Winter Simulation Conference
E.A. MacNair, K.J. Musselman, P. Heidelberger (eds.)

MARKED EVENT METHOD IN DISCRETE EVENT SIMULATION

Shu Li
Department of Systems & Industrial Engineering
University of Arizona
Tucson, AZ 85721

ABSTRACT

In this paper, we develop a new method for sensitivity analysis of discrete event dynamic systems. The method is based on the observation that event lists under nominal system parameters can be shared by simulations with slightly perturbed parameters. Thus the simulation on different system configurations may be more efficient. The resulting algorithm is exact for Markovian systems, and approximate for non-Markovian systems. Experimental results are given to study the new algorithms.

1. INTRODUCTION.

Modelling, analysis, and optimization of discrete event dynamic systems (DEDS) are gaining more interest from both research institutions and industry. A typical discrete event system evolves from one state to the other driven by a large set of events occurring in discrete times. The occurrence of these events may be affected by the required logical conditions as well as the random phenomenon in the system. This is a common feature of most modern man-made systems such as flexible manufacturing systems. Examples of random discrete events in these systems are: "parts arrival", "machine down", etc.

Perturbation Analysis (PA), among others, is a relatively new technique aiming at more efficient use of simulation data for sensitivity analysis and optimization of DEDS (Ho (1987,1988)). Given a simulated or observed sample path of a discrete event dynamic system, PA attempts to derive information about perturbed systems without additional simulation or observation. Recently, the concepts of *state matching* and *event matching* have been proposed, leading to the Extended Perturbation Analysis (EPA) (Ho and Li (1988), Li (1989)) and the Event Matching Algorithm (EMA) (Ho, Li, and Vakili (1988)). The motivation of these methods is to overcome the limitations of infinitesimal perturbation analysis (IPA) (Heidelberger et al (1988)). Although EPA and event matching algorithm extend the applicability of IPA, they construct a perturbed sample path by selectively cut and paste portions of the nominal sample path, wasting some usable simulation data.

In this paper, we provide a new method, which extends the idea of EPA and EMA. The new method uses all portions of nominal sample path. The algorithm is exact for Markovian systems, and approximate for non-Markovian systems.

The paper is organized as follows: In section 2, we formulate our problem in the discrete event simulation model. In section 3, we present our main alogrithms. Simulation results are presented in section 4 to experimentally validate the new algorithms. The paper ends with a concluding remark.

2. PROBLEM FORMULATION.

Let $x(t,\theta)$ be a random process parameterized by a real number θ. Our purpose is to construct a realization of the perturbed process $x(t,\theta+\Delta\theta)$ from a given realization of the nominal process $x(t,\theta)$. We assume that the sample realization of the nominal process is generated by discrete event simulation. Sensitivity can be calculated from the nominal and the perturbed paths. To assist the presentation, we review the basic concept of discrete event simulation based on the generalized semi-markov process (GSMP) model (Barbour and Schassberger (1981)).

The basic building blocks of a discrete event simulation consist of a countable state space X, an allowable finite event list $\Gamma(x)$ for each state $x \in X$, a transition probability $p(\bullet;x,e)$ for $x \in X$ and $e \in \Gamma(x)$, and event life time distribution ϕ_e, $e \in \Gamma = \cup_{x \in X} \Gamma(x)$.

During simulation, at any time t, a current state x(t) and event list E(t) are maintained. A realization (path) can be generated as follows: At the initial time, the system is in state $x(0)=x^0$ with its event list $E(0)=\Gamma(x^0)$. For each e in E(0), event life time t(e) is generated according to ϕ_e's. Let t^* and e^* represent the smallest event time and the corresponding event respectively. Simulation clock is then advanced to the triggering time t^*. At t^*, e^* triggers the next transition to a new state $x(t^*)=x^1$ chosen according to the transition probability $p(\bullet;x^0,e^*)$. The new event list $E(t^*)$ is updated according the event scheduling rule $\Gamma(x^1)$. For any events $e \neq e^*$ in E(0), and $e \in \Gamma(x^1)$, e is allowed to continue in $E(t^*)$ with event time $t(e)-t^*$. New event times are generated in $E(t^*)$ for those events in $\Gamma(x^1)$ but not in E(0) according to ϕ_e's. With the new state and event list, the simulation continues in a similar fashion. In order to make the above meaningful, we assume that all events in $\Gamma(x)$ are different, and that at any time t, the event activation times are all different almost surly so that a unique triggering event is defined.

With the above GSMP formulation, we are in a position to discuss the main results.

3. A SIMPLE EXAMPLE.

In this section, we study a simple example to introduce the marked event idea.

