View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by HKU Scholars Hub

The HKU Scholars Hub The University of Hong Kong 香港大學學術庫

Title	The decomposition of a blocking model for connection-oriented networks
Author(s)	Li, CY; Wai, PKA; Li, VOK
Citation	Ieee/Acm Transactions On Networking, 2004, v. 12 n. 3, p. 549-558
Issued Date	2004
URL	http://hdl.handle.net/10722/42979
Rights	Creative Commons: Attribution 3.0 Hong Kong License

The Decomposition of a Blocking Model for Connection-Oriented Networks

C. Y. Li, *Member, IEEE*, P. K. A. Wai, *Senior Member, IEEE*, and Victor O. K. Li, *Fellow, IEEE*

***Abstract*—Two general-purpose decomposition methods to calculate the blocking probabilities of connection-oriented networks are presented. The methods are based on either the *call status* or the *link status* of the networks, and can significantly reduce the required computational times. A heuristic is presented to simplify the application of the proposed decomposition methods on networks with irregular topologies. Numerical examples are given to demonstrate the applications of the proposed methods.**

***Index Terms*—Blocking probability, circuit-switched networks, connection-oriented networks, decomposition methods, Monte Carlo summation.**

I. Introduction

MANY communication networks are connection oriented, e.g., telephone systems, asynchronous transfer mode (ATM) networks, and wavelength-routed optical networks [1]–[4]. Some simple blocking models of circuit-switched networks can be used to provide insight into the performance of these networks. If these blocking models can be solved efficiently, they will be useful in the planning and dimensioning of these networks. For Poisson arrivals and certain routing policies, the exact call blocking probability is given by a simple product form formula [5]. Since the computational complexity of this formula grows exponentially with the network size, it is not suitable for direct modeling of the performance of large networks [6]. Therefore, Monte Carlo summations [7], heuristics of the Monte Carlo summations [8], approximation methods [9]–[12], and parallel simulations [13] have been developed to solve the blocking probabilities of large networks. These methods in general require less computational time but the results are not exact. The computational time can still be very long if high accuracy is required in the Monte Carlo methods and simulations. Furthermore, the accuracy of some methods such as approximations depends on the traffic conditions. In many occasions, exact results are required to validate the approximation methods and simulations.

Manuscript received September 14, 2000; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor J. Daigle. This work was supported in part by the Areas of Excellence Scheme established under the University Grants Committee of the Hong Kong Special Administrative Region, China, (Project AoE/E-01/99) and in part by a grant from The Hong Kong Polytechnic University (Project Number A-PD69).

C. Y. Li and P. K. A. Wai are with the Photonics Research Center and Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong SAR, China (e-mail: enli@polyu.edu.hk; enwai@polyu.edu.hk).

V. O. K. Li is with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong SAR, China (e-mail: vli@eee.hku.hk).

Digital Object Identifier 10.1109/TNET.2004.828937

When exact blocking probabilities are required, decomposition can be the most effective means to reduce the computational cost [14]–[19]. Several orders of magnitude reduction in computational time can be obtained. In decomposition methods, the computation of the blocking probabilities is broken down to the subnetwork level. Besides a reduction in the size of the subnetwork, we can choose the most suitable method to solve the blocking probability in each individual subnetwork. In networks with regular topology, e.g., tree networks, decompositions have been used to develop efficient algorithms to compute the blocking probabilities [14]–[17]. General-purpose topology-based decomposition methods have been proposed to reduce the computational cost of exact blocking probability computation in product form models [18]. Numerical inversion of generating functions, normally used to compute the blocking probability exactly, have been used together with decomposition and truncation to speed up the computation [19]. The decomposition methods mentioned so far are based on the network topology. For network with arbitrary topology, they do not always provide the optimal decomposition. In this paper, we develop two general-purpose decomposition methods based on the concept of *noninterference* in network traffic [20]. We found that the computational savings with the proposed methods is larger than that of the methods reported in the literature to date.

In Section II, we give a brief review of the general blocking models for circuit-switched networks. We introduce the concept of *noninterference* between call classes in Section III and propose a decomposition method based on the call status. In Section IV, we propose another decomposition strategy that makes use of the link status. Both strategies do not rely on network topology and can significantly reduce the computations on the blocking probability in circuit-switched networks, but under very different traffic conditions. An obstacle of using decomposition methods is to find a suitable partition of the call class set. We propose a simple heuristic in Section V to simplify this task. Numerical examples are given in Section VI to demonstrate the applications of the proposed methods.

II. Product Form Networks

The network is defined as a set of p links labeled from ℓ_1 to ℓ_p. Link ℓ_i has a capacity of N_i channels. Let $\mathbf{L} = \{\ell_1, \ldots, \ell_p\}$ be the set of links and $\mathbf{N} = (N_1, \ldots, N_p)$ be the corresponding link capacity vector. We assume that the routing paths of calls are fixed. Calls are classified according to their requirement of links and channel capacity. The r classes of calls in the network are labeled from c_1 to c_r. We define $\mathbf{R} = \{c_1, \ldots, c_r\}$ as the set of call classes. Class c_j calls arrive at the system according to a

1063-6692/04$20.00 © 2004 IEEE

Poisson process with mean arrival rate λ_j. The holding time is an independent exponential random variable, with mean $1/\mu_j$. Hence, the offered load is $\rho_j = \lambda_j/\mu_j$. A class c_j call requests a nonnegative number a_{ij} of channels on link ℓ_i. The demand matrix $\mathbf{A}$ is defined as $(a_{ij} : \ell_i \in \mathbf{L}, c_j \in \mathbf{R})$. If any of the links that a new call requests has insufficient free channels, the call is blocked. The call blocking probabilities can be solved based on either call status or link status probabilities.

A. Blocking Computation With Call Status Probability

Let n_j be the number of class c_j calls in progress, and $\mathbf{n} = (n_1, \ldots, n_r)$ be the call status vector. We define $\Gamma(\mathbf{N}) = \{\mathbf{n} : \mathbf{0} \leq \mathbf{An} \leq \mathbf{N}\}$ as the set of valid states for $\mathbf{n}$ under the link capacity vector $\mathbf{N}$. The stationary probability distribution $\pi(\mathbf{n})$ of the call status vector $\mathbf{n}$ is given by

$$\pi(\mathbf{n}) = \frac{1}{G(\mathbf{N})} \prod_{j=1}^{r} \frac{\rho_j^{n_j}}{n_j!}, \quad \mathbf{n} \in \Gamma(\mathbf{N}) \tag{1}$$

where

$$G(\mathbf{N}) = \sum_{\mathbf{n} \in \Gamma(\mathbf{N})} \prod_{j=1}^{r} \frac{\rho_j^{n_j}}{n_j!} \tag{2}$$

is the normalization constant [21, Sec. 1.6]. The probability of a class c_j call being blocked, B_j, is the ratio of the number of rejected class c_j calls to the number of offered class c_j calls. We have

$$B_j = 1 - \frac{1}{\rho_j} \sum_{\mathbf{n} \in \Gamma(\mathbf{N})} \pi(\mathbf{n}) \times n_j. \tag{3}$$

From the expression of $\pi(\mathbf{n})$ in (1) and the definition of $G(\mathbf{N})$ in (2), we have

$$B_j = 1 - \frac{G(\mathbf{N} - \mathbf{AI}_j)}{G(\mathbf{N})} \tag{4}$$

where $\mathbf{I}_j \in \Gamma(\mathbf{N})$ is a unit vector which represents only one class c_j call in progress.

B. Blocking Computation With Link Status Probability

We can calculate the blocking probabilities of the above network using link status probabilities instead of call status probabilities. Let m_i be the number of channels used on link ℓ_i and $\mathbf{m} = (m_1, \ldots, m_p)$ be the link status vector. Given the valid call status set $\Gamma(\mathbf{N})$, the set of valid states for $\mathbf{m}$ is defined as $\mathbf{M}(\mathbf{N}) = \{\mathbf{m} : \mathbf{An} = \mathbf{m}, \forall\, \mathbf{n} \in \Gamma(\mathbf{N})\}$. We define $Q(\mathbf{m})$ as the unnormalized stationary probability distribution for the link status being $\mathbf{m}$. From the definition of $\mathbf{M}(\mathbf{N})$, we have $Q(\mathbf{0}) = G(\mathbf{N}) \times \pi(\mathbf{0}) = 1$. The distribution $Q(\mathbf{m})$ can be solved from the call status probability as

$$Q(\mathbf{m}) = G(\mathbf{N}) \times \sum_{\mathbf{An}=\mathbf{m}} \pi(\mathbf{n}).$$

Dziong and Roberts proposed an algorithm [24] to solve $Q(\mathbf{m})$ without involving the call status probability as

$$Q(\mathbf{m} + \mathbf{1}_i) = \sum_{j=1}^{r} \frac{\rho_j a_{ij}}{m_i + 1} Q(\mathbf{m} + \mathbf{1}_i - \mathbf{AI}_j) \tag{5}$$

for all $\mathbf{m} + \mathbf{1}_i \in \mathbf{M}(\mathbf{N})$ and $Q(\mathbf{m}) = 0$ otherwise, where $\mathbf{1}_i$ is the unit vector of length p with a 1 in position i and 0 elsewhere. As the normalization constant of $Q(\mathbf{m})$ is also $G(\mathbf{N})$, we can solve $G(\mathbf{N})$ using

$$G(\mathbf{N}) = \sum_{\mathbf{m} \in \mathbf{M}(\mathbf{N})} Q(\mathbf{m}). \tag{6}$$

Hence, the blocking probability of a class c_j call can be solved from (4)–(6).

