Efficient Simulation of Delay in Tandem Networks Using Splitting*

Ahmet A. Akyamaç
Make Systems - Makelabs
200 Regency Forest Dr., Suite 150
Cary, NC 27511
e-mail: akyamac@makesys.com

J. Keith Townsend†
Dept. of Elec. and Comp. Eng.
North Carolina State University
Box 7914, Raleigh, NC 27695-7914
e-mail: jkt@eos.ncsu.edu

Abstract

The complexity of modern networks and the stringent quality of service requirements that result in very small important event probabilities can render standard Monte Carlo (MC) simulation intractable. Importance sampling (IS) provides many orders of magnitude speedup but is difficult to apply to tandem networks due to the indirect relationship between the modifiable input traffic parameters and the system response. Splitting can potentially overcome some of these difficulties, but conventional methods cannot be successfully applied to systems for which the occurrence of the important event does not temporally coincide with the conditions that lead to it. Cell delay experienced through single and tandem switches exhibits this behavior.

We develop two enhanced splitting methods based on a splitting technique which was previously used to estimate rare delay probabilities through a single ATM switch. The enhanced methods accurately capture the queueing behavior that leads to excessive cell delay through tandem switching networks. We use the enhanced methods to efficiently estimate rare delay probabilities for tagged traffic traversing tandem ATM switches in the presence of background traffic. The enhanced methods penetrate into a significantly higher delay (and lower probability) region compared to the previous method. Speedup over standard MC simulation is observed to be inversely proportional to the probability being estimated.

*This work was supported by the Center for Advanced Computing and Communication, Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC.

†Corresponding author, Tel. (919) 515-7353, Fax. (919) 515-2285.

1 Introduction

Performance estimation using simulation is an important step in the design of telecommunications systems. Due to the complexity of modern networks and stringent Quality of Service (QoS) requirements resulting in very low important event probabilities (e.g., cell loss probabilities below 10^{-9}), standard Monte Carlo (MC) simulation techniques can be computationally intractable. Importance sampling (IS) techniques [1] provide many orders of magnitude speedup but are usually difficult to implement for tandem networks due to the indirect relationship between the modifiable input traffic parameters and the system response (queues at the farther edge of the network).

Splitting can potentially overcome some of the difficulties associated with IS. Splitting techniques have recently been used in the context of queueing networks [2]-[9].

The main notion behind splitting is to partition the system states into subsets. When a subset is entered during system simulation, numerous retrials are initiated with the current subset as the starting or entry state of the system. Essentially, the main system trajectory is split into a number of subtrajectories. The number of subtrajectories spawned in each subset as a result of the split is referred to as the splitting or oversampling factor. The splitting method involves the selection of the subsets and oversampling factors, where the higher subsets correspond to the rare events, and increases the relative frequency of the rare events by placing an emphasis on higher subsets. Efficient splitting can be achieved by equalizing the steady state probabilities of the subsets under splitting [3, 4, 7, 8]. We apply the splitting technique called *direct probability redistribution* (DPR) [8], which requires the subsets to be disjoint (but not necessarily nested), and accounts for multiple subset transitions.

There are some systems for which the occurrence of the important event does not necessarily temporally coincide with the conditions that lead to the important event. Conventional splitting techniques cannot be applied to these systems due to the fact that even though numerous subtrajectories are spawned when conditions indicate a potential important event, the subtrajectories may be killed before the important event is actually observed. For example, in the case of cell loss and excessive queue occupation (queue tails), cells that observe

large queue lengths upon arrival typically directly contribute to the important event. However, in the case of excessive cell delay, which is also caused by high queue occupancy, the measurement of the delay event (which occurs when the cell exits the queue) does not coincide temporally with the large queue occupancy upon the arrival of the cell. In fact, by the time the cell exits, the system most probably exhibits a lower queue length, resulting in the subtrajectory being killed before the excessive delay event can be observed.

A splitting technique was presented in [10] for delay through queues with random service. An efficient application of splitting to more complicated configurations, including constant service times using a technique based on the concepts of *primary* and *secondary* indicator functions has been applied to successfully estimate rare delay probabilities in [11]. However, the technique developed loses efficiency for tandem queues.

In this paper, we enhance the concepts of primary and secondary indicator functions to more accurately represent the dynamics of the tandem queues. We develop two enhanced splitting methods to estimate rare delay probabilities resulting from tagged traffic traversing tandem ATM switches in the presence of background traffic. The enhanced methods accurately capture the queueing behavior that leads to excessive cell delay through tandem switching networks. We apply the enhanced methods to some of the configurations in [11] to demonstrate the more efficient results for tandem queues and also to a different example system which better illustrates the improvement of the enhanced methods. Application examples show that the enhanced methods penetrate into a significantly higher delay (and lower probability) region compared to the previous method. For the systems considered, speedup over standard MC simulation is observed to be inversely proportional to the probability being estimated.

This paper is organized as follows: A system description is given in Section 2. Following a brief description of the DPR method in Section 3, the concepts of primary and secondary indicator functions are described in Section 4. The method based on the standard primary and secondary indicator functions is presented in Section 5 and the two enhanced methods are presented in Section 6. Application examples are presented in Section 7, followed by a conclusion in Section 8.

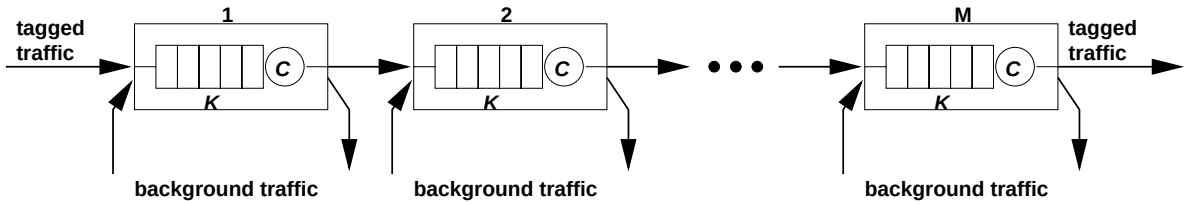

Figure 1: Network of tandem ATM switches.

2 System Description

We consider the network of M tandem ATM switches shown in Fig. 1. Each switch has a queue of size K and a constant service rate of C. There is a tagged traffic source with a triplet $(\hat{\lambda}_t, \bar{\lambda}_t, \hat{B}_t)$ which enters the network at switch 1 and exits at the output of switch M. Here, $\hat{\lambda}_t$ is the peak cell arrival rate, $\bar{\lambda}_t$ is the mean or sustained cell arrival rate and $\hat{B}_t$ is the mean burst length at the peak rate. For each switch there are N background sources with triplets $(\hat{\lambda}, \bar{\lambda}, \hat{B})$. The background traffic exits at the output of the corresponding switch, i.e., background traffic does not traverse the switches.