Consider a simple Markov process (described in GSMP terminology) with state space $X=\{a_1,a_2,\ldots,a_{2N}\}$, where N is some positive integer. Let there be two types of events e_1 and e_2, with exponential event life times, respectively. Suppose that the allowable event list is $\Gamma(a_i)=\{e_1\}$ for $i=1,3,\ldots,2N-1$, and $\Gamma(a_i)=\{e_1,e_2\}$ for $i=2,4,\ldots,2N$. The transition probabilities are defined as follows: At any a_i, $i=1,3,\ldots,2N-1$, the only event e_1 triggers with probability one the transition to a_{i+1}. At any state a_i, $i=2,4,\ldots,2N$, e_1 triggers with probability 1 the transition to itself i.e. a_i, and e_2 triggers with probability 1 the transition to the next state a_{i+1} ($a_{N+1} \equiv a_1$). In our former notation, $p(a_{i+1};a_i,e_1)=1$, for $i=1,3,\ldots, 2N-1$; $p(a_i;a_i,e_1)=1$, and $p(a_{i+1};a_i,e_2)=1$ for $i=2,4,\ldots,2N$, where $a_{2N+1} \equiv a_1$. A sample realization for this simple system is shown in Fig.1. For more information on the issue of sample path analysis of GSMP, the reader is referred to Li (1988).

Now recall that our purpose is to construct a perturbed path (PP) from the given nominal path (NP) without an additional simulation. Let us consider the following perturbed process:

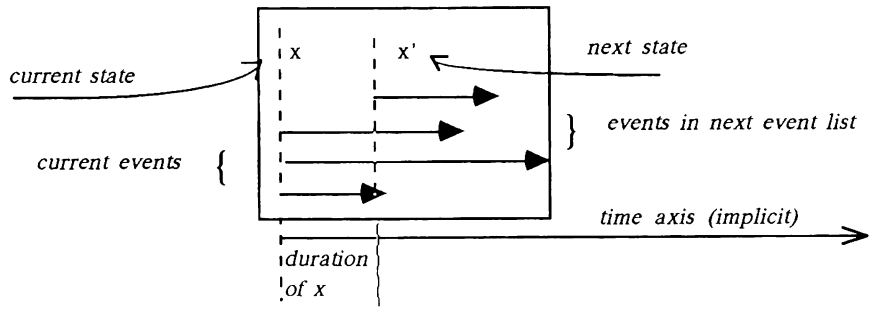
Perturbed system I. The perturbed system is the same as the nominal one except: i) $p(a_{i+1};a_i,e_1)=1-\Delta p$, $p(a_i;a_i,e_1)=\Delta p$, for i=1,3,..., 2N-1, where $\Delta p>0$ is some small number; ii) $p(a_i;a_i,e_1)=1-\Delta p$, $p(a_{i+1};a_i,e_1)=\Delta p$, for i=2,4,...,2N.

How do NP and PP differ? If the nominal and perturbed systems start from the same state, they become different when either of the following occurs:

i) At state 2k-1 (for k=1,2,...,N), with probability Δp, the next state on NP jumps to 2k, but the next state on PP stays the same, i.e. 2k-1.

ii) At state 2k, if e_1 triggers the next transition, then with probability Δp, the next state is 2k on NP but 2k+1 (2N+1≡1) on PP.

Definition of the graphic notation of GSMP used in the figures



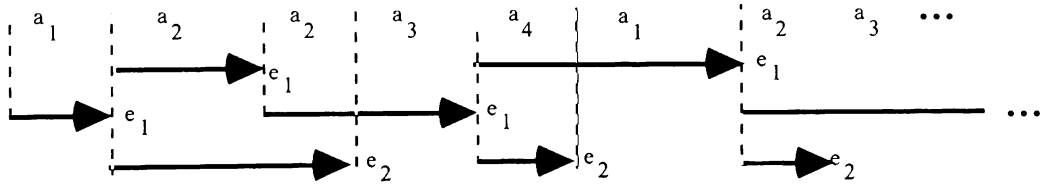


Fig.1. A realization of the nominal process.

We call the above *branching of states*. When a branching of states occurs, neither states nor event lists on NP and PP are the same. Let us now describe the idea of constructing a PP from the NP. At any time t, let the event lists be $E^{NP}(t)$ and $E^{PP}(t)$ on NP and PP, respectively. Let us study case i) first. Suppose at time t, both NP and PP are on state 2k-1, i.e. $E^{NP}(t)=E^{PP}(t)$, and that at the transition time t*, the next states on NP and PP are 2k and 2k-1, respectively. Thus, $E^{NP}(t^*)=\Gamma(2k)=\{e_1,e_2\}$, and $E^{PP}(t^*)=\Gamma(2k-1)=\{e_1\}$. An immediate observation is $E^{PP}(t^*)\subseteq E^{NP}(t^*)$. This shows that the nominal sample path carries all the information about events of the the perturbed path. To make use of this, we can simply mark the event e_1 in $E^{NP}(t^*)$. Then we know that $E^{PP}(t^*)$ equals to the marked subset of $E^{NP}(t^*)$. Thus we do not have to generate additional events for PP. However we still have to generate the state on PP, or at least to remember the difference from NP. Let us call such algorithm the Marked Event Method (MEM). Fig.2 shows how this procedure is used to construct a realization of the perturbed process from the given nominal path in Fig.1.