In the next section, we will show that the *noninterference* of the call classes can be used to reduce the computational cost.

III. Call-Status-Based Decomposition

Two or more subsets of call classes $\mathbf{R}_1, \ldots, \mathbf{R}_K \subset \mathbf{R}, K \geq 2$, are said to be *noninterfering* with each other if the setup of a call in a class belonging to any subset $\mathbf{R}_x, 1 \leq x \leq K$ will not cause blocking of the calls in classes belonging to any other subset $\mathbf{R}_y, y \neq x$, given that the status of calls in the classes belonging to $\mathbf{R}_0 = \mathbf{R} - \bigcup_{k=1}^{K} \mathbf{R}_k$ have been fixed [20]. Some properties of the noninterfering call classes are given in Appendix A. In [20], we have proved that if $\mathbf{R}$ can be partitioned into $\mathbf{R}_0 + \cdots + \mathbf{R}_K$ and the call classes belonging to different sets $\mathbf{R}_1, \ldots, \mathbf{R}_K$ are noninterfering with each other, $G(\mathbf{N})$ can be written as

$$G(\mathbf{N}) = \sum_{\mathbf{n} \in \Gamma_0(\mathbf{N})} \prod_{k=1}^{K} G_k(\mathbf{N} - \mathbf{An}) \prod_{c_j \in \mathbf{R}_0} \frac{\rho_j^{n_j}}{n_j!} \tag{7}$$

where $\Gamma_k(\mathbf{N}) = \{\mathbf{n} \in \Gamma(\mathbf{N}) : n_j = 0\ \forall\, c_j \notin \mathbf{R}_k\}$ is the reduced set of valid states in which arrivals of calls not belonging to $\mathbf{R}_k$ are suspended, and

$$G_k(\mathbf{N}) = \sum_{\mathbf{n} \in \Gamma_k(\mathbf{N})} \prod_{c_j \in \mathbf{R}_k} \frac{\rho_j^{n_j}}{n_j!}. \tag{8}$$

Conway *et al.* [18] partition a network into subnetworks and derive equations similar to those of (7) and (8). Since the partition is based on the network topology, it does not always give the maximum computational savings. We note that Conway *et al.* use call status probabilities to carry out the computations related to the inter-subnetwork traffic. In the following, we show that one can perform the computation using link status probabilities even if the decomposition is based on the status of the call classes. In some situations, this approach will give larger computational savings.

A. Computations With Link Status Probability Only

We define $\mathcal{P}_k(\cdot)$ as a projection operator on $\mathbf{n}$ such that only the components representing the status of the calls in the classes belonging to $\mathbf{R}_k$ are retained. In (7), $\prod_{k=1}^{K} G_k(\mathbf{N} - \mathbf{An}) \prod_{c_j \in \mathbf{R}_0} (\rho_j^{n_j}/n_j!)$ is equal to the unnormalized probability of the calls in the classes belonging to $\mathbf{R}_0$ having status of $\mathcal{P}_0(\mathbf{n})$. We define $\mathbf{M}_0(\mathbf{N})$ as the reduced set of valid links status where the arrivals of the calls in the classes not belonging to $\mathbf{R}_0$ are suspended. Given that the blocking probability computation

of a network can be decomposed in the form of (7), we can transform it into

$$G(\mathbf{N}) = \sum_{\mathbf{m}\in\mathbf{M}_0(\mathbf{N})} \prod_{k=1}^{K} G_k(\mathbf{N}-\mathbf{m}) Q_0(\mathbf{m}) \tag{9}$$

where $Q_0(\mathbf{m})$ is the unnormalized probability of the links with status $\mathbf{m}$ under the condition that only the arrivals of the calls in the classes belonging to $\mathbf{R}_0$ are permitted. Recall that the link status vector $\mathbf{m}$ represents the capacities used by the calls in the classes belonging to $\mathbf{R}_0$; thus $\prod_{k=1}^{K} G_k(\mathbf{N}-\mathbf{m})Q_0(\mathbf{m})$ is the unconditional unnormalized probability of the calls in the classes belonging to $\mathbf{R}_0$ reserving the link capacity represented by the status vector $\mathbf{m}$. We can compute $Q_0(\mathbf{m})$ using (5) for $\mathbf{m} \in \mathbf{M}_0(\mathbf{n})$. If $|\mathbf{M}_0(\mathbf{N}))| \ll |\Gamma_0(\mathbf{N})|$, then (9) can give us more computational savings than that of (7). Such an example is given in Section VI-B.

Apart from using (9) $(r+1)$ times to compute the blocking probabilities with (4), we may reduce the computational complexity further by using the special form of B_j. For c_j belonging to one of the noninterfering subset $\mathbf{R}_y, y \neq 0$, the blocking probability B_j has the expression

$$B_j = 1 - \sum_{\mathbf{m}\in\mathbf{M}_0(\mathbf{N}-\mathbf{AI}_j)} \frac{G_y(\mathbf{N}-\mathbf{m}-\mathbf{AI}_j)}{G_y(\mathbf{N}-\mathbf{m})} \hat{P}(\mathbf{m}) \tag{10}$$

where $\hat{P}(\mathbf{m})$ is the probability of the calls in the classes belonging to $\mathbf{R}_0$ reserving the link capacity represented by status vector $\mathbf{m}$, and is equal to $G(\mathbf{N})^{-1} \times \prod_{k=1}^{K} G_k(\mathbf{N}-\mathbf{m})Q_0(\mathbf{m})$. When the capacity reserved by the calls in $\mathbf{R}_0$ is $\mathbf{m}$, the conditional probability of a new class c_j call being accepted by the network is $G_y(\mathbf{N}-\mathbf{m}-\mathbf{AI}_j)/G_y(\mathbf{N}-\mathbf{m})$ because the status of calls belonging to other subsets $\mathbf{R}_x, x \neq y \neq 0$ has no effect on the blocking of the class c_j calls [20]. $\hat{P}(\mathbf{m})$ is the probability that the calls in the classes belonging to $\mathbf{R}_0$ have status $\mathcal{P}_0(\mathbf{n})$. The blocking probability of a class c_j call is equal to one minus the sum of the probabilities of a class c_j call being accepted under all conditions of $\mathbf{m} \in \mathbf{M}_0(\mathbf{N})$. The resource requirement of (9) and (10) is given in Appendix B. Section VI-B shows an example where link status probability computation has lower complexity than traditional call-status-based decomposition.

IV. Link-Status-Based Decomposition

From (9) and (10), we can compute the blocking probabilities using link status probabilities while the decomposition is based on call status. In some network topologies, e.g., the tree networks, there are efficient algorithms that are based on link status probability for solving the blocking probability [14]–[17]. Call-status-based decompositions cannot provide similar computational savings even if (9) and (10) are used. Thus, one should be able to develop link-status-based decomposition methods which can provide significant computational savings for some of the network topologies that call status decomposition methods fail to provide savings for. We define $\mathbf{L}_k^+$ as the set of links that are used by calls in the classes belonging to $\mathbf{R}_k$. Even if the call class sets $\mathbf{R}_0, \mathbf{R}_1, \ldots, \mathbf{R}_K$ meet the requirement of noninterference in Section III and we fix the status of links belonging to $\mathbf{L}_0^+$, the links belonging to the different link sets (subnetworks) $\mathbf{L}_k^+ - \mathbf{L}_0^+, k = 1, \ldots, K$ will not necessarily be independent of each other. However, if we could increase the capacity of links in $\mathbf{L}_0^+$ beyond certain limits, the links belonging to different links sets $\mathbf{L}_k^+ - \mathbf{L}_0^+, k = 1, \ldots, K$ would then become independent of each other. It is because the calls in the classes belonging to different subnetworks share common links belonging to $\mathbf{L}_0^+$. Giving sufficient capacities to these common links will render the call classes noninterfering (Appendix A). Using this observation, we compute the link status probabilities of the subnetworks individually as if there is no other traffic in the network. For example, we transform a network into three subnetworks as shown in Fig. 1. After summing up the probabilities of possible combinations of subnetwork traffic that generate the specified status of the links belonging to $\mathbf{L}_0^+$, we obtain the status probability of the links in $\mathbf{L}_0^+$. We can then solve the $G(\mathbf{N})$ and B_j accordingly. We summarize the procedure as follows.

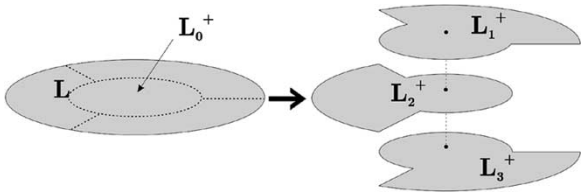

Fig. 1. Partition of network into three subnetworks based on the link status. $\mathbf{L}_0^+$ is the set of common links shared by the three subnetworks defined by link set $\mathbf{L}_k^+, k = 1, 2, 3$.