We assume the entire system is normalized with respect to $\max\{\hat{\lambda}_t, \hat{\lambda}\}$ and thus consider time to be discrete, where one time slot corresponds to the arrival time of a single cell at the peak rate $\max\{\hat{\lambda}_t, \hat{\lambda}\}$. With this normalization, we assume that for all sources, cells arrive according to an interrupted Bernoulli process (IBP). The IBP sources can be in one of two states $s \in \{0, 1\}$. All queues exhibit an early arrival and late departure behavior. Tagged cells arrive at the first line at each switch.

Tagged cells experience a delay of D_i at switch i, $1 \leq i \leq M$. Thus, the overall delay is given by $D = \sum_{i=1}^{M} D_i$. For the tagged source, we are interested in *delay threshold probabilities* (the probability that the delay exceeds a given threshold), $\Pr\{D > \tau\}$ where D is the delay experienced by the tagged cells and τ represents different threshold values. We are interested in cases where τ is high and thus the delay threshold probabilities are very small (in the range of 10^{-6} and below).

3 The DPR Simulation Technique

The DPR simulation technique was developed in detail in [8]. Here, we summarize some key results. Let the state space of the system be denoted by $\mathcal{S}$ and assume there are n states. The system state is defined by a set of system parameters. The individual states are indexed from $\mathcal{S} = \{1, \ldots, n\}$. Let $\boldsymbol{V}_0, \boldsymbol{V}_1, \boldsymbol{V}_2, \ldots$ ($\boldsymbol{V}_k \in \mathcal{S}$) represent the system evolution (the states that are visited in successive observation points). Using DPR, the set $\mathcal{S}$ is partitioned into m disjoint, nonempty, indexed subsets $\mathcal{S}_1, \ldots, \mathcal{S}_m$. The partitioning is uniquely defined by a subset indicator function:

$$\Gamma(\boldsymbol{V}_k) \in [1, m]: \qquad \Gamma(\boldsymbol{V}_k) = i \Longleftrightarrow \boldsymbol{V}_k \in \mathcal{S}_i \tag{1}$$

Let $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_m\}$ denote the *oversampling vector*. We assume that $\mu_1 = 1$ and $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_m$. If this condition is not satisfied, the subset indices described by $\Gamma(\cdot)$ can be reordered according to $\boldsymbol{\mu}$ and normalized to form a set of indices described by a new subset indicator function $\Gamma'(\cdot)$ such that $\mu'_1 = 1$ and $\mu'_1 \leq \mu'_2 \leq \ldots \leq \mu'_m$. Subsequently, $\Gamma'(\cdot)$ would replace $\Gamma(\cdot)$ as the subset indicator function. Using DPR, each subset $\mathcal{S}_i$ is visited μ_i times more often than it would have been using MC simulation [8]. Denoting the steady state probabilities of the original and m-partitioned system by $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_n\}$ and $\boldsymbol{\pi}^{(m)} = \{\pi_1^{(m)}, \ldots, \pi_n^{(m)}\}$, respectively, we have that $\pi_l^{(m)} = \Phi \cdot \mu_{\Gamma(l)} \pi_l$, for $1 \leq l \leq n$, where $\Phi = 1/\sum_{l=1}^{n} \mu_{\Gamma(l)} \pi_l$ is a normalization constant.

During DPR simulation, a trajectory is split whenever the system makes a transition from subset $\mathcal{S}_i$ to subset $\mathcal{S}_j$ such that $i < j$. The expected number of new subtrajectories generated upon such a transition is $\mu_j/\mu_i - 1$. The life of the new sub-trajectories is controlled by a *ticketing* mechanism [8]. Unbiased estimates can be obtained when the event counters are updated by weighting by a factor of $1/\mu_i$ when in subset $\mathcal{S}_i$, or $1/\mu_{\Gamma(\boldsymbol{V})}$ in general, where $\boldsymbol{V}$ is the current state of the system. Note that, as we indicated in Section 1, the splitting method involves the selection of the subsets and oversampling factors. Efficient splitting can be achieved by equalizing the steady state probabilities of the subsets under splitting [3, 4, 7, 8].

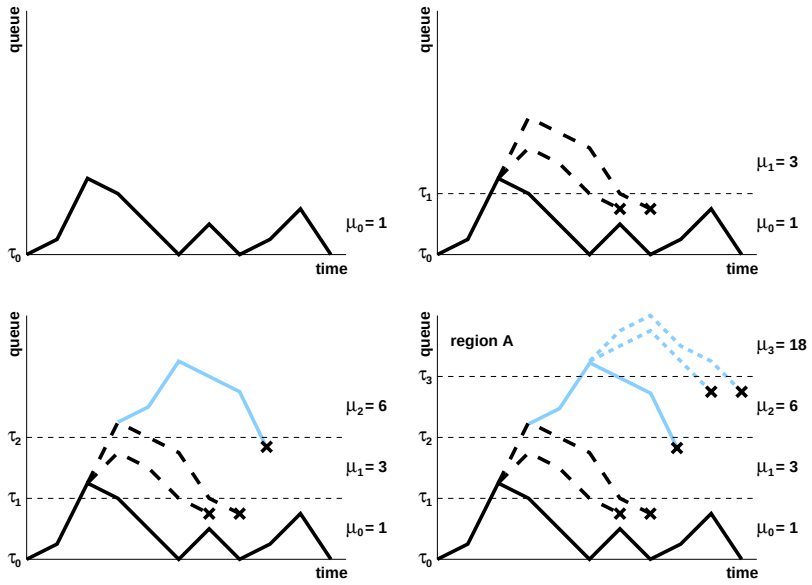

Figure 2: Example showing the splitting subtrajectories.

For example, assume we are interested in rare queue tail probabilities in a single queue system (as in Fig. 1 with $M = 1$). Denote the event that the queue length exceeds τ by A. We wish to estimate the probability $\gamma = P(A)$. For queue tail probabilities, we can define the state of the system to be q_k, where q_k is the queue length at time k. We can choose the subsets $\mathcal{S}_i$ to represent thresholds τ_i of the queue length. If there are m such subsets, then $\tau_0 \leq \ldots \leq \tau_{m-1}$ and $\tau \geq \tau_{m-1}$. An illustration of a set of subtrajectories that might result from applying the splitting method to this queueing system is shown in Fig. 2, where $m = 4$, $\tau_0 = 0$ and $\tau = \tau_3$ defines the important region A. The oversampling factors are given by $\mu_0 = 0$, $\mu_1 = 3$, $\mu_2 = 6$ and $\mu_3 = 18$. In Fig. 2, the solid black curve represents a typical trajectory that would be taken by MC simulation and the remaining curves represent trajectories initiated by splitting. The trajectories represented by the dashed gray curves penetrate into the important region A. The cross marks indicate terminated subtrajectories. Note that, as indicated in Section 1, such a splitting method cannot be used to estimate rare delay probabilities since the measurement of the delay event does not coincide temporally with the large queue occupancy upon the arrival of a cell.