To compare with other algorithms, we apply EPA (Ho and Li (1988) and Event Matching Algorithm (Ho,Li, and Vakili (1989)) to the same problem as shown in Figures 3 and 4. We see that MEM makes more use of the given nominal data.

4. MARKED EVENT METHOD.

To generalize the idea discussed in the previous section, we need to address the following two problems: First, from Fig.2, e_2 is marked only for the last portion (see the shaded event). At the third state a_2 on PP, we used the residule life time of e_2 as though it were the newly generated event life time. Thus we need justify the correctness of the algorithm. Second, in the example, the event list on the perturbed path is always a subset of nominal one. However, it may not be the case for general systems. In this section, we answer these questions and present the main algorithm.

An important concept is the so called *statistical similarity* discussed in Ho and Li (1988). Two sample realizations are called statistical similar if they obey the same probabilistic law. In the construction of a perturbed path, the minimum requirement is that the constructed perturbed path is statistically similar to the one from brute force simulation.

The property for justifying the marked event procedure is based on the property of Markov process. Let there be m events at time t in the event list E(t), and that the residule life times of these events are: $R^1(t)$ ... $R^m(t)$. Suppose we replace $R^j(t)$, for some $1 \le j \le m$ by a new random variable $\underline{R}^j(t)$ independently generated from the same distribution. Clearly $\underline{R}^j(t) \neq R^j(t)$ a.s. Thus the sample path generated after this single replacement is different from the original one. The point is that the two paths are statistically similar! This is due to the memoryless property of the event life times of a Markov process. Let us summarize this observation as the following:

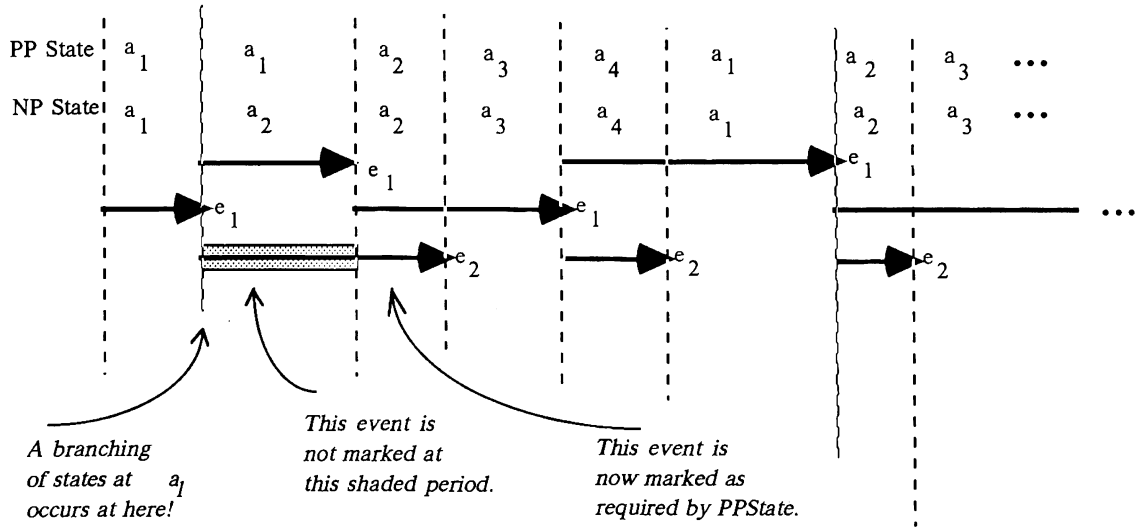


Fig.2. A PP constructed from the NP in (a) by MEM. This PP is represented by the unshaded part of NP events, along with the sequence of PPState.