Assume that a network has a link set $\mathbf{L}_0^+ \subset \mathbf{L}$ and removing the capacity limits of the links belonging to $\mathbf{L}_0^+$ will partition the remaining links into K independent subnetworks of link sets $\mathbf{L}_k$'s where $k = 1, \ldots, K$.

1) First construct $\mathbf{R}_k$, the set of the classes of calls using any link belonging to the link set $\mathbf{L}_k$ for $k = 1, \ldots, K$.
2) Then construct the set $\mathbf{R}_0 = \mathbf{R} - \bigcup_{k=1}^{K} \mathbf{R}_k$.
3) Then construct $\mathbf{L}_k^+$, the sets of the links that are required by the calls in the classes belonging to $\mathbf{R}_k$ for $k = 1, \ldots, K$.
4) Solve for $Q_k^+(\mathbf{m})$, the unnormalized probability of the calls in the classes belonging to $\mathbf{R}_k$ reserving the link capacity represented by status vector $\mathbf{m}$ as if there is no other traffic in the networks, where $k = 0, \ldots, K$.
5) After determining all the $Q_k^+(\mathbf{m})$, solve for $\tilde{Q}(\boldsymbol{\xi})$, the unnormalized probability of the links belonging to $\mathbf{L}_0^+$ that have status $\boldsymbol{\xi}$

$$\tilde{Q}(\boldsymbol{\xi}) = \sum_{\boldsymbol{\zeta}_0+\cdots+\boldsymbol{\zeta}_K=\boldsymbol{\xi}} \prod_{k=0}^{K} \left[\sum_{\mathcal{P}_0(\mathbf{m})=\boldsymbol{\zeta}_k} Q_k^+(\mathbf{m}) \right] \tag{11}$$

where $\mathcal{P}_0(\cdot)$ is a projection operator on $\mathbf{m}$ such that only the components representing the status of links belonging to $\mathbf{L}_0^+$ are retained.

6) The normalization constant of the call status probability, $G(\mathbf{N})$, is equal to that of $Q(\mathbf{m})$ in (6), and that of $\tilde{Q}(\boldsymbol{\xi})$ in (11). Hence

$$G(\mathbf{N}) = \sum_{\boldsymbol{\xi}\in\mathcal{P}_0(\mathbf{M}(\mathbf{N}))} \tilde{Q}(\boldsymbol{\xi}). \tag{12}$$

To compute $Q_k^+(\mathbf{m})$, we can use (5) in Section II-B recursively on the links and call classes belonging to $\mathbf{L}_k^+$ and $\mathbf{R}_k$, respectively. To solve the blocking probability of a class j call, we use $B_j = 1 - G(\mathbf{N} - \mathbf{A}\mathbf{I}_j)/G(\mathbf{N})$, where

$$G(\mathbf{N} - \mathbf{A}\mathbf{I}_j) = \sum_{\boldsymbol{\xi} \in \mathcal{P}_0(\mathbf{M}(\mathbf{N}-\mathbf{A}\mathbf{I}_j))} \hat{Q}(\boldsymbol{\xi}), \quad j \in \mathbf{R} \tag{13}$$

and

$$\hat{Q}(\boldsymbol{\xi}) = \sum_{\boldsymbol{\zeta}_0 + \cdots + \boldsymbol{\zeta}_K = \boldsymbol{\xi}} \prod_{k=0}^{K} \left[\sum_{\substack{\mathcal{P}_0(\mathbf{m}) = \boldsymbol{\zeta}_k, \\ \mathbf{m} \in \mathbf{M}(\mathbf{N}-\mathbf{A}\mathbf{I}_j)}} Q_k^+(\mathbf{m}) \right]. \tag{14}$$

The analysis of the resource requirement for using (11)–(14) is given in Appendix B.

V. Heuristic for Partitioning the Call Classes Set

An obstacle of using decomposition methods to compute the blocking probabilities is to find a suitable partition of $\mathbf{R}$ with acceptable costs. Exhaustive search is the only means that guarantee the optimum partition that gives the largest computational savings. It is, however, not practical because the cost of the search grows rapidly with the network size. When the network topology is not regular, it is difficult to obtain a suitable partition by observation. We observe that instead of finding the optimal partition of $\mathbf{R}$ for a given decomposition method, it will be easier just to determine a partition of $\mathbf{R}$ such that the computational cost is less than a chosen value. The choice of this value depends on the available computational resources, and must be reasonably large such that at least one partition of $\mathbf{R}$ exists. The estimates of the resource requirement of the proposed decomposition methods given in Appendix B serves as a guide in choosing the value. Given a demand matrix $\mathbf{A}$ and a link capacity vector $\mathbf{N}$, we propose the following heuristic to generate a partition of $\mathbf{R}$ without relying on observation of the network topology.

Heuristic 1: Given a demand matrix $\mathbf{A}$ and a link capacity vector $\mathbf{N}$:

1) Set an acceptable computational cost in the determination of the blocking probabilities.
2) Randomly re-label the links by generating random permutations of the rows of the demand matrix $\mathbf{A}$. Then interchange the columns of the re-labeled matrix to the form shown in Fig. 2. The grey area may contain nonzero elements, while the white area is all zeros. Repeat the procedure to generate a number of transformed matrices, and select the matrix with the minimum grey area for Step 3.
3) Choosing $K, |\mathbf{R}_k|$ and/or $|\mathbf{L}_k^+|$'s, we determine the partition of $\mathbf{R} = \mathbf{R}_0 + \cdots + \mathbf{R}_K$ by interchanging columns in the transformed demand matrix of Fig. 2 to a matrix in a form similar to Fig. 3(a) or (b) depending on which decomposition method is used. Fig. 3(a) and (b) shows a partition of $\mathbf{R}$ for a network using call (link) status-based decomposition with $K = 3$. The call classes sets $\mathbf{R}_k$ (link sets $\mathbf{L}_k$), $k = 1, 2, 3$ are given by the dashed rectangles.
4) Since the common links of two noninterfering call classes can be ignored in all subnetwork blocking probability computations, use Condition 1 in Appendix A to identify these links in the partition. Also use Condition 1 to determine if the further partitioning of a subset of $\mathbf{R}$ is possible.
5) Use the estimate in Appendix B to determine whether the resources required in using a specific decomposition method with the partition $\mathbf{R}$ exceeds the requirement in Step 1. Otherwise, repeat the procedure from Step 3 with a new set of K and $|\mathbf{R}_k|$. If the possible sets of K and $|\mathbf{R}_k|$ are exhausted, stop the procedure and report the recorded partition of $\mathbf{R}$ that has the lowest computational requirements.

Fig. 2. Grouping the call classes in the demand matrix according to similar pattern of link utilization.

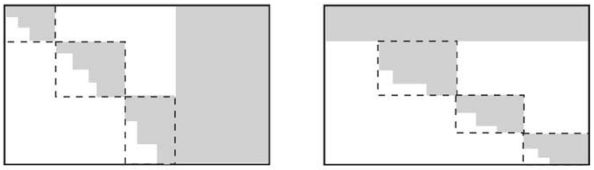

Fig. 3. Partition of the set of calls $\mathbf{R}$ for a network using (a) call-status-based and (b) link-status-based decomposition with $K = 3$. The set of (a) call classes $\mathbf{R}_k$ and (b) links $\mathbf{L}_k, k = 1, 2, 3$ are given by the dashed rectangles.

Step 3 transforms the demand matrix such that the required partition of $\mathbf{R}$ or $\mathbf{L}$ can be determined directly. The dashed rectangles in Fig. 3(a) show the sets of call classes that do not share any common links in the network. One can verify that the call classes in these sets are independent of each other if one fixes the status of the call classes identified by the rightmost large rectangle (discussion of *noninterference* is provided in Appendix A). Hence, the columns in the rightmost large rectangle represent the call classes belonging to $\mathbf{R}_0$, and those in the dashed rectangles are for the call classes belonging to sets $\mathbf{R}_1, \ldots, \mathbf{R}_K$ in Section III, respectively. Similarly, the dashed rectangles in Fig. 3(b) show the sets of links that do not have any common calls. Thus, they form a partition of $\mathbf{L}$ in Section IV.

The proposed heuristic is useful especially when the network topology is irregular. If all the links in a network have similar capacities, we set all $|\mathbf{R}_k|$ in Fig. 3(a), or $|\mathbf{L}_k^+|$ in Fig. 3(b), to be the same and choose a K that minimizes $|\mathbf{R}_0|$ and/or $|\mathbf{L}_0^+|$ of the transformed matrices in Fig. 3. In general, we estimate the resource requirement using the equations in Appendix B. Step 2 uses a random approach instead of exhaustive search to group the call classes according to the similarity in link utilization. Adjacent call classes in the transformed matrix at Step 3,

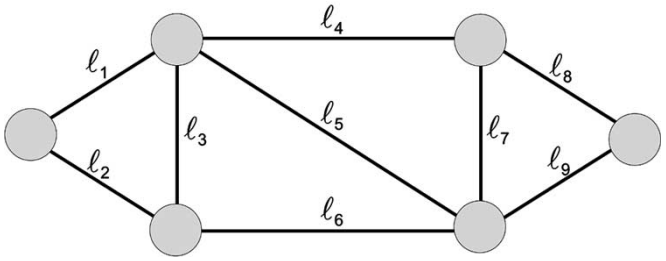

Fig. 4. Network with six nodes and nine links. The link capacity vector is $\mathbf{N} = (C, 2C, C, C, 2C, C, C, 2C, C)$, where $C \geq 2$. Fourteen paths are defined. Arriving calls can request either one or two channels. There are 28 call classes, c_1–c_{28}.

therefore, have larger probability of using common links. This reduces the number of iterations between Steps 3–5. An estimate of the required number of randomly generated transformed matrices is the maximum of the square of the number of classes r and the square of the number of links p. If the final partition of $\mathbf{R}$ obtained cannot satisfy the requirement, we may increase the number of randomly generated transformed matrices and repeat the procedure from Step 2. From experience, however, the improvement may be small. The estimated computational savings by the heuristic when compared to the trial-and-error approach is given in Appendix C.