The equalization of the visiting probabilities of the subsets can be achieved by choosing an oversampling vector $\boldsymbol{\mu}$ such that the resulting unweighted or raw subset probabilities are

roughly equal [3, 7, 8]. Equalized hit probabilities can be achieved by setting the factors in $\boldsymbol{\mu}$ according to the inverse of the estimated steady state probabilities. Denoting the latter by $\hat{P}_1, \ldots, \hat{P}_m$, where $\hat{P}_i = e_i / \sum_{j=1}^{m} e_j$, $\boldsymbol{\mu}$ can be constructed such that $\mu_k = (\max_i \hat{P}_i)/\hat{P}_k$ for $1 \leq i \leq m$. We use *iterative subset exploration* to estimate such an oversampling vector $\boldsymbol{\mu}$ [8, 9]. We start with a simple MC simulation (DPR simulation with $\boldsymbol{\mu}$=**1**) which provides relatively accurate estimates of the high probability subsets. Using a loose confidence constraint, we produce the first oversampling vector. Typically, assuming the subset indicator function is properly chosen, a subsequent run with the new oversampling vector will give further details about the higher subsets and can be used to generate the next oversampling vector. We repeat the above iterative exploration process until the proper oversampling vector which covers all subsets is obtained.

Thus, in the subsequent sections, we focus on the selection of a subset indicator function to accurately estimate rare delay threshold probabilities through tandem queues.

4 Primary and Secondary Indicator Functions

We illustrate the technique of splitting based on primary and secondary indicator functions [11] using an example. We consider a system such as in Fig. 1 with $M = 1$ (single stage queue) with input consisting of a tagged source multiplexed with background sources. We estimate delay threshold probabilities $\Pr\{D > \tau\}$ for the tagged cells.

Upon entry into the queue, the tagged sources are given a timestamp based on the arrival time slot. Upon exit, the delay is determined by the difference between the exit slot and the timestamp. Fig. 3 represents a sample evolution of such a system with a constant service rate of $C = 1$ cell/slot between observation time slots $k = 0$ and $k = 40$. In Fig. 3, the solid lines represent the queue length q_k. In this example, a tagged cell arrivals at slot $k = 16$ observing a queue length of 10 and exits at slot $k = 27$.

For this system, a traditional subset indicator function would be in the form of $\Gamma(V_k) = q_k + 1$. Thus, upon the arrival of the tagged cell, the indicator function is $\Gamma(V_{16}) = 11$ with an oversampling factor of μ_{11}, resulting in new subtrajectories. However, as seen in

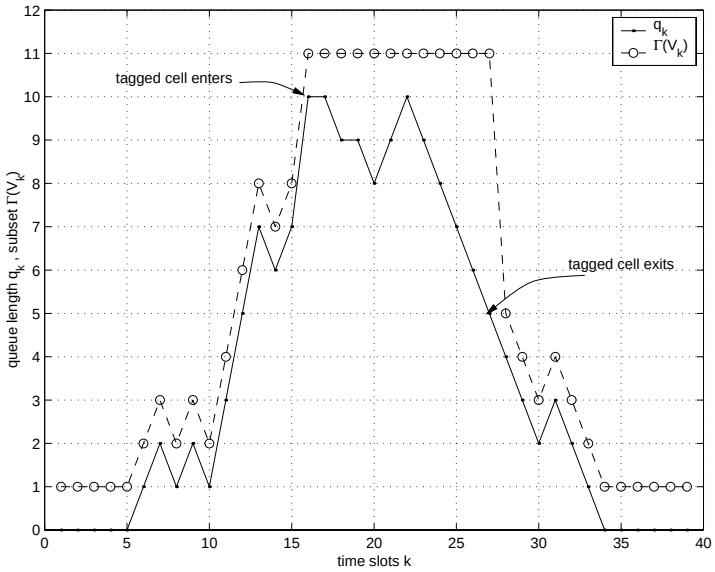

Figure 3: Sample evolution and the primary/secondary indicator functions.

Fig. 3, by the time the tagged cell exits, the queue length will have already dropped to $q_k = 5$. The subset just before the exit of the tagged cell is $\Gamma(V_{26}) = 6$. Thus, even though new subtrajectories were initiated to observe more delay events for tagged cells, those subtrajectories terminated before the tagged cell departed from the queue. The traditional subset indicator function does not account for the time lag between the time when the conditions for the important event occur and the time when the important event is actually observed. In fact, the spawned subtrajectories will not result in new important events being collected and will cause computational inefficiency.

Thus, a good indicator function must not only cause the system states that can potentially lead to important events to be frequently visited, but it must keep the spawned subtrajectories alive until the important events have occurred. In [11], a DPR-based splitting approach based on primary and secondary indicator functions was introduced.

The objective of the primary indicator function (PIF) is to increase the production of the cells that have the potential to experience long delay. This can be achieved by using $T(V_k) = q_k + 1$ as the PIF since this PIF enhances those subtrajectories for which entering tagged cells observe large queues. The PIF maintains the first objective of a good indicator function.

The objective of the secondary indicator function (SIF) is to keep the subtrajectories that will lead to important events alive. This can be achieved with the use of an auxiliary state variable [11]. When a tagged cell enters the queue, a label ϕ (called the *potential*) is added to it which represents the current PIF. Thus, if a tagged cell enters at time k, $\phi = T(V_k)$. Then, the overall potential can be found by maximizing over all tagged cells L_k in the queue at time k as follows:

$$\bar{\phi}_k = \max_{i \in \{1,\ldots,L_k\}} \phi_k(i) \tag{2}$$

where $\phi_k(i)$ is the potential for tagged cell i at time k. Then, the overall indicator function can be found by maximizing the PIF and SIF as follows:

$$\Gamma(V_k) = \max\{T(V_k), \bar{\phi}_k\} \tag{3}$$

We will use the term PSIF to denote the indicator function $\Gamma(V_k)$ using the PIF and SIF in [11]. The PSIF function $\Gamma(V_k)$ is illustrated using dashed lines in Fig. 3. As can be seen from the figure, the $\Gamma(V_{16}) = 11$ upon the arrival of the tagged cells and does not change until the departure of the cell in slot $k = 27$. Thus, the subtrajectory is not terminated until the delay of $D = 11$ slots is observed for the tagged cell.

The PSIF function has been successfully applied to single stage systems in [11]. However, it loses efficiency as the number of tandem queues is increased. In the following, we present an enhanced method based on the PSIF that generates efficient results for tandem queues.