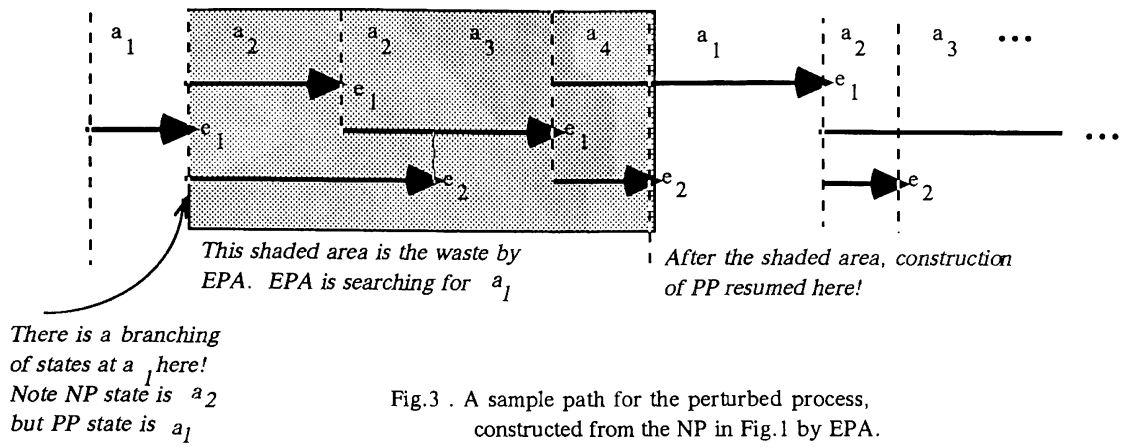


Fig.3 . A sample path for the perturbed process, constructed from the NP in Fig.1 by EPA.

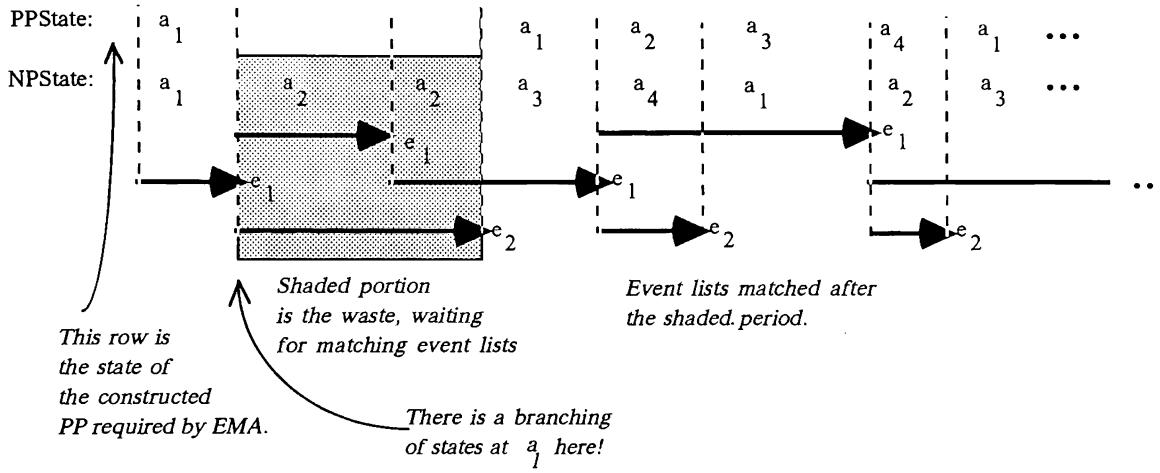


Fig.4. A PP constructed from the NP in (a) by event matching alg...

Invariance Property. Given a sample path x(t) of a Markovian GSMP, arbitrary replacement of any residual times in R(t) by an independent event life time of the same type generates a statistically similar sample path to the original one. □

We point out that the result reduces to the invariance under cut-and-paste property in Ho and Li (1988) if all the residual times are replaced simultaneously. Also note that the number of replacements along the time horizon is arbitrary, giving an infinitely number of different statistically similar paths.

It is readily seen that this invariance property guarantees the legitimacy of the marked event procedure.

Now let us answer the second question. Namely what should we do if the nominal event list does not include the perturbed one? In general, the events in the nominal and perturbed event lists can be partitioned into three parts. They are: $A=E^{NP}(t)\cap E^{PP}(t)$, $B=E^{NP}(t)\cap[E^{PP}(t)]^c$ (where the superscript c denotes complement of a set), and $C=[E^{NP}(t)]^c\cap E^{PP}(t)$. If C is empty, we can use the marked event procedure discussed in section 2 to construct the perturbed path. On the other hand, if C is not empty, we can simply generate those events in C, and mark them only

for the perturbed path. This procedure is again justified by the invariance property.

To show a case that C is not empty, let us consider the following example.