VI. Examples

In this section, we will present three examples to demonstrate the use of the proposed decomposition methods and heuristic to compute the blocking probabilities. In the first example, we use a simple network to illustrate the basic features of the proposed decomposition methods. Conway *et al.* proposed a topology-based general-purpose decomposition method similar to that of (7) and (8). They showed that large computational savings can be obtained [18]. In the second example, we compared the performance of our proposed methods with that of Conway *et al.* using the same network topology and routing paths as that of [18] but with twice the number of call classes. We showed that our proposed methods provided significant savings over that of [18]. In the last example, we used the heuristic proposed in Section V to solve the call blocking probabilities of a network with irregular topologies. All computations were carried out using Fortran programs on a 600-MHz Alpha CPU workstation. In the first two examples, the system loadings were set at random because we are only interested in the computational times.

A. Example 1

Fig. 4 shows a network with six nodes and nine links. The link capacity vector is $\mathbf{N} = (C, 2C, C, C, 2C, C, C, 2C, C)$, where $C \geq 2$. Fourteen paths are defined. They are $a : \{\ell_1, \ell_2\}, b : \{\ell_1, \ell_4\}, c : \{\ell_1, \ell_2\, \ell_4\}$, $d : \{\ell_2, \ell_3\}, e : \{\ell_7, \ell_8\}, f : \{\ell_6, \ell_8, \ell_9\}, g : \{\ell_6, \ell_9\}$, $h : \{\ell_8, \ell_9\}, i : \{\ell_1, \ell_5\}, j : \{\ell_3, \ell_5\}$, $k : \{\ell_4, \ell_5\}, l : \{\ell_5, \ell_6\}, m : \{\ell_5, \ell_7\}$, and $n : \{\ell_5, \ell_9\}$. Arriving calls can request either one or two channels. There are 28 call classes in which c_1–c_{14} (c_{15}–c_{28}) request one channel (two channels) on the paths a–n, respectively.

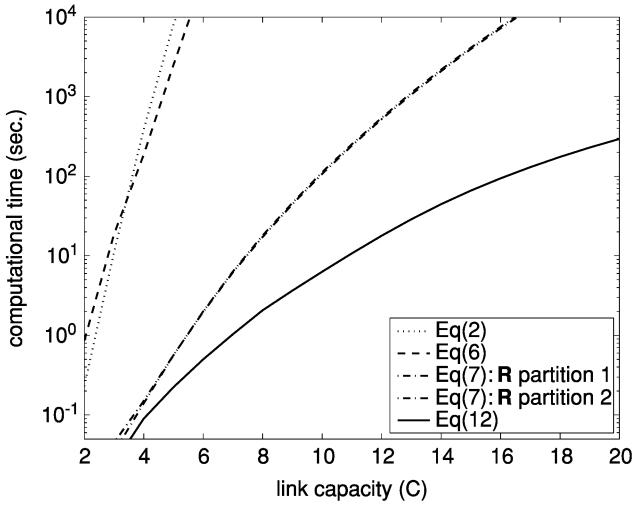

Fig. 5. Computational times required to calculate the blocking probabilities of the network shown in Fig. 4. The dotted and dashed lines represent direct computation of the blocking probabilities using call status probability (2) and link status probability (6), respectively, without the use of decomposition methods. The black dash-dotted line and grey dash-dotted line represent computations using call-status-based decomposition method (7) using two different partitions of $\mathbf{R}$. The solid line represents computations using link-status-based decomposition methods (12).

In Fig. 5, the dotted and dashed lines give the time required to compute the blocking probabilities using call status probability (2) and link status probability (6), respectively. The computational time grows rapidly with C. In 10^4 seconds, we can only compute the blocking probabilities of the network up to $C = 5$.

To apply the call-status-based decomposition method, we observe that removal of link ℓ_5 will separate the network into two subnetworks with link sets $\mathbf{L}_1^+ = \{\ell_1, \ldots, \ell_4\}$ and $\mathbf{L}_2^+ = \{\ell_6, \ldots, \ell_9\}$. We therefore partition $\mathbf{R}$ into $\mathbf{R}_0 + \mathbf{R}_1 + \mathbf{R}_2$, where $\mathbf{R}_0 = \{c_9, \ldots, c_{14}, c_{23}, \ldots, c_{28}\}, \mathbf{R}_1 = \{c_1, \ldots, c_4, c_{15}, \ldots, c_{18}\}$, and $\mathbf{R}_2 = \{c_5, \ldots, c_8, c_{19}, \ldots, c_{22}\}$. The black dash-dotted line in Fig. 5 gives the required computational time for using this partition of $\mathbf{R}$ and (7). The computational savings is very large when compared with that of (2) and (6).

We observe that c_4 and c_{18} are noninterfering with the rest of the call classes in $\mathbf{R}_1$, similarly for c_8 and c_{22} in $\mathbf{R}_2$. We can therefore partition $\mathbf{R}$ as $\tilde{\mathbf{R}}_0 + \cdots + \tilde{\mathbf{R}}_4$, where $\tilde{\mathbf{R}}_0 = \mathbf{R}_0, \tilde{\mathbf{R}}_1 = \{c_4, c_{18}\}, \tilde{\mathbf{R}}_2 = \{c_8, c_{22}\}, \tilde{\mathbf{R}}_3 = \mathbf{R}_1 - \tilde{\mathbf{R}}_1$, and $\tilde{\mathbf{R}}_4 = \mathbf{R}_2 - \tilde{\mathbf{R}}_2$. The new partition of $\mathbf{R}$ reduces the part of the computations arising from the subnetwork traffic but has no effects on that of the inter-subnetwork traffic. The required computational time using the new $\mathbf{R}$ partition is plotted as the grey dash-dotted line in Fig. 5. The extra savings due to the new partition is negligible in this case because most of the computations are due to the inter-subnetwork traffic. Equation (9) is not useful here because the number of links used by the calls in the classes belonging to $\mathbf{R}_0$ is large, i.e., seven links.

We then apply the procedure described in Section IV. We observe that the links in the sets $\{\ell_1, \ldots, \ell_4\}$ and $\{\ell_6, \ldots, \ell_9\}$ will be independent of each other if the capacity limit of link ℓ_5 is removed. Hence, we decompose the network into three subnetworks with link sets $\mathbf{L}_0^+ = \{\ell_5\}, \mathbf{L}_1^+ = \{\ell_1, \ldots, \ell_5\}$, and $\mathbf{L}_2^+ = \{\ell_5, \ldots, \ell_9\}$. In addition, we have $\mathbf{R}_0 = \phi, \mathbf{R}_1 =$

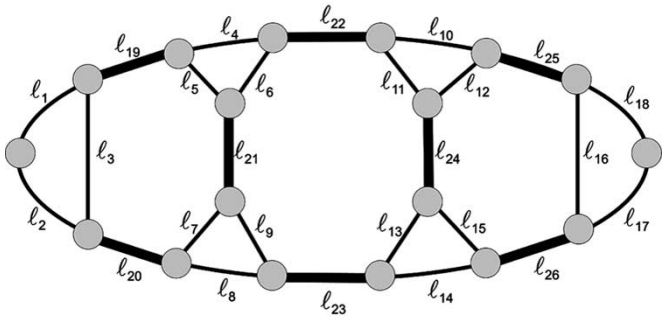

Fig. 6. Topology of the network used by Conway *et al.* [18]. All the links are set to have the same capacity of five channels. Thirty routing paths are defined. The set of interconnecting links is $\{\ell_{19}, \ldots, \ell_{26}\}$.

$\{c_1, \ldots, c_4, c_9, c_{10}, c_{11}, c_{15}, \ldots, c_{18}, c_{23}, c_{24}, c_{25}\}$, and $\mathbf{R}_2 = \{c_5, \ldots, c_8, c_{12}, c_{13}, c_{14}, c_{19}, \ldots, c_{22}, c_{26}, c_{27}, c_{28}\}$. We compute the blocking probabilities using (11)–(14), and plot the required times as the solid line in Fig. 5. The savings is significant because the computations related to the interconnecting link ℓ_5 are much smaller than that of the inter-subnetwork traffic, i.e., $\{c_9, \ldots, c_{14}, c_{23}, \ldots, c_{28}\}$, in the call-status-based decompositions.