5 PSIF-Based Subset Indicator Function

For the tandem configuration, the primary indicator function used by the PSIF approach is calculated as follows:

$$T(V_k) = \sum_{i=1}^{M} q_k(i) + 1 \tag{4}$$

where $q_k(i)$ is the occupancy of queue i at time k. Upon arrival, a tagged cell is labeled with the above indicator function with $\phi_k = T(V_k)$. The labels are used to generate the secondary indicator function $\bar{\phi}_k = \max_{j \in \{1,\ldots,L_k\}} \phi_k(j)$ where $\phi_k(j)$ is the label for tagged cell j at time k and $\bar{\phi}_k$ is the secondary indicator function at time k.

The PSIF function loses efficiency as the number of tandem switches (M) is increased since the potential delay for the tagged cell is equal to the sum of the queue lengths upon arrival. However, the queue lengths change as the tagged cell traverses the tandem queues. Thus, the secondary indicator function becomes less and less accurate as M is increased because it cannot predict the queue occupancies which will be observed as the tagged cell traverses the network. In fact, in many cases, a tagged cell with a very high potential based on the queue lengths at its arrival time will experience delays which are considerably lower than the potential. This causes a loss of efficiency as observed in [11].

6 Enhanced Subset Indicator Functions

Here, we present two enhanced indicator functions based on the concept of primary and secondary indicator functions. We will use the term EIF to denote the enhanced indicator functions.

The first function, which we call EIF Type 1, does not use any prediction of potential delay, but rather "keeps track" of the delay experienced by the tagged cell as it traverses the network. The primary indicator function is given by the queue length observed in the first switch as follows:

$$T^*(V_k) = q_k(1) + 1 \tag{5}$$

where $(^*)$ indicates the EIF Type 1 approach and $q_k(1)$ is the current queue length at switch 1. Thus, for a tagged cell that enters the network, the enhanced secondary indicator function is given by $\phi_k^* = T^*(V_k)$. For a tagged cell that exits switch s and enters switch $s+1$, $1 \leq s \leq M-1$, the enhanced potential function is computed by calculating the delay observed until switch s and the current queue length at switch $s+1$ as follows:

$$\phi_k^* = \sum_{i=1}^{s} D_i + q_k(s+1) + 1 \tag{6}$$

where D_i is the delay experienced by the tagged cell through switch i and $q_k(s+1)$ is the currently observed queue length at switch $s+1$. Note that even though we preserve the name potential function for ϕ^* for the EIF Type 1, it represents the actual delay experienced

by the tagged cell as it traverses the network. The enhanced secondary indicator function at time k is then computed as:

$$\bar{\phi}_k^* = \max_{j \in \{1,\ldots,L_k\}} \phi_k^*(j) \tag{7}$$

where $\phi_k^*(j)$ is the potential function using EIF Type 1 for tagged cell j at time k and L_k is the number of tagged cells in the network at time k. The EIF Type 1 function is then given as follows:

$$\Gamma^*(V_k) = \max\{T^*(V_k), \bar{\phi}_k^*\} \tag{8}$$

Since the EIF Type 1 indicator function does not make any predictions on the potential delay of the tagged cell (except for the currently observed delay and the delay to be observed in the next switch), it does not suffer the loss of accuracy as in the case of the PSIF method. Thus, it is expected to generate more efficient results than the PSIF method (resulting in higher observed delays). However, since no prediction is employed, it is also expected that EIF Type 1 may lose efficiency as the number of tandem queues is increased due to the fact that a tagged cell has to traverse multiple tandem switches to sufficiently increase the indicator function and generate subtrajectories which potentially lead to large delay observations.

The second function, which we call EIF Type 2, uses forward prediction as in the PSIF case, but updates and enhances the prediction of potential delay as the tagged cells traverse the network. For the EIF Type 2, the primary indicator function is calculated as in the case of $T(V_k)$ for the PSIF in Eq. 4 as follows:

$$T^{**}(V_k) = \sum_{i=1}^{M} q_k(i) + 1 \tag{9}$$

where $(^{**})$ indicates the EIF Type 2 approach and $q_k(i)$ is the current queue length at switch i. For this system, this choice of primary indicator function represents the best prediction of the delay of the newly arriving tagged cells.

However, in order to better represent the queueing behavior in the tandem switches down the line, we update the secondary indicator function as the tagged cells traverse the network. Essentially, for a tagged cell that enters the network, the enhanced secondary

Method	Primary Indicator Func., $T(V_k)$	Potential Func. ϕ_k, upon entry	Potential Func. ϕ_k, traverse switch $s \rightarrow s+1$
PSIF	$\sum_{i=1}^{M} q_k(i) + 1$	$T(V_k)$	unchanged
EIF Type 1	$q_k(1) + 1$	$T(V_k)$	$\sum_{i=1}^{s} D_i + q_k(s+1) + 1$
EIF Type 2	$\sum_{i=1}^{M} q_k(i) + 1$	$T(V_k)$	$\sum_{i=1}^{s} D_i + \sum_{i=s+1}^{M} q_k(i) + 1$

Table 1: Comparison of the functions used by the three methods.

indicator function is given by $\phi_k^{**} = T^{**}(V_k)$. For a tagged cell that exits switch s and enters switch $s+1$, $1 \leq s \leq M-1$, the enhanced potential function is computed by calculating the delay observed until switch s and making an updated prediction for the potential delays through switches $s+1$ to M as follows:

$$\phi_k^{**} = \sum_{i=1}^{s} D_i + \sum_{i=s+1}^{M} q_k(i) + 1 \tag{10}$$

where D_i is the delay experienced by the tagged cell through switch i and $q_k(i)$ is the currently observed queue length at switch i. The enhanced secondary indicator function at time k is then computed as:

$$\bar{\phi}_k^{**} = \max_{j \in \{1,\ldots,L_k\}} \phi_k^{**}(j) \tag{11}$$

where $\phi_k^{**}(j)$ is the potential function using EIF Type 2 for tagged cell j at time k and L_k is the number of tagged cells in the network at time k. The EIF Type 2 function is then given as follows:

$$\Gamma^{**}(V_k) = \max\{T^{**}(V_k), \bar{\phi}_k^{**}\} \tag{12}$$

The EIF Type 2 approach is expected to generate more efficient results (i.e., larger observed delays) than the PSIF and EIF Type 1 methods since its uses prediction-based indicator functions that are updated to represent the queue changes as the tagged cell traverses the network. The prediction function enables trajectories which will potentially result in generation of large delays. The updated predictions ensure that only those subtrajectories that will eventually result in higher delays are kept alive. Table 1 shows a comparison of the primary indicator function and potential functions (both upon entry and after a traversal from switch s to $s+1$) for the three methods.