Example 4.1. Consider the same system as in section 3. Let us define the following perturbed system:

Perturbed system II. We modify the nominal system as follows: Let $p(a_{i+1};a_i,e_2)=1-\Delta p$, and $p(a_i;a_i,e_2)=\Delta p$, $i=2,4,\ldots,2N$. In this case, if both NP and PP are at state $2k$ and event e_2 triggers the next transition on NP, the next NP state will be $2k+1$ and the next PP state will be $2k$ with probability Δp. Thus $C=\{e_2\}$. An implementation of the above idea of generating new events is shown in Fig.5. □

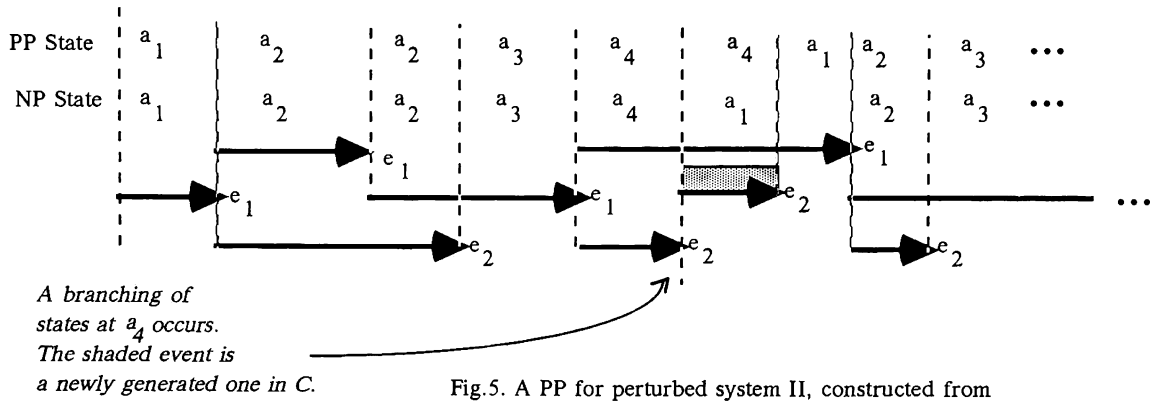


Fig.5. A PP for perturbed system II, constructed from the NP of Fig.1.

Now let us discuss the general Marked Event Algorithm. For easier presentation, we assume the following parameterization:

Assumption 4.1. The GSMP depends on θ only through the routing probabilities $p(\bullet;x,e)$'s. The event life time distributions ϕ_i's are independent of θ. The GSMP is Markov. □

The generalization to other parameterizations is straightforward as discussed in Li (1988). In the case of non-Markovian GSMP, the Marked Event algorithm is only approximate..

Let us first see how perturbation is generated from the given nominal sample path. From Assumption 4.1, perturbation comes from the routing probabilities. Let us distinguish the variables on NP and PP by subscripts θ and $\theta'=\theta+\Delta\theta$ (e.g. $p_\theta(\bullet;x,e)$ and $p_{\theta'}(\bullet;x,e)$), respectively. An apparent constraint is

$$\Sigma_{x'\in X}\Delta p_\theta(x';x,e)=0 \text{ for all } x\in X \text{ and } e\in\Gamma.$$

We show how we identify a branching of states. At each jump time t^*, the next state x' is determined by the triggering event e^*, the current state x, and the routing probability as: $x'=y$, if

$$\Sigma_{y\le x',y\in X}p_\theta(y;x,e^*)\le u<\Sigma_{y<x',y\in X}p_\theta(y;x,e^*) \quad (4.1)$$

where we assume there is a total ordering on the state space X, and u is a uniform random variable in [0,1]. To identify a branching of states, we use the same u, and apply (4.1) to $p_{\theta'}(\bullet;x,e^*)$.

With the above mechanism to identify a possible branching of states, we give the following schematic algorithm to construct a perturbed path:

Algorithm 4.1

Definition:

x_θ : state of nominal path;

$x_{\theta'}$: state of perturbed path; x^0 : initial state.

N : length of the nominal path (# transitions);

e^* : next triggering event on NP;

$C=[E^{NP}]^c \cap E^{PP}$

Initialize:

$x_\theta = x_{\theta'} = x^0$;

generate E^{NP} according to $\Gamma(x^0)$;

mark all events in E^{NP}; {initial NP events are also PP events}

C=empty set

Begin

for n:=0 to N

choose next event from $E^{NP} \cup C \rightarrow e^*$

if $e^* \in E^{NP}$ and is not marked do i)

else if $e^* \in E^{NP}$ and is marked, do i) and ii)

else if $e^* \in C$, do ii)

i) generate $x_\theta = p_\theta^{-1}(u; x_\theta, e^*), u \sim U(0,1)$}

ii) generate $x_{\theta'} = p_{\theta'}^{-1}(u; x_{\theta'}, e^*)$

update E^{NP} according to $\Gamma(x_\theta)$

if $[\Gamma(x_\theta)]^c \cap \Gamma(x_{\theta'})$ not empty, new events$\rightarrow$C

end.