B. Example 2

Fig. 6 shows the network used by Conway *et al.* in [18]. The capacity of each link is five channels. In [18], 30 call classes are defined. The calls in classes $c_{4x+1}, c_{4x+2}, c_{4x+3}$, and c_{4x+4} request a channel on links $\{\ell_{3x+1}\}, \{\ell_{3x+2}\}, \{\ell_{3x+3}\}$, and $\{\ell_{3x+1}, \ell_{3x+3}\}$, respectively, where $x = 0, \ldots, 5$. The calls in the classes $c_{25}, c_{26}, c_{27}, c_{28}, c_{29}$, and c_{30} request a channel on links $\{\ell_1, \ell_4, \ell_{10}, \ell_{19}, \ell_{22}\}, \{\ell_1, \ell_5, \ell_{19}, \ell_{21}\}$, $\{\ell_3, \ell_7, \ell_{20}\}, \{\ell_8, \ell_{11}, \ell_{13}, \ell_{23}, \ell_{24}\}, \{\ell_{14}, \ell_{17}, \ell_{23}, \ell_{26}\}$, and $\{\ell_{12}, \ell_{16}, \ell_{25}, \ell_{26}\}$, respectively. To demonstrate the capability of the proposed decomposition methods, we add another 30 call classes and determine the computational time to determine the blocking probabilities when C varies from 5 to 15. The calls in the class c_{30+y} use the same set of links as that of $c_y, 1 \leq y \leq 30$, but request two channels instead of one.

In [18], $\mathbf{L}_0 = \{\ell_{19}, \ldots, \ell_{26}\}$ is defined as the set of interconnecting links. Conway *et al.* partitioned the network into six subnetworks each having links set $\mathbf{L}_{x+1} = \{\ell_{3x+1}, \ell_{3x+2}, \ell_{3x+3}\}$ where $x = 0, \ldots, 5$. Using this topology information, we partitioned $\mathbf{R}$ into $\mathbf{R}_0 + \cdots + \mathbf{R}_6$ where $\mathbf{R}_0 = \{c_{25}, \ldots, c_{30}, c_{55} \ldots, c_{60}\}$ is the set of inter-subnetwork traffic. $\mathbf{R}_{x+1} = \{c_{4x+1}, \ldots, c_{4x+4}, c_{4x+31}, \ldots, c_{4x+34}\}$ are the set of subnetwork traffic, where $x = 0, \ldots, 5$. This partition of $\mathbf{R}$ and (7) are then used to calculate the blocking probabilities for different values of C. The required computational time is plotted as the dashed line in Fig. 7. It takes only about 7 s to compute the blocking probabilities at $C = 5$, but 13 397 s when $C = 12$.

We check the partition with Condition 1 in Appendix A. No call classes in $\mathbf{R}_0$ can be further classified into subnetwork traffic. To reduce the computational overhead due to the interconnecting traffic, we applied (9). At first glance, it may appear that the capacity utilization state set $\mathbf{M}_0(\mathbf{N})$ is large because of the wide span of the interconnecting traffic; only links $\ell_2, \ell_6, \ell_9, \ell_{15}$, and ℓ_{18} are not involved in interconnecting traffic. Actually, $\mathbf{M}_0(\mathbf{N})$ can be constructed by considering six links only; they are $\ell_{19}, \ell_{20}, \ell_{22}, \ell_{23}, \ell_{25}$, and ℓ_{26}. The status of the rest of the links can be determined by those of the above six links. Furthermore, the links belonging to the sets $\{\ell_{19}, \ell_{22}\}, \{\ell_{20}\}, \{\ell_{23}, \ell_{25}, \ell_{26}\}$ are independent of each other if only the calls in the classes belonging to $\mathbf{R}_0$ are considered. This further reduces the computations required for $Q_0(\mathbf{m})$ in (9). The computational time to determine the blocking probabilities using (9) and (10) is plotted as the solid line in Fig. 7. There are significant computational savings when compared to that of Conway *et al.*

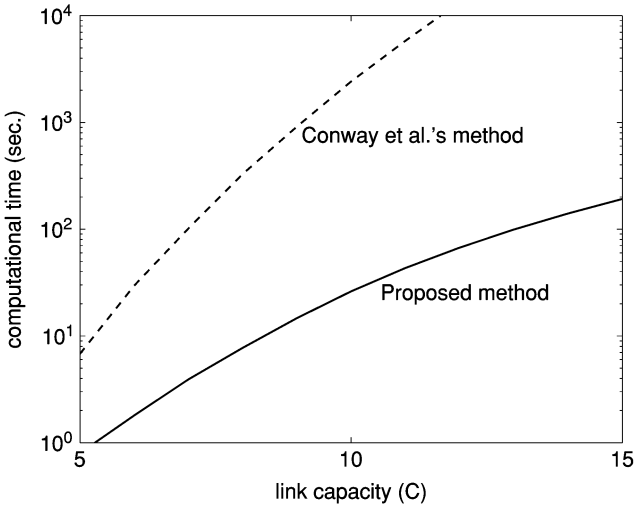

Fig. 7. Computational times required to compute the blocking probabilities of the network shown in Fig. 6. We use the same set of routing paths defined in [18] but double the number of call classes to 60. The dashed line represents the results using (7) and a partition of $\mathbf{R}$ following the topology-based decomposition method of [18]. The solid line represents computation using (9) with the same partition of $\mathbf{R}$ as above.

We have used Heuristic 1 to check if there are any better partitions of $\mathbf{R}$ (or $\mathbf{L}$). The best partition of $\mathbf{R}$ from Heuristic 1 is equivalent to that suggested by Conway *et al.* [18]. Further reduction of the computational overhead due to the interconnection traffic can be obtained but it relies on direct inspection of the traffic distribution. We did not use the link-status-based decomposition method described in Section IV because of the large number of interconnecting links. We cannot find a valid partition of $\mathbf{L}$ such that the corresponding required computational time is less than 10^4 seconds. Finally, the computational time required if either (1) or (6) is used directly, i.e., without any decomposition methods, is more than 100 hours even for $C = 5$.

C. Example 3

In this example, we demonstrate Heuristic 1. We will also show that the proposed method can be combined with Monte Carlo summation in determining the network blocking probability. In Monte Carlo summation, $G(\mathbf{N})$ is calculated using a large set of randomly generated call status vector $\mathbf{n}$. The average of the results is then used as an estimate of $G(\mathbf{N})$ to calculate the blocking probabilities [8]. The required computational time does not grow exponentially with the network size

but is proportional to the square of the accuracy requirement. Ross *et al.* developed a heuristic based on importance sampling to reduce the computational time but the reduction is not significant when the system is heavily loaded [8]. Since Monte Carlo summations use call status probability (1)–(4) to compute the blocking probabilities, call-status-based decomposition is chosen to reduce the required computational time. We do not consider link-status-based decomposition because at this time we do not have an efficient way to use Monte Carlo summation with link status probability to solve the blocking probabilities. We found that using (7), the variance of the outputs is greatly reduced if the G_k's are pre-computed and the Monte Carlo summation is only used to deal with the computations related to the inter-subnetwork traffic [26]. Hence, the computational time required to calculate the blocking probabilities at a given accuracy is reduced.

Fig. 8 shows the topology of the pan-European research network TEN-155.[1] The links in the network have four types of bandwidth: 622 Mb/s, 2 × 155 Mb/s, 155 Mb/s, and 34/45 Mb/s. We only consider links having capacity not less than 155 Mb/s. The resulting network consists of 13 nodes and 17 links. To simplify the discussion, we set all links to the same capacity of 32 channels, i.e., $\mathbf{N} = (32, \ldots, 32)$. We label the links from ℓ_1 to ℓ_{17} as shown in Fig. 8. We assume that there is only a path between any two nodes. Using shortest path method, we have 78 paths, which are listed in Table I of Appendix D. As it is not easy to partition $\mathbf{R}$ based on observation, we apply Heuristic 1.

Since the storage requirement grows rapidly with the subnetwork size, we set the maximum subnetwork size to four links. Using Heuristic 1, we first choose a transformed matrix from a set of 6 000 randomly permuted demand matrices, and obtain a partition of $\mathbf{R}$ with $|\mathbf{R}_0| = 44$. We have increased the number of randomly permuted demand matrices to 60 000 and repeat the procedure from the first step. We find a partition of $\mathbf{R}$ that has $|\mathbf{R}_0| = 42$. The search takes about 30 s. Thus, the improvement is small even if we used 10 times the estimate of the required number of randomly permuted demand matrices. We obtain a partition of $\mathbf{R}$ as $\mathbf{R} = \mathbf{R}_0 + \cdots + \mathbf{R}_5$. The details of the partitions of $\mathbf{R}$ and $\mathbf{L}$ are listed in Appendix D. The required storage is over $\mathcal{O}(10^7)$. We set the loading for each call class to two erlangs in order to compare with Ross's heuristic which can provide computational savings in such traffic condition. We require that the 95% confidence intervals are within 5% of the results. To limit the required computational time, we only consider the blocking probabilities that are larger than 10^{-4}. The Monte Carlo summation with the proposed decomposition and heuristic requires only 99 s while Ross's heuristic takes 27 454 s and simple Monte Carlo summations take 38 188 s. The reduction factor is 1/277 when compared with Ross heuristic [8] and 1/386 with simple Monte Carlo summation.