In Fig.'s 4 and 5, we present a high level pseudo-code section depicting the EIF calculation phase for both Type 1 and Type 2 and system transition at time k. Note that due to the early arrival and late departure behavior of the queues, the algorithm logically behaves as if the processing occurs from switch M down to switch 1 (this is equivalent to the actual operation of the system where the queues are processed simultaneously).

```
/* EIF calculations */

{ T(V_k) := q_k(1) + 1 or                    /* Primary indicator function, EIF Type 1 */
T(V_k) := sum_{s=1}^{M} q_k(s) + 1 }         /* Primary indicator function, EIF Type 2 */
φ*_k := max_{j∈{1,...,L_k}} φ*_k(j)          /* Secondary indicator function */
Γ(V_k) := max{T(V_k), φ*_k}                  /* The EIF function */

for i = M downto 1 do                        /* Process all queues */

    /* First, the arrivals */

    if i == 1 then                           /* Tagged cell arrivals only at switch 1 */
        if tagged_cell_arrival then
            if q_k(1) + 1 ≤ K then
                φ* := T(V_k)                 /* Label the tagged cell */
                q_{k+1}(1) := q_k(1) + 1     /* Increment queue */
            else
                tagged_cell_lost
            end
        end
    end
    for k = 1 to N do                        /* Background sources */
        if background_cell_arrival then
            if q_k(i) + 1 ≤ K then
                q_{k+1}(i) := q_k(i) + 1     /* Increment queue */
            else
                background_cell_lost
            end
        end
    end
Continued in Fig. 5 ...
```

Figure 4: Pseudo-code section showing the EIF calculation and arrival procedure.

```
... continued from Fig. 4

    /* Next, the service */

    if i == M then                          /* Delay calculated only at exit of switch M */
        if q_k(M) > 0 then
            q_{k+1}(M) = q_k(M) - 1                                   /* Cell departs */
            if tagged_cell_departed then
                D := sum_{s=1}^{M} D_s                                /* Calculate final delay */
            else
                background_cell_departed
            end
        end
    else                                    /* Cell departures for the other switches */
        if q_k(i) > 0 then
            q_{k+1}(i) := q_k(i) - 1                                  /* Cell departs */
            if tagged_cell_departed then
                if q_k(i+1) + 1 <= K then       /* Tagged cell traverses to next switch */
                    { phi* = sum_{s=1}^{i} D_i + q_k(i+1) + 1 or /* Update the EIF Type 1 label */
                    phi* = sum_{s=1}^{i} D_i + sum_{s=i+1}^{M} q_k(s) + 1 } /* Update the EIF Type 2 label */
                    q_{k+1}(i+1) := q_k(i+1) + 1
                else
                    tagged_cell_lost
                end
            else
                background_cell_departed
            end
        end
    end
end
```

Figure 5: Pseudo-code section showing the service procedure.

7 Application Examples

We first consider the system in [11]. For this system, $\hat{\lambda}_t = 1$, $\bar{\lambda}_t = 0.1$, $\hat{B}_t = 10$ for the tagged source and $\hat{\lambda} = 1$, $\bar{\lambda} = 0.4/7$ and $\hat{B} = 10$ for $N = 7$ background sources at each switch. The queue size is $K = 500$ and $C = 1$ for each switch.

We generated *delay threshold probability* results for this system for $M = 2$ and $M = 5$

switches (note that the delay threshold probability is the probability that the cell delay exceeds a specified threshold). For each M, the minimum delay is given by $D_{min} = M$ and the maximum delay is given by $D_{max} = M \cdot K$. Delay threshold probabilities are generated for $D_{min} \leq \tau \leq D_{max} - 1$. By definition, $\Pr\{D > \tau\} = 1$ for $0 \leq \tau \leq D_{min} - 1$ and $\Pr\{D > \tau\} = 0$ for $\tau = D_{max}$. For each M, we run sets of simulations (resulting in N_{DPR} simulation slots) that generate delay threshold probability estimates $\hat{p}_\tau = \Pr\{D > \tau\}$ and variance estimates $\hat{\sigma}^2(\hat{p}_\tau)$ for each threshold $D_{min} \leq \tau \leq D_{max} - 1$. The improvement factors (speedup over traditional MC simulation) are estimated by calculating the number of MC simulation slots required to estimate $\hat{p}_\tau$ (using Bernoulli trials) with a variance of $\hat{\sigma}^2(\hat{p}_\tau)$ and dividing by N_{DPR} as follows:

$$\hat{R}_{net} = \left. \frac{N_{MC}}{N_{DPR}} \right|_{\hat{\sigma}^2_{MC} = \hat{\sigma}^2_{DPR}} = \frac{\hat{p}_\tau(1 - \hat{p}_\tau)}{N_{DPR} \cdot \hat{\sigma}^2(\hat{p}_\tau)} \tag{13}$$

Note that for $M = 1$, the PSIF, EIF Type 1 and EIF Type 2 methods are equivalent. The results of the simulation runs are plotted in Fig.'s 6 to 9.

In Fig. 6, the delay threshold probabilities are plotted as solid curves for the EIF Type 2 method and dashed curves for the EIF Type 1 method. The number of simulation slots was $N_{DPR} = 40 \times 10^7$ for both $M = 2$ and $M = 5$ for all methods. The black circular marks indicate the maximum delay achieved using the PSIF approach in [11] and the white circular marks indicate the maximum delay achieved using the EIF Type 1 method. The EIF Type 2 method was able to achieve the maximum delay of $D_{max} = 1000$ slots for $M = 2$ and up to a maximum delay of approximately 1400 slots (of $D_{max} = 2000$) for $M = 5$ as compared to approximately 800 and 844 slots, respectively, using the EIF type 1 approach and approximately 525 and 625 slots, respectively, using the PSIF approach in [11].

The relative half-size confidence intervals plotted in Fig.'s 8 and 9 for $M = 2$ and $M = 5$ indicate that the entire curves in Fig. 6 for the EIF Type 1 and Type 2 methods correspond to statistically valid estimates. The improvement factors are plotted as solid lines for the EIF Type 2 method and dashed lined for the EIF Type 1 method in Fig. 7. It can be observed from Fig. 7 that the improvement is inversely proportional to the probability being