Remark 4.1. It is interesting to note that the above algorithm can be interpreted as a kind of parallel simulation. Imagine that we simulate a set of N slightly different systems. We can use a similar algorithm to generate all the N simulations simultaneously. The triggering event is chosen each time from a minimal joint event list covering all the event lists for different systems. Thus N jobs share minimal amount of data in the simulation, offering more efficiency than simulating N jobs separately. We believe that this kind algorithm may also find application in a multiprocessor environment (another interpretation of parallel simulation).

We have seen that the Marked Event Method has the advantage of making use of the whole nominal trajectory, as opposed to earlier algorithms such as EPA and event matching algorithms. Hence, intuitively it is more efficient. The algorithm should be more advantageous for GSMPs which have large sets of events and simpler state specification. The efficiency also depends on how the simulation is implemented (e.g. the algorithm for updating states and events etc.). We do not intend to provide a theoretical evaluation at the present time. Instead, we will validate our intuition by simulation experiments.

5. EXPERIMENTAL RESULTS.

In this section, we perform several experiments to test the Marked Event Method (MEM). Throughput this section, we will use Algorithm 4.1. We will compare the results with those obtained by EPA (Ho and Li (1988)), Event Matching Algorithm (EMA) (Ho,Li, and Vakili (1988)), and brute force simulation (BF). Not all of the test systems necessarily require simulation. We choose them partially because of the exsistence of analytical results. Since efficiency is our major motivation, we define the following measure of efficiency for data utilization and CPU time utilization. Define

$$\text{cpu} = \frac{\text{cpu time for the specified algorithm}}{\text{cpu time for brute force simulation}}$$

$$\text{L.R} = \frac{\text{length to construct PP}}{\text{length for brute force simulation}}$$

where for a given simulation length L (in terms of transitions of the GSMP), the denominator is 2L (one for NP, one for PP), while the nominator represents the length of NP to obtain a PP of length L by either EPA or EMA or MEM.

Experiment 5.1. We use the system in section 3. Suppose we want to estimate the sensitivity of the steady state probability at state a_1 with respect to the perturbation Δp. With N=2, we calculate the theoretical values by solving

the balance equations. The theoretical values for the nominal and perturbed probabilities are given as:

$$\pi_{np}(1)=\frac{1}{N(1+\mu_1/\mu_2)},\quad \pi_{pp}(1)=\frac{1}{N[1+\mu_1(1-\Delta p)/(\mu_2+\mu_1\Delta p)]}$$

where $\pi_{np}(1)$ and $\pi_{pp}(1)$ are the steady state probabilities at state a_1 for nominal and perturbed systems respectively. The input parameters and experimental results are shown in Table 1 in the Appendix. The data is also plotted in Fig.6. It is readily seen that MEM is the most efficient one.

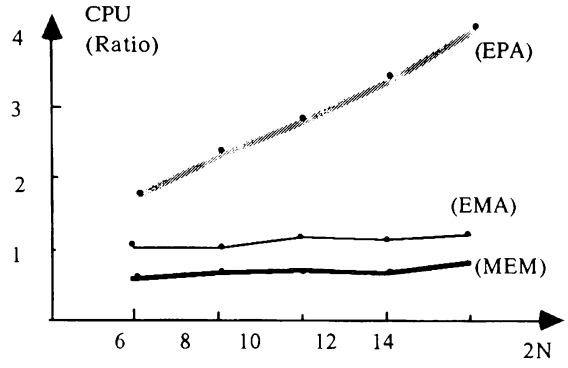


Fig.6

Experiment 2. Consider a closed single class queueing network, with 5 single server service stations $S_1,\ldots,S_5$, and infinite buffers. Let the service times be exponentially distributed with mean s_k for k=1,2,3,4,5. Our goal is to estimate the throughput sensitivity at S_1 with respect to some routing probability changes. Both the input parameters and the simulation results are shown in Table 5.2. The results show that MEM offers more savings than EPA and Event Matching method. □

Experiment 3. This network has been studied in Ho and Li (1988). In that paper, because of large number of states, approximate EPA has to be used. Let us simulate this system by MEM. There will be no approximation necessary by using MEM. The system is a closed multiple class queueing network. Each service station consists of a single server, a infinite buffer with FCFS discipline. Service time at a station is the same for all classes. Let s_i be the mean service time at the ith server. Performance measure is the throughput of class 3 at server 3. We calculate the throughput sensitivity with respect to routing parameter changes. Table 5.3 shows the input data and simulaiton results. The simulation result shows that MEM is more robust to problem size. The saving over brute force simulation seems to reduce as the problem size becomes large. The reason may be that updating state consumes most part of simulation. To have a better state representation and updating scheme may help raise the efficiency of MEM. □

Experiments also show reasonable accuracy for MeM to approximate non-Markovian systems. Due to space limitation, we omit this part.