VII. Summary

In this paper, we proposed two decomposition methods and a heuristic to calculate the blocking probabilities of circuit switched networks. Based on the *noninterference* property of traffic, we have derived two decomposition methods from two perspectives: one is based on call status, and the other is based on link status. Both methods can significantly reduce the computational time in calculating the blocking probabilities, but under very different traffic conditions. We observe that the proposed methods can have better computational savings than that of the currently available decomposition methods. Examples are given to demonstrate the power of the proposed methods.

The computational savings from decomposition methods mainly depends on the partitioning of the network traffic. Traditionally, the network traffic is partitioned by inspection, which is very difficult for networks with irregular topologies. We proposed a heuristic to address this problem. From experience, the heuristic gives good computational savings when a sufficiently large number of random partitions is searched. We gave an estimate of the number of random partitions required in this search. Finally, we showed that the proposed heuristic can be combined with other methods such as Monte Carlo summation to further improve the computational efficiency.

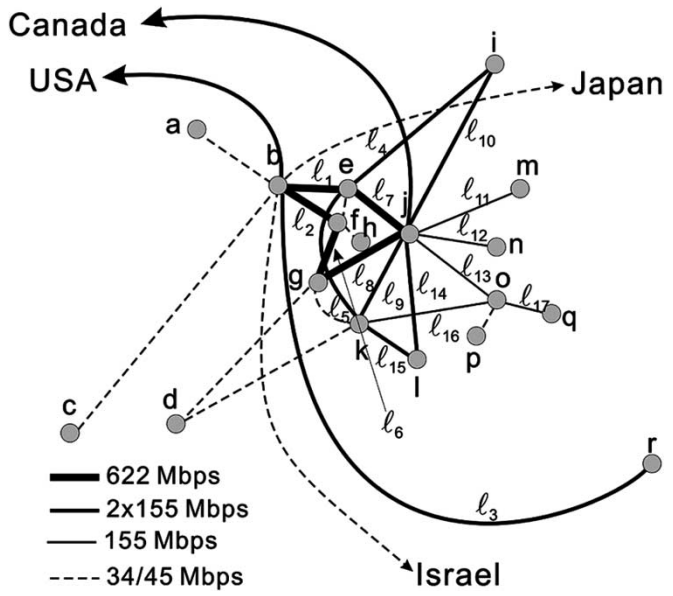

Fig. 8. Pan-European research network TEN-155.

Appendix A
Noninterference

Two call classes c_x and c_y are said to be *noninterfering* with each other if there exists a set of call classes $\mathbf{R}_0$ such that c_x and c_y are independent of each other if the status of the calls in the classes $\mathbf{R}_0$ is fixed. Note that two call classes can still be noninterfering with each other even if they share some common links. For example, we assume that classes c_x, c_y, and c_z calls request a channel on links $\{\ell_1, \ell_2\}, \{\ell_2, \ell_3\}$, and $\{\ell_1, \ell_2, \ell_3\}$, respectively. Classes c_x and c_y calls will still be independent of each other if $N_2 \geq N_1 + N_3$ where N_i, $i = 1, 2, 3$ are the capacities of links ℓ_i and the number of class c_z calls is fixed. We summarize the requirement on the link capacities for noninterfering call classes in the following condition.

[1][Online.] Available: http://www.dante.net/ten-155/

Condition 1: Let class c_x and c_y calls use sets of links $\mathbf{L}_x$ and $\mathbf{L}_y$, respectively. The two classes are noninterfering with each other if either:

- they use no common link, i.e., $\mathbf{L}_x \bigcap \mathbf{L}_y = \phi$; or
- the capacities available to both call classes from the links belonging to $\mathbf{L}_x \bigcap \mathbf{L}_y$ are equal or larger than the summation of that obtained from $\mathbf{L}_x - \mathbf{L}_x \bigcap \mathbf{L}_y$ and $\mathbf{L}_y - \mathbf{L}_x \bigcap \mathbf{L}_y$ individually under all traffic conditions.

To determine the noninterference property between call classes, one must consider both the demand matrix $\mathbf{A}$ and the link capacity vector $\mathbf{N}$.

Appendix B
Computational and Storage Requirements Using the Proposed Decomposition Methods

In this section, we estimate the computational cost and storage requirement of using (9) and (10) to compute the blocking probabilities. The resource requirement of (7) can be found in [18]. With all intermediate results saved for further computations, the computational cost and the storage requirement for solving the G_k's are $\mathcal{O}(r_k \prod_{\ell_i \in \mathbf{L}_k^+} N_i)$ $(|\Gamma_k(N)|)$ and $\mathcal{O}(\prod_{\ell_i \in \mathbf{L}_k^+} N_i)$ $(|\Gamma_k(N)|)$, respectively, if (6) [(2)] is used, where r_k is the number of call classes belonging to $\mathbf{R}_k$, and $\mathbf{L}_k^+$ is the set of links used by the calls in the classes belonging to $\mathbf{R}_k$. Equation (9) requires $\mathcal{O}(K)$ multiplication operations with $\mathcal{O}(|\mathbf{M}_0(\mathbf{N})|)$ iterations for the $G(\mathbf{N})$ given that the $Q_0(\mathbf{m})$'s are pre-computed. We need similar computations for each $B_j, c_j \in \mathbf{R}_0$. For $B_j, c_j \notin \mathbf{R}_0$, computations of $\mathcal{O}(|\mathbf{M}_0(\mathbf{N})|)$ are required according to (10). To compute $\hat{P}(\mathbf{m})$ for all $\mathbf{m} \in \mathbf{M}_0(\mathbf{N})$, we need $\mathcal{O}(r_0 \prod_{\ell_i \in \mathbf{L}_0^+} N_i)$ computations and $\mathcal{O}(\prod_{\ell_i \in \mathbf{L}_0^+} N_i)$ units of storage, where $\mathbf{L}_0^+$ is the set of links used by the calls in the classes belonging to $\mathbf{R}_0$. As the maximum size of $\mathbf{M}_0(\mathbf{N})$ is $\mathcal{O}(\prod_{\ell_i \in \mathbf{L}_0^+} N_i)$, the total computational complexity for solving all B_j with (9) and (10) is

$$\mathcal{O}\left((Kr_0 + r) \prod_{\ell_i \in \mathbf{L}_0^+} N_i + \sum_{k=1}^{K} \min\left(|\Gamma_k(\mathbf{N})|, r_k \prod_{\ell_i \in \mathbf{L}_k^+} N_i\right)\right). \tag{15}$$

The total storage requirement for the blocking probabilities computation will be

$$\mathcal{O}\left(\prod_{\ell_i \in \mathbf{L}_0^+} N_i + \sum_{k=1}^{K} \min\left(|\Gamma_k(\mathbf{N})|, \prod_{\ell_i \in \mathbf{L}_k^+} N_i\right)\right). \tag{16}$$

The computational complexity of computing $Q_k^+(\mathbf{m})$ in (11)–(14) is $\mathcal{O}(r_k \prod_{\ell_i \in \mathbf{L}_k^+} N_i)$ for $k = 0, \ldots, K$. We assume that all intermediate results during the calculation of $Q_k^+(\mathbf{m})$ are re-used for blocking probability computation. There is no extra computational cost for computing $\sum_{\mathcal{P}_0(\mathbf{m}) = \boldsymbol{\zeta}_k} Q_k^+(\mathbf{m})$. The number of iterations required to find each $\tilde{Q}(\boldsymbol{\xi})$ with (11) is $\mathcal{O}(K \prod_{\ell_i \in \mathbf{L}_0^+} N_i)$. To sum up $\tilde{Q}(\boldsymbol{\xi})$ for $G(\mathbf{N})$ or $G(\mathbf{N} - \mathbf{A}\mathbf{I}_j), c_j \in \mathbf{R}$, requires $\mathcal{O}(\prod_{\ell_i \in \mathbf{L}_0^+} N_i)$ iterations. As we need to compute the blocking probabilities of the r call classes, the computational complexity of blocking probability computation using (11)–(14) is

$$\mathcal{O}\left((r + K) \prod_{\ell_i \in \mathbf{L}_0^+} N_i^2 + \sum_{k=1}^{K} r_k \prod_{\ell_i \in \mathbf{L}_k^+} N_i\right). \tag{17}$$

Here we assume that the subnetworks do not have any special topology. Otherwise, we may take advantage of the topology features to derive efficient algorithms such as those for tree type networks [14]–[17].