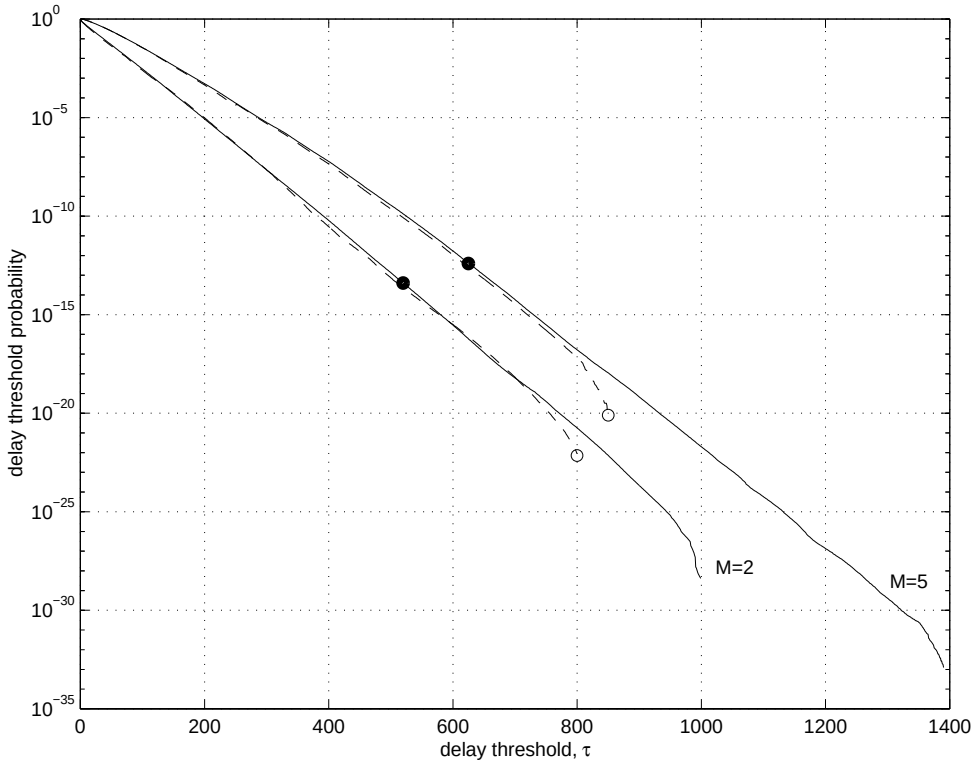

Figure 6: Delay threshold probability results for the first system.

estimated for the EIF methods. The cutoff point (after which $\hat{R}_{net} > 1$) corresponds to probabilities of approximately 10^{-3} to 10^{-4}.

The above results indicate that, as expected, the EIF Type 1 and Type 2 methods represent more efficient indicator functions than the PSIF in [11], particularly as the number of tandem queues M is increased. The EIF Type 1 method starts to lose efficiency at the delay tails and as M is increased. The EIF Type 2 method, as expected, is the most efficient of the three and generates the highest observed delay among the three methods. The comparison of the maximum delays is made by fixing the number of simulation slots for all three methods. Note that this first system is such that the tail probabilities for $M > 2$ are well below 10^{-30} and do not realistically represent common parameters in ATM networks. Furthermore, simulation times become excessively long for $M > 5$, even though the improvement factors are still inversely proportional to the probability being estimated.

To better illustrate the EIF and PSIF methods for larger values of M we consider a

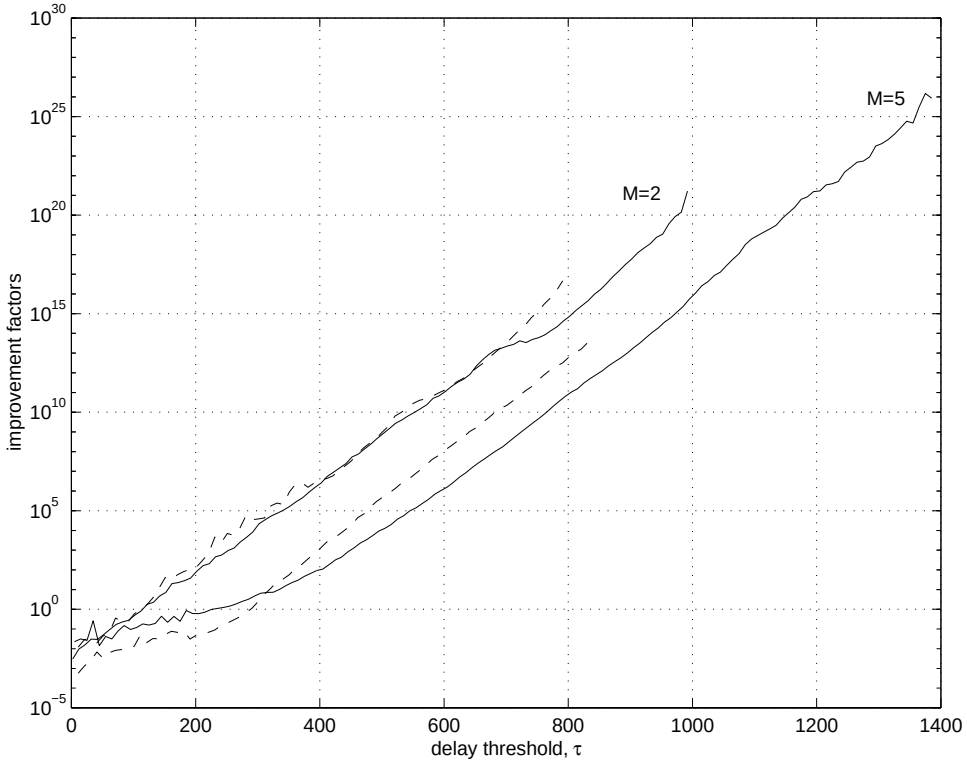

Figure 7: Improvement factors for the first system, EIF methods.

second system. For the second system, $\hat{\lambda}_t = 1$, $\bar{\lambda}_t = 0.1$, $\hat{B}_t = 10$ for the tagged source and $\hat{\lambda} = 1$, $\bar{\lambda} = 0.065$ and $\hat{B} = 10$ for $N = 10$ background sources at each switch. The queue size is $K = 200$ and $C = 1$ for each switch. The results of the simulation runs are plotted in Fig.'s 10 and 11.

In Fig. 10, the delay threshold probabilities are plotted as solid curves for the EIF Type 2 method, dashed curves for the EIF Type 1 method and dotted curves for the PSIF method for $M = 5$, 10 and 15 tandem switches. The number of simulation slots was set to 10×10^7 slots for all configurations and for all methods in order to compare the maximum delay achieved by the EIF and PSIF approaches. The black circular marks and white circular marks indicate the maximum delay achieved by the PSIF method and EIF Type 1 method, respectively, with valid statistical estimates, i.e., normalized half-size confidence interval less than 1.0 (note that, for brevity, the normalized confidence intervals are not shown for this system).