6. CONCLUDING REMARKS.

We have presented a new class of perturbation analysis algorithms, the Marked Event algorithm. The new method may provide more efficiency than EPA and event matching algorithms. The efficiency has been supported by the simulation experiments performed in terms of CPU time ratio and extra run length. Experiments show that the MEM is quite robust to the problem size.

7. REFERENCES.

Barbour,A.D. and Schassberger,R. (1981), "Insensitive Average Residence Times in Generalized Semi-Markov Processes.", *Adv. Appl. Prob.*, 13, pp720-735.

Heidelberger, P., Cao, X., Zazanis, M.A., and Suri, R. (1988), "Convergence Properties of Infinitesimal Perturbation Analysis Estimates," manuscript.

Ho,Y.C. (1987), "Performance Evaluation and Perturbation Analysis of Discrete Event Dynamic System, Perspective and Open Problems", *IEEE Transaction on Automatic Control*, AC-32, 6.

Ho,Y.C. (1988), "Perturbation Analysis Explained", *IEEE Transaction on Automatic Control*, Vol.33,No.8.

Ho,Y.C. and Li,S. (1988), "Extensions of Inifinitesimal Perturbation Analysis", *IEEE Transaction on Automatic Control*, Vol.33,No.5,pp427-438.

Ho,Y.C., Li,S., and Vakili,P (1988), "On the Efficient Generation of Discrete Event Sample Paths", *Mathematics and Computers in Simulation*, No.30.

Law,A.M. and Kelton, W.D. (1982), *Simulation Modeling and Analysis*, McGraw Hall.

Li,S. (1988a), "Extended Perturbation Analysis of Discrete Event Dynamical Systems", Ph.D thesis, Applied Sciences, Harvard University, January.

Li,S. (1989), "A Tutorial on the Extended Perturbation Analysis Technique of DiscreteEvent Dynamic Systems," *Computer, Mathematics with Applications*, Vol.17,No.11,pp1467-1473.

AUTHORS' BIOGRAPHIES

SHU LI is an assistant professor in the Department of Systems and Industrial Engineering, University of Arizona. He received a BSEE from Huazhong University in 1982, MSEE from University of Illinois in 1985, and Ph.D from Harvard University in 1988. His research interests include discrete event dynamic systems, perturbation analysis, manufacturing systems, and decision making.

APPENDIX

TABLE 1

INPUT PARAMETER: μ_1=1/1.5, μ_2=1/2, Δp=0.5.
DEFINITION: (2N,L)=(# of states, simulation length in terms of transitions in CPP or PP);

SIMULATION RESULTS (Each entry of simulation is the average of 10 i.i.d simulations):

(2N,L)	πnp	πpp	πpp (BF)	πpp (EPA)	πpp (EMA)	$\Delta\pi/\Delta p$ (BF)	$\Delta\pi/\Delta p$ (EPA)	$\Delta\pi/\Delta p$ (EMA)	$\Delta\pi/\Delta p$ (MEM)
(6,5)	0.147374	0.242457	0.233399	0.243022	0.238095	-0.1902	-0.1859	-0.1952	-0.1855
True	0.142857	0.238095	0.238095	0.238095	0.238095	-0.1905	-0.1905	-0.1905	-0.1905
CPU						1.0	1.85	1.05	0.69 L.R.
						1.0	1.95	0.98	0.5
(8,6)	0.106151	0.179405	0.175115	0.179705	0.181527	-0.1465	-0.1372	-0.1479	-0.1508
True	0.107143	0.178571	0.178571	0.178571	0.178571	-0.1429	-0.1429	-0.1429	-0.1429
CPU						1.0	2.39	1.04	0.70
L.R.						1.0	2.49	0.97	0.5
(10,7)	0.084304	0.144099	0.143499	0.142455	0.143166	-0.1196	-0.1142	-0.1148	-0.1177
True	0.085714	0.142857	0.142857	0.142857	0.142857	-0.1143	-0.1143	-0.1143	-0.1143
CPU						1.0	2.90	1.12	0.75
L.R.						1.0	2.94	0.98	0.5
(12,8)	0.070326	0.119411	0.116720	0.119745	0.116892	-0.0982	-0.0905	-0.0959	-0.0931
True	0.071429	0.119048	0.119048	0.119048	0.119048	-0.0952	-0.0952	-0.0952	-0.0952
CPU						1.0	3.46	1.06	0.71
L.R.						1.0	3.43	0.99	0.5
(14,9)	0.060962	0.102151	0.102765	0.100968	0.102895	-0.0824	-0.0829	-0.0813	-0.0839
True	0.061224	0.102041	0.102041	0.102041	0.102041	-0.0816	-0.0816	-0.0816	-0.0816
CPU						1.0	4.04	1.13	0.76
L.R.						1.0	3.95	0.98	0.5

TABLE 2

INPUT PARAMETERS:

Mean service times: s_1=1.0, s_2=1.2, s_3=1.2, s_4=2.0, s_5=1.5.