The storage requirement for solving $Q_k^+(\mathbf{m})$ is $\mathcal{O}(\prod_{\ell_i \in \mathbf{L}_k^+} N_i)$ for $k = 1, \ldots, K$. The total storage requirement for the blocking probabilities computation will be

$$\mathcal{O}\left(\sum_{k=0}^{K} \prod_{\ell_i \in \mathbf{L}_k^+} N_i\right). \tag{18}$$

Appendix C
Computational Requirements Using Heuristic 1

We need computations of $\mathcal{O}(p)$ to generate a random permutation of the rows. To transform a randomly permuted demand matrix to the matrix shown in Fig. 2, we need $\mathcal{O}(r^2)$ operations. Given a set of K, $|\mathbf{R}_k|$, and/or $|\mathbf{L}_k^+|$, a number of $\mathcal{O}(r^2)$ operations are required to obtain the transformed matrix shown in Fig. 3(a) and (b). In addition, we need $\mathcal{O}(r^2)$ operations to check the noninterference property between the call classes in the subsets of $\mathbf{R}$. If we need to use Monte Carlo method to estimate the size of $\Gamma_k(\mathbf{N})$, the computation is $\mathcal{O}(|\mathbf{R}_k| \times |\mathbf{L}_k^+| \times Z)$, where Z is the number of random samples used [8]. The transformed matrix shown in Fig. 2 is chosen from a set of average $\bar{S}$ randomly permuted demand matrices. We assume that on average we have to repeat from the second step T_0 times before a suitable partition of $\mathbf{R}$ can be found. We also assume that on average T_1 sets of K, $|\mathbf{R}_k|$, and/or $|\mathbf{L}_k^+|$ are tried for each chosen transformed matrix from the second step. The average computational cost for finding a suitable partition of $\mathbf{R}$ is

$$\mathcal{O}(T_0(\bar{S}p + T_1(r^2 + \bar{K}(\bar{r}_k \bar{p}_k Z)))) \tag{19}$$

where $\bar{K}$ is the average number of subnetworks, $\bar{r}_k$ is the average number of call classes in a subset of $\mathbf{R}$, and $\bar{p}_k$ is the average subnetwork size. If all randomly permuted demand matrices in Step 2 are checked without selection, the computational cost will be

$$\mathcal{O}(\hat{T}_0(p + T_1(r^2 + \bar{K}(\bar{r}_k \bar{p}_k Z)))) \tag{20}$$

where $\hat{T}_0$ is the number of iterations required for repeating from the second step. We have $\hat{T}_0 \leq T_0 \times \bar{S}$. We approximate $\hat{T}_0$ by $T_0 \times \bar{S}$ and find that the computational savings by the heuristic will be

$$\mathcal{O}(T_0 T_1 \bar{S}(r^2 + \bar{K}(\bar{r}_k \bar{p}_k Z)))$$

when compared to the pure random trial and error approach.

TABLE I
PATHS DEFINED IN THE NETWORK SHOWN IN FIG. 8

$c_1 = \{\ell_1\}$,	$c_2 = \{\ell_2\}$,	$c_3 = \{\ell_2, \ell_6\}$,
$c_4 = \{\ell_1, \ell_4\}$,	$c_5 = \{\ell_1, \ell_7\}$,	$c_6 = \{\ell_1, \ell_7, \ell_8\}$,
$c_7 = \{\ell_1, \ell_7, \ell_{14}\}$,	$c_8 = \{\ell_1, \ell_7, \ell_{11}\}$,	$c_9 = \{\ell_1, \ell_7, \ell_{12}\}$,
$c_{10} = \{\ell_1, \ell_7, \ell_{13}\}$,	$c_{11} = \{\ell_1, \ell_7, \ell_{13}, \ell_{17}\}$,	$c_{12} = \{\ell_3\}$,
$c_{13} = \{\ell_1, \ell_2\}$,	$c_{14} = \{\ell_7, \ell_8\}$,	$c_{15} = \{\ell_4\}$,
$c_{16} = \{\ell_7\}$,	$c_{17} = \{\ell_5\}$,	$c_{18} = \{\ell_7, \ell_{14}\}$,
$c_{19} = \{\ell_7, \ell_{11}\}$,	$c_{20} = \{\ell_7, \ell_{12}\}$,	$c_{21} = \{\ell_7, \ell_{13}\}$,
$c_{22} = \{\ell_7, \ell_{13}, \ell_{17}\}$,	$c_{23} = \{\ell_1, \ell_3\}$,	$c_{24} = \{\ell_6\}$,
$c_{25} = \{\ell_2, \ell_1, \ell_4\}$,	$c_{26} = \{\ell_6, \ell_8\}$,	$c_{27} = \{\ell_6, \ell_8, \ell_9\}$,
$c_{28} = \{\ell_6, \ell_8, \ell_{14}\}$,	$c_{29} = \{\ell_6, \ell_8, \ell_{11}\}$,	$c_{30} = \{\ell_6, \ell_8, \ell_{12}\}$,
$c_{31} = \{\ell_6, \ell_8, \ell_{13}\}$,	$c_{32} = \{\ell_6, \ell_8, \ell_{13}, \ell_{17}\}$,	$c_{33} = \{\ell_2, \ell_3\}$,
$c_{34} = \{\ell_8, \ell_{10}\}$,	$c_{35} = \{\ell_8\}$,	$c_{36} = \{\ell_8, \ell_9\}$,
$c_{37} = \{\ell_8, \ell_{14}\}$,	$c_{38} = \{\ell_8, \ell_{11}\}$,	$c_{39} = \{\ell_8, \ell_{12}\}$,
$c_{40} = \{\ell_8, \ell_{13}\}$,	$c_{41} = \{\ell_8, \ell_{13}, \ell_{17}\}$,	$c_{42} = \{\ell_5, \ell_2, \ell_3\}$,
$c_{43} = \{\ell_{10}\}$,	$c_{44} = \{\ell_{10}, \ell_9\}$,	$c_{45} = \{\ell_{10}, \ell_{14}\}$,
$c_{46} = \{\ell_{10}, \ell_{11}\}$,	$c_{47} = \{\ell_{10}, \ell_{12}\}$,	$c_{48} = \{\ell_{10}, \ell_{13}\}$,
$c_{49} = \{\ell_{10}, \ell_{13}, \ell_{17}\}$,	$c_{50} = \{\ell_4, \ell_1, \ell_3\}$,	$c_{51} = \{\ell_9\}$,
$c_{52} = \{\ell_{14}\}$,	$c_{53} = \{\ell_{11}\}$,	$c_{54} = \{\ell_{12}\}$,
$c_{55} = \{\ell_{13}\}$,	$c_{56} = \{\ell_{13}, \ell_{17}\}$,	$c_{57} = \{\ell_7, \ell_1, \ell_3\}$,
$c_{58} = \{\ell_{15}\}$,	$c_{59} = \{\ell_9, \ell_{11}\}$,	$c_{60} = \{\ell_9, \ell_{12}\}$,
$c_{61} = \{\ell_{16}\}$,	$c_{62} = \{\ell_{16}, \ell_{17}\}$,	$c_{63} = \{\ell_5, \ell_1, \ell_3\}$,
$c_{64} = \{\ell_{14}, \ell_{11}\}$,	$c_{65} = \{\ell_{14}, \ell_{12}\}$,	$c_{66} = \{\ell_{15}, \ell_{16}\}$,
$c_{67} = \{\ell_{15}, \ell_{16}, \ell_{17}\}$,	$c_{68} = \{\ell_{15}, \ell_5, \ell_1, \ell_3\}$,	$c_{69} = \{\ell_{11}, \ell_{12}\}$,
$c_{70} = \{\ell_{11}, \ell_{13}\}$,	$c_{71} = \{\ell_{11}, \ell_{13}, \ell_{17}\}$,	$c_{72} = \{\ell_{11}, \ell_7, \ell_1, \ell_3\}$,
$c_{73} = \{\ell_{12}, \ell_{13}\}$,	$c_{74} = \{\ell_{12}, \ell_{13}, \ell_{17}\}$,	$c_{75} = \{\ell_{12}, \ell_7, \ell_1, \ell_3\}$,
$c_{76} = \{\ell_{17}\}$,	$c_{77} = \{\ell_{16}, \ell_5, \ell_1, \ell_3\}$,	$c_{78} = \{\ell_{17}, \ell_{16}, \ell_5, \ell_1, \ell_3\}$

APPENDIX D
PARTITIONS OF $\mathbf{R}$ AND $\mathbf{L}$ IN EXAMPLE 3

The paths c_1 to c_{78} in Example 3 are listed in Table I. Using the proposed heuristic, we have $\mathbf{R} = \mathbf{R}_0 + \cdots + \mathbf{R}_5$, where $\mathbf{R}_0 = \{c_3, \ldots, c_{11}, c_{14}, c_{18}, c_{22}, c_{25}, \ldots, c_{32}, c_{38}, \ldots, c_{41}, c_{46}, \ldots, c_{50}, c_{56}, c_{57}, c_{59}, c_{60}, c_{64}, c_{65}, c_{68}, c_{71}, c_{72}, c_{74}, c_{75}, c_{77}, c_{78}\}$, $\mathbf{R}_1 = \{c_{34}, \ldots, c_{37}, c_{43}, c_{44}, c_{45}, c_{51}, c_{52}\}$, $\mathbf{R}_2 = \{c_{16}, c_{19}, c_{20}, c_{21}, c_{53}, c_{54}, c_{55}, c_{69}, c_{70}, c_{73}\}$, $\mathbf{R}_3 = \{c_1, c_2, c_{12}, c_{13}, c_{17}, c_{23}, c_{33}, c_{42}, c_{63}\}$, $\mathbf{R}_4 = \{c_{58}, c_{61}, c_{62}, c_{66}, c_{67}, c_{76}\}$, and $\mathbf{R}_5 = \{c_{15}, c_{24}\}$. The links in the five subnetworks are $\{\ell_8, \ell_9, \ell_{10}, \ell_{14}\}$, $\{\ell_7, \ell_{11}, \ell_{12}, \ell_{13}\}$, $\{\ell_1, \ell_2, \ell_3, \ell_5\}$, $\{\ell_{15}, \ell_{16}, \ell_{17}\}$, and $\{\ell_4, \ell_6\}$, respectively.