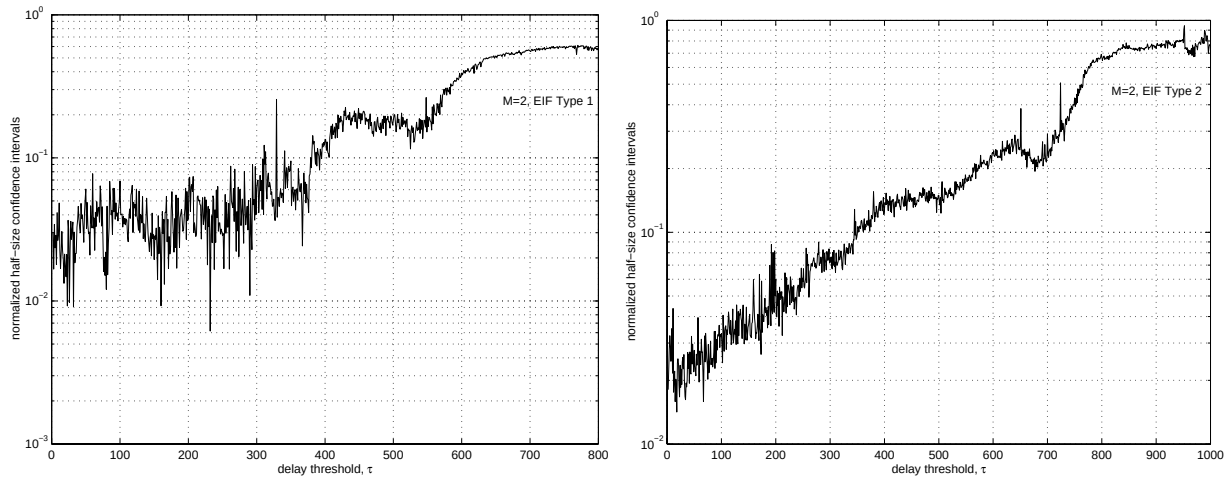

Figure 8: Normalized half-size 95% confidence intervals for $M = 2$, EIF methods.

As seen from Fig. 10, the EIF Type 2 method was able to generate maximum delays of $D_{max} = 1000$ slots for $M = 5$, 1700 slots (of $D_{max} = 2000$ slots) for $M = 10$ and 2200 slots (of $D_{max} = 3000$ slots) for $M = 15$. On the other hand, the maximum delays were 850, 1516 and 1720 slots for $M = 5$, 10 and 15, respectively, for the EIF Type 1 method. The maximum delays were 650, 1035 and 1200 slots for $M = 5$, 10 and 15, respectively, for the PSIF method. As expected, the EIF methods can generate higher maximum delays than the PSIF method for a given number of runs. The efficiency of the EIF methods compared to the PSIF method increases as the number of tandem switches is increased. Also, as expected, the EIF Type 2 method is more efficient than the EIF Type 1 method, especially for higher number of tandem switches.

The improvement factors are plotted as solid lines for the EIF Type 2 method and dashed lined for the EIF Type 1 method in Fig. 11. As seen from Fig. 11, the improvement factors (over standard MC simulation) for the EIF methods are inversely proportional to the probability being estimated. The cutoff point occurs at probabilities of approximately 10^{-3} to 10^{-4} for all cases.

Note that, for all cases, better penetration into the higher delay region can occur if the number of simulation slots is sufficiently increased (although this may cause a decrease in improvement factors). However, the EIF and PSIF methods are compared based on the

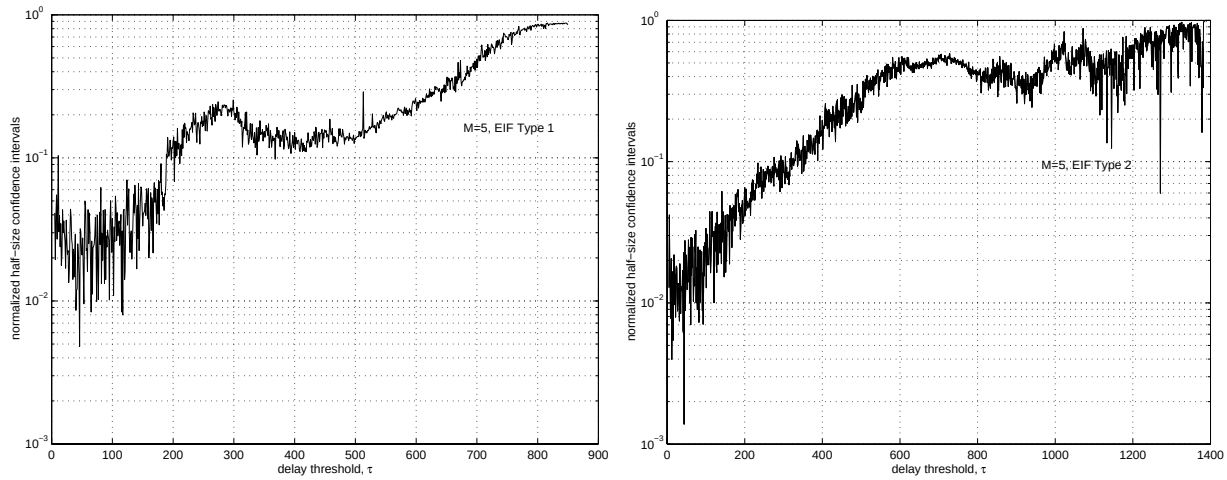

Figure 9: Normalized half-size 95% confidence intervals for $M = 5$, EIF methods.

maximum delay for a given number of simulation slots. As observed from the results in Fig.'s 6 and 10, for a fixed number of simulation slots, the EIF methods penetrate into a significantly higher delay (and lower probability) region compared to the PSIF method. Also, as expected, the EIF Type 2 method is more efficient than the other methods and can accurately estimate much lower probabilities for a given number of simulation slots, especially as the number of tandem switches is increased.

8 Conclusion

In this paper, we considered the problem of estimating rare delay threshold probabilities through tandem ATM switches. We developed two enhanced splitting methods based on a splitting technique which was previously used to estimate rare delay probabilities through a single ATM switch. Based on primary and secondary indicator functions and a single prediction for delay, the previous method produced efficient results for single queues but lost efficiency for tandem queues. We enhanced the concepts of primary and secondary indicator functions to more accurately represent the dynamics of tandem queues.

The first enhanced method does not use prediction of potential delay, but keeps track of delay experienced by cells as they traverse the network. The second enhanced method uses forward prediction, but updates and enhances the prediction of potential delay as cells

