Finding Critical Traffic Matrices

Yin Zhang
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712, USA
yzhang@cs.utexas.edu

Zihui Ge
AT&T Labs – Research
180 Park Avenue
Florham Park, NJ 07932, USA
gezihui@research.att.com

Abstract

A traffic matrix represents the amount of traffic between origin and destination in a network. It has tremendous potential utility for many IP network engineering applications, such as network survivability analysis, traffic engineering, and capacity planning. Recent advances in traffic matrix estimation have enabled ISPs to measure traffic matrices continuously. Yet a major challenge remains towards achieving the full potential of traffic matrices. In practical networking applications, it is often inconvenient (if not infeasible) to deal with hundreds or thousands of measured traffic matrices. So it is highly desirable to be able to extract a small number of "critical" traffic matrices. Unfortunately, we are not aware of any good existing solutions to this problem (other than a few ad hoc *heuristics). This seriously limits the applicability of traffic matrices.*

To bridge the gap between the measurement and the actual application of traffic matrices, we study the critical traffic matrices selection (CritMat) problem in this paper. We developed a mathematical problem formalization after identifying the key requirements and properties of CritMat in the context of network design and analysis. Our complexity analysis showed that CritMat is NP-hard. We then developed several clustering-based approximation algorithms to CritMat. We evaluated these algorithms using a large collection of real traffic matrices collected in AT&T's North American backbone network. Our results demonstrated that these algorithms are very effective and that a small number (e.g., 12) of critical traffic matrices suffice to yield satisfactory performance.

1 Introduction

Today's large operational IP networks often consist of hundreds of routers, thousands of links, tens of thousands of routes, and may carry over one peta-byte ($= 10^{15}$ bytes) traffic per day. How to effectively design, engineer, and manage such large networks is crucial to end-to-end network performance and reliability. Until recently, a major obstacle to developing sound methods for network engineering in operational IP networks has been the inability of network operators to measure the traffic matrix. A *traffic matrix* represents the amount of traffic between origin and destination in a network. It is an essential input for a variety of IP network engineering applications, such as capacity planning, traffic engineering, and network survivability analysis. Due to the extreme importance of traffic matrices, there has been tremendous efforts and many recent advances in the area of traffic matrix estimation [7, 12, 15, 16]. These techniques have enabled Internet service providers to accurately measure the traffic matrix of their network in a continuous fashion (in the granularity of minutes to an hour).

Having gladly left behind the days without a good traffic matrix, however, network operators and engineers are now facing the new challenge of having to deal with hundreds or even thousands of traffic matrices, all from real measurement at different time instances. Ideally, network engineers would like to base their design and analysis on all traffic matrices for a significant period of time (*e.g.*, a couple of months). These traffic matrices can capture the normal traffic variation from temporal-geographical patterns (*e.g.*, traffic from east-coast and west-coast reaches peak usage at different hour of day) to traffic engineering activities, (*e.g.*, a customer network shifts its traffic to a different egress point during maintenance). However, in practice, it is usually inconvenient or infeasible to use a large number of traffic matrices. It is inconvenient since many traffic analysis tasks require human intervention (*e.g.*, examine the scenario where congestion has occurred). Dealing with a large number of traffic matrices is very undesirable. It is infeasible since many traffic engineering applications are very computationally expensive. For example, finding the optimal OSPF link weights that minimize link utilization for an IP network is known to be NP-complete [8]. In [8, 9], Fortz and Thorup have developed a local search technique that is demonstrated to find good solutions for real networks. However, this technique can only handle a relatively small number of traffic matrices, as the computational cost becomes prohibitive when the number of input traffic matrices is large. Given such difficulties, it is natural to ask the following question:

> *With hundreds or thousands of traffic matrices available, can we extract a small number of "crit-*

ical" ones and use only them for network design and analysis?

The above question has been asked frequently by network operators and engineers. Unfortunately, the solutions developed so far are often quite *ad hoc*. One common practice is to generate a "peak-all-elements" traffic matrix that has the peak demand for each origin-destination flow. Another approach is to take the traffic matrix at the network-wide busiest time (*i.e.*, the traffic matrix with the largest total volume). However, none of them are satisfactory – the "peak-all-elements" traffic matrix are usually too conservative since it significantly over-estimates the total traffic volume; whereas the busiest-time traffic matrix runs the risk of underestimating the demands since not all flows peak at the network peak.

In this paper, we seek to bridge the gap between the measurement and the actual application of traffic matrices by developing effective solutions to the *critical traffic matrix selection (CritMat)* problem. To the best of our knowledge, this is the first study on the critical traffic matrices selection problem. Our contributions in this paper can be summarized as follows.

- We have closely examined the critical traffic matrices selection problem in the context of network design and analysis, and identified the key requirements and desired properties.
- We have developed a mathematical problem formulation, which addresses the need for a range of practical network applications.
- We have formally analyzed the complexity of this problem and shown that it is NP-hard. We have then developed several clustering-based approximation solutions to the problem.
- We have evaluated our algorithms using a large collection of real traffic matrices. Our results have demonstrated that these algorithms are very effective and that a small number (*e.g.*, 12) of critical traffic matrices suffice to yield satisfactory performance.

The rest of the paper is organized as follows. Section 2 characterizes the requirements and properties of the critical traffic matrices selection problem and provides a mathematical formalization and a complexity analysis of the problem. Section 3 describes the algorithms that we proposed for critical traffic matrices selection. We describe our data set and evaluation methodology in Section 4 and present the performance results of our algorithms in Section 5. We conclude in Section 6.