REFERENCES

[1] P. E. Green Jr., "Optical networking update," *IEEE J. Select. Areas Commun.*, vol. 14, pp. 764–779, June 1996.

[2] R. Ramaswami and K. Sivarajan, *Optical Networks: A Practical Perspective*, 2nd ed. San Mateo, CA: Morgan Kaufmann, 2002.

[3] T. Li, "MPLS and the evolving Internet architecture," *IEEE Commun. Mag.*, vol. 27, pp. 38–41, Dec. 1999.

[4] K. C. Lee and V. O. K. Li, "A wavelength rerouting algorithm in wide-area all-optical networks," *J. Lightwave Technol.*, vol. 14, pp. 1218–1229, June 1996.

[5] K. W. Ross, *Multiservice Loss Models for Broadband Telecommunication Networks*. New York: Springer-Verlag, 1995.

[6] G. Louth, M. Mitzenmcher, and F. Kelly, "Computational complexity of loss networks," *Theoretical Comput. Sci.*, no. 125, pp. 45–59, 1994.

[7] C. Harvey, "Determining the end-to-end grade of service in a network—some new results," *Post Office Electr. Eng. J.*, pt. 2, vol. 73, pp. 109–114, 1980.

[8] K. W. Ross and J. Wang, "Monte Carlo summation applied to product-form loss networks," *Probabil. Eng. Inform. Sci.*, vol. 6, pp. 323–348, 1992.

[9] S. P. Chung and K. W. Ross, "Reduced load approximations for multirate loss networks," *IEEE Trans. Commun.*, vol. 41, pp. 1222–1231, Aug. 1993.

[10] F. P. Kelly, "Blocking probabilities in large circuit-switched networks," *Adv. Appl. Probabil.*, vol. 18, pp. 473–505, 1986.

[11] F. Theberge and R. R. Mazumdar, "New reduced load heuristic for computing blocking in large multirate loss networks," *IEE Proc.—Commun.*, vol. 143, no. 4, pp. 206–211, 1996.

[12] W. Whitt, "Blocking when service is required from several facilities simultaneously," *AT&T Tech. J.*, vol. 64, no. 8, pp. 1807–1856, 1985.

[13] A. G. Greenberg *et al.*, "Superfast parallel discrete event simulations," *ACM Trans. Modeling Comput. Simul.*, vol. 6, no. 2, pp. 107–136, 1996.

[14] Y. Kogan, "Exact analysis for a class of simple, circuit-switched networks with blocking," *Adv. Appl. Probabil.*, vol. 21, pp. 952–955, 1989.

[15] D. Mitra, "Asymptotic analysis and computational methods for a class of simple, circuit-switched networks with blocking," *Adv. Appl. Probabil.*, vol. 19, pp. 219–239, 1987.

[16] D. H. K. Tsang and K. W. Ross, "Algorithms to determine exact blocking probabilities for multirate tree networks," *IEEE Trans. Commun.*, vol. 38, pp. 1266–1271, Aug. 1990.

[17] V. B. Iversen and S. N. Stepanov, "The usage of convolution algorithm with truncation for estimation of individual blocking probabilities in circuit-switched telecommunication networks," in *Proc. 15th Int. Teletraffic Conf.*, vol. 2, 1997, pp. 1327–1336.

[18] A. E. Conway, E. Pinsky, and S. Tridandapani, "Efficient decomposition methods for the analysis of multi-facility blocking models," *J. Assoc. Comput. Mach.*, vol. 41, no. 4, pp. 648–675, 1994.

[19] G. L. Choudhury, K. K. Leung, and W. Whitt, "An algorithm to compute blocking probabilities in multi-rate multi-class multi-resource loss models," *Adv. Appl. Probabil.*, vol. 27, pp. 1104–1143, 1995.

[20] C. Y. Li and P. K. A. Wai, "General criterion for the decomposition of exact blocking computation in circuit-switched networks," *Electron. Lett.*, vol. 34, pp. 1723–1724, 1998.

[21] F. P. Kelly, *Reversibility and Stochastic Networks*. New York: Wiley, 1979.

[22] D. Everitt, "Product form solution in cellular mobile communication systems," in *Proc. 13th Int. Teletraffic Conf.*, 1991, pp. 483–488.

[23] F. P. Kelly, "Loss networks," *Ann. Appl. Probabil.*, vol. 1, no. 3, pp. 319–378, 1991.

[24] Z. Dziong and J. W. Roberts, "Congestion probabilities in a circuit-switched integrated services network," *Perform. Eval.*, vol. 7, pp. 267–284, 1987.

[25] E. Pinsky and A. E. Conway, "Computational algorithms for blocking probabilities in circuit-switched networks," *Ann. Oper. Res.*, vol. 35, pp. 31–41, 1992.

[26] C. Y. Li and P. K. A. Wai, "Computation of the blocking probabilities of circuit-switched networks using Monte Carlo method and decomposition," in *Proc. TENCON 2000*, vol. 3, pp. 60–65.

C. Y. Li (M'93) received the B.S. degree from the National Taiwan University, Taiwan, R.O.C., in 1986, and the Ph.D. degree from The Hong Kong Polytechnic University, Hong Kong, in 2000.

In 1986, he joined Taicom Ltd., where he worked as a Transmission Engineer on the M90, M135, and M405 Optical Fiber Telecommunication System projects. In 1988, he joined ROCTEC Ltd., where he worked as a Design Engineer on the computer products development. In 1993, he joined the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, as a Research Assistant, and then as a Research Fellow. His research interests include network performance evaluation, all-optical network routing, and network theory. He has published 15 papers in these areas.

P. K. A. Wai (SM'96) received the B.S. (Hons) degree with first class honors from the University of Hong Kong in 1981, and the M.S. and Ph.D. degrees from the University of Maryland, College Park, in 1985 and 1988, respectively.

In 1988, he joined Science Applications International Corporation, McLean, VA, where he worked as a Research Scientist on the Tethered Satellite System project. In 1990, he became a Research Associate in the Department of Electrical Engineering, University of Maryland. In 1996, he joined the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, as an Assistant Professor. He became an Associate Professor in 1997, and Professor and Head of the Department in 2002. His research interests include theory of solitons, modeling of fiber lasers, simulations of integrated optical devices, long-distance optical communications, all-optical packet switching, and network theories. He is an active contributor to the technical field, having over 100 international publications.

Dr. Wai is a member of the Optical Society of America.

Victor O. K. Li (M'81–SM'86–F'92) was born in Hong Kong in 1954. He received the S.B., S.M., E.E., and Sc.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1977, 1979, 1980, and 1981, respectively.

He joined the University of Southern California (USC), Los Angeles, in February 1981, and became Professor of Electrical Engineering and Director of the USC Communication Sciences Institute. Since September 1997, he has been with the University of Hong Kong, where he is Chair Professor of Information Engineering in the Department of Electrical and Electronic Engineering, and Managing Director of Versitech Ltd. (http://www.versitech.com.hk/), the technology transfer and commercial arm of the University. He also serves on various corporate boards. His research is in information technology, including high-speed communication networks, wireless networks, and Internet technologies and applications. He is a Principal Investigator of the Area of Excellence in Information Technology project funded by the Hong Kong government. Sought by government, industry, and academic organizations, he has lectured and consulted extensively around the world. He has been appointed to the Hong Kong Information Infrastructure Advisory Committee by the Chief Executive of the Hong Kong Special Administrative Region. He also serves on the Innovation and Technology Fund (Electronics) Vetting Committee, the Small Entrepreneur Research Assistance Programme Committee, the Engineering Panel of the Research Grants Council, and the Task Force for the Hong Kong Academic and Research Network (HARNET) Development Fund of the University Grants Committee. He was a Distinguished Lecturer at the University of California at San Diego, at the National Science Council of Taiwan, and at the California Polytechnic Institute. He has also delivered keynote speeches at many international conferences.

Dr. Li chaired the Computer Communications Technical Committee of the IEEE Communications Society from 1987 to 1989, and the Los Angeles Chapter of the IEEE Information Theory Group from 1983 to 1985. He cofounded the International Conference on Computer Communications and Networks (IC3N), and chaired its Steering Committee from 1992 to 1997. He also chaired various international workshops and conferences, and is most recently appointed General Chair of IEEE INFOCOM 2004. He has served as an editor of *IEEE Network*, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS Wireless Communications Series, and *Telecommunication Systems*. He has also guest edited special issues of IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, *Computer Networks and ISDN Systems*, and the KICS/IEEE *Journal of Communications and Networking*. He is now serving as an editor of ACM/Kluwer *Wireless Networks* and *IEEE Communications Surveys and Tutorials*. He has received numerous awards, including, most recently, the Outstanding Researcher Award of the University of Hong Kong, the K. C. Wong Education Foundation Lectureship, the Croucher Foundation Senior Research Fellowship, and the Bronze Bauhinia Star, Government of the Hong Kong SAR, China.