Perturbation: Δp=0.1 (this perturbation is shown in rows 1,2,3 of the following routing probabilities)

Routings:

	Nominal Routing						Perturbed Routing				
	S_1	S_2	S_3	S_4	S_5		S_1	S_2	S_3	S_4	S_5
S_1	0.5	0.3	0.1	0.1	0.0	S_1	0.4	0.3	0.2	0.1	0.0
S_2	0.2	0.2	0.2	0.2	0.2	S_2	0.1	0.3	0.2	0.2	0.2
S_3	0.4	0.0	0.6	0.0	0.0	S_3	0.3	0.1	0.6	0.0	0.0
S_4	0.0	0.1	0.7	0.2	0.0	S_4	0.0	0.1	0.7	0.2	0.0
S_5	1.0	0.0	0.0	0.0	0.0	S_5	1.0	0.0	0.0	0.0	0.0

Initial state: (x_1,x_2,x_3,x_4,x_5)=(5,4,3,0,0), where x_i=# of customer at S_i.

DEFINITION: TP=Throughput at S_1.

SIMULATION RESULTS: (500000 customers/run, each entry is the average of 10 i.i.d runs)

TP_{np}	TP_{pp}(BF)	TP_{pp}(MEM)	$\Delta TP/\Delta p$(BF)	$\Delta TP/\Delta p$(MEM)
0.902593	0.619879	0.621042	-2.8271	-2.8155
CPU ratio:			1.0	0.78
L.R.			1.0	0.5

TABLE 3

INPUT PARAMETERS:

Mean service time: s_1=1.0, s_2=1.2, s_3=1.2, s_4=2.0, s_5=1.5.

Perturbation: Δp=0.2 (This perturbation is shown in the routing matrix for class 3 customers below)

Class 1:

Nominal Routing						Perturbed Routing
	S_1	S_2	S_3	S_4	S_5	
S_1	0.36	0.07	0.09	0.38	0.10	
S_2	0.29	0.21	0.28	0.14	0.08	*SAME AS NOMINAL*
S_3	0.14	0.08	0.26	0.25	0.27	
S_4	0.28	0.14	0.16	0.18	0.24	
S_5	0.21	0.05	0.15	0.16	0.43	

Class 2:

Nominal Routing						Perturbed Routing
	S_1	S_2	S_3	S_4	S_5	
S_1	0.24	0.01	0.21	0.23	0.31	
S_2	0.14	0.22	0.31	0.09	0.24	
S_3	0.11	0.29	0.26	0.20	0.14	*SAME AS NOMINAL*
S_4	0.20	0.18	0.13	0.41	0.08	
S_5	0.06	0.25	0.27	0.31	0.11	

Class 3:

	Nominal Routing						Perturbed Routing				
	S_1	S_2	S_3	S_4	S_5		S_1	S_2	S_3	S_4	S_5
S_1	0.0	0.0	1.0	0.0	0.0	S_1	0.0	0.2	0.8	0.0	0.0
S_2	1.0	0.0	0.0	0.0	0.0	S_2	1.0	0.0	0.0	0.0	0.0
S_3	0.0	0.0	0.0	1.0	0.0	S_3	0.0	0.2	0.0	0.8	0.0
S_4	0.0	0.0	0.0	0.0	1.0	S_4	0.0	0.0	0.0	0.0	1.0
S_5	1.0	0.0	0.0	0.0	0.0	S_5	1.0	0.0	0.0	0.0	0.0

Initial state: (x_1,x_2,x_3,x_4,x_5), where x_1=(1,1,1), x_2=(2,2), x_3=(3,3,3,3), x_4=( ), x_5=( ).

SIMULATION RESULTS: (500000 customers/run, each entry is the average of 10 i.i.d runs)

TP_{np}	TP_{pp}(BF)	TP_{pp}(MEM)	$\Delta TP/\Delta p$(BF)	$\Delta TP/\Delta p$(MEM)
0.918456	0.868081	0.866333	-0.2519	-0.2606
CPU ratio:			1.0	0.82
L.R.			1.0	0.5