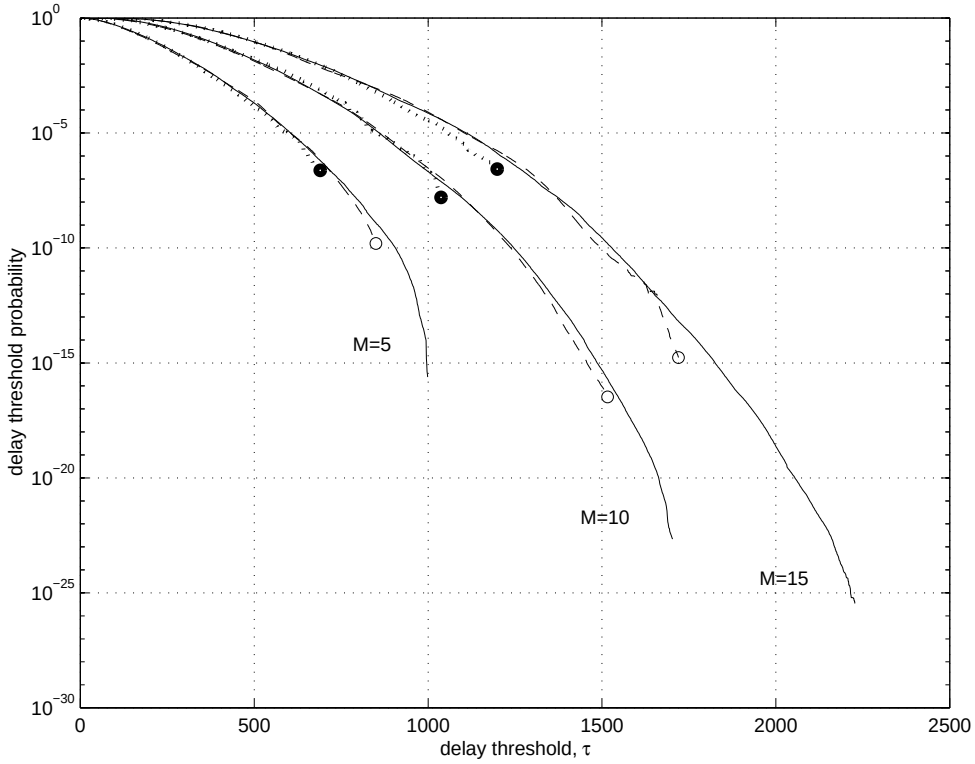

Figure 10: Delay threshold probability results for the second system.

traverse the network. The enhanced methods accurately capture the queueing behavior that leads to excessive cell delay through tandem switching networks. We used the enhanced methods to efficiently estimate rare delay probabilities for tagged traffic traversing tandem ATM switches in the presence of background traffic. The enhanced methods penetrated into a significantly higher delay (and lower probability) region compared to the previous method, and, as expected, the second enhanced method was the most efficient. Speedup over standard Monte Carlo simulation was observed to be inversely proportional to the probability being estimated.

References

[1] P. Heidelberger. Fast Simulation of Rare Events in Queueing and Reliability Models. *ACM Trans. on Modeling and Comp. Sim.*, 5(1):43–85, January 1995.

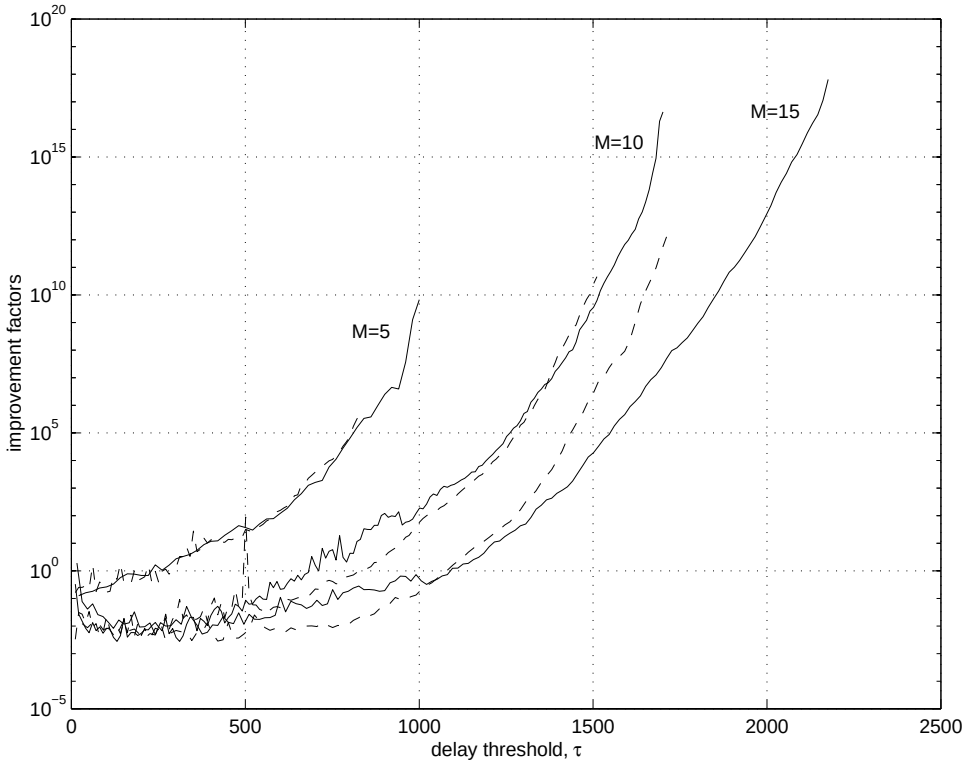

Figure 11: Improvement factors for the second system, EIF methods.

[2] M. Villén-Altamirano and J. Villén-Altamirano. RESTART: A Method for Accelerating Rare Event Simulations. In *Proc. 13th Int. Teletraffic Congress, ITC 13 (Queueing, Performance and Control in ATM)*, pages 71–76, Copenhagen, Denmark, June 1991.

[3] M. Villén-Altamirano, A. Martínez-Marrón, J. Gamo, and F. Fernández-Cuesta. Enhancement of the Accelerated Simulation Method RESTART by Considering Multiple Thresholds. In *Proc. 14th Int. Teletraffic Congress, ITC 14*, volume 1a, pages 797–810, France, 1994.

[4] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic. Splitting for Rare Event Simulation: Analysis of Simple Cases. In *1996 Winter Simulation Conference (San Diego)*, pages 302–308, Piscataway, NJ, December 1996. IEEE Press.

[5] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic. Multilevel Splitting for Estimating Rare Event Probabilities. Technical Report RC 20478, IBM Research

Division, T. J. Watson Research Center, June 1996. To appear in *Operations Research* in 1999.

[6] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic. A Large Deviations Perspective on the Efficiency of Multilevel Splitting. Technical Report RC 20691, IBM Research Division, T. J. Watson Research Center, Jan. 1997.

[7] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic. A Look at Multilevel Splitting. Technical Report RC 20692, IBM Research Division, T. J. Watson Research Center, Jan. 1997.

[8] Z. Haraszti and J. K. Townsend. The Theory of Direct Probability Redistribution and its Application to Rare Event Simulation. In *Proc. IEEE Int. Conf. Commun., ICC '98 (Atlanta)*, pages 1443–1450, Piscataway, NJ, June 1998. IEEE Press.

[9] A. A. Akyamac, Z. Haraszti, and J. K. Townsend. Efficient Rare Event Simulation Using DPR for Multidimensional Parameter Spaces. In *Proc. the IEE Int. Teletraffic Congress, ITC '16*, pages 767–776, Edinburgh, UK, June 1999.

[10] C. Görg and O. Fuß. Simulating Rare Event Details of ATM Delay Time Distributions with RESTART/LRE. in the *Proceedings of the IEE International Teletraffic Congress, ITC16*, June 1999.

[11] Z. Haraszti and J. K. Townsend. Rare Event Simulation of Delay in Packet Switching Networks Using DPR-Based Splitting. in the *Proceedings of the Winter Simulation Conference, WSC'99*, Dec. 1999.