2 Problem Description and Formalization

Network engineers are constantly facing the challenge driven by two conflicting requirements – on one hand, there is a need to design and provision the network in a robust way so that the network is prepared for any worst-case scenarios; on the other hand, constraint due to limited capital expenditure requires the network to operate in a most efficient manner. Therefore, when choosing candidate traffic matrices for network design and analysis, network engineers are often looking for a small set of *critical* traffic matrices that can represent all worst-case scenarios without inflating the traffic demand by too much. Our focus in this paper is to address the following problem: *how to extract a small number of such critical traffic matrices from a large collection (hundreds or thousands) of measured traffic matrices for network engineering purposes?*

So far, we have been vague about "worst-case" scenarios. In the context of network engineering, worst-case often refers to the situation when some link in the network is congested or heavily loaded. As manifested to network users, congestion often means high packet loss and long link delay on all traffic that traverses through this link. Thus, it is natural to use link utilization based metrics, *e.g.*, the network-wide maximum link utilization, to measure the level of "badness" of a traffic matrix. Two properties about traffic matrix and link utilization become useful here – the monotonic property and the linear property. The **monotonic property** says that when a traffic matrix X_1 has every flow smaller than or equal to the corresponding flow in another traffic matrix X_2, *i.e.*, X_2 *dominates* X_1, the link utilization for demand X_1 should always be smaller than or equal to the link utilization for demand X_2 under any routing configuration. The **linear property** states that if under a routing configuration, the link utilizations are Y_1 and Y_2 for traffic demand matrix X_1 and X_2 respectively, then for demand $X_3 = aX_1 + (1-a)X_2$, the resulting link utilizations should be $Y_3 = aY_1 + (1-a)Y_2$ under the same routing configuration. Here, a is a scaler between 0 and 1, and Y_1, Y_2, Y_3 are vectors with size equal to the number of links in the network.

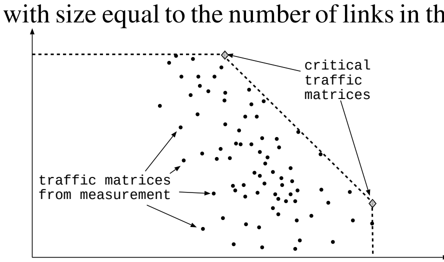


Figure 1. Example for a 2-Dimensional TM Set

The monotonic and linear properties allow us to focus on critical traffic matrices that can dominate all measurement traffic matrices by linear combination. Figure 1 illustrates an example for a two dimensional traffic matrices set. In Figure 1, the dots represent the real traffic matrices obtained from data measurement, and the two squares are the two critical traffic matrices desired. The concerns for "prepare-for-worst-case" and "minimum-oversizing" by network engineers become the requirements of 1) critical traffic matrices dominating all measurement traffic matrices by linear combination and 2) the convex hull of the critical traffic matrices having small volume (or area in the two-dimensional example). However, since traffic matrix usually has very high dimensions – a network of N nodes has N^2 flows in its traffic matrix, computing the volume of the polyhedron formed by the critical traffic matrices is a difficult task by itself. Thus we further restrict ourselves to the solution space where all critical traffic matrices are constrained to be close to an original measured traffic matrix. This turns out to be a reasonable requirement in practice. This is because for tasks such as capacity planning and weight optimization, it is desirable to conduct analysis based on traffic matrices that match real traffic patterns or scenarios.

2.1 Problem Formalization

Having carefully examined the properties and requirements of the problem, we now provide a formal statement of this optimization problem, namely the **critical traffic matrices selection** (**CritMat**) problem. We will represent each traffic matrix as a K-dimensional vector where K is a large number. Given a finite set of traffic matrices $X = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n\}$ and an integer m, the **CritMat** problem is to find a set of critical traffic matrices, $Y = \{\vec{y}_1, \vec{y}_2, \dots, \vec{y}_m\}$, that minimizes distance function

$$\max_{\vec{y} \in Y} ||\vec{y}, X||$$

subject to dominance constraint

$$\forall \vec{x} \in X, \exists a_1, a_2, \dots, a_m, \sum_{i=1}^{m} a_i = 1 \text{ and } \vec{x} \leq_d \sum_{i=1}^{m} a_i \vec{y}_i.$$

Here, $\vec{x} \leq_d \vec{y}$, *i.e.*, $\vec{x}$ being dominated by $\vec{y}$, is true if and only if $\vec{x}$ is smaller than $\vec{y}$ in all K dimensions.

The distance function $||\vec{y}, X||$ can be defined in a number of ways:

- the minimum distance to an input traffic matrix
$$||\vec{y}, X|| = \min_{\vec{x} \in X} |\vec{y} - \vec{x}|_2$$
- the maximum difference in each dimension from the closest input traffic matrix
$$||\vec{y}, X|| = \min_{\vec{x} \in X} |\vec{y} - \vec{x}|_\infty$$
- the minimum traffic demand oversizing
$$||\vec{y}, X|| = \min_{\vec{x} \in X, \vec{x} \leq_d \vec{y}} |\vec{y} - \vec{x}|_1$$
- the minimum weighted traffic demand oversizing
$$||\vec{y}, X|| = \min_{\vec{x} \in X, \vec{x} \leq_d \vec{y}} \vec{w} \cdot (\vec{y} - \vec{x})$$
where $\vec{w}$ is the geographical distance (*e.g.*, air mile) between flow source and flow destination of each flow.
- the minimum link load oversizing
$$||\vec{y}, X|| = \min_{\vec{x} \in X, \vec{x} \leq_d \vec{y}} |A(\vec{y} - \vec{x})|_1$$
where A is a routing matrix.
- the minimum oversizing in bandwidth mileage product
$$||\vec{y}, X|| = \min_{\vec{x} \in X, \vec{x} \leq_d \vec{y}} \vec{d} \cdot (A(\vec{y} - \vec{x}))$$
where A is a routing matrix and $\vec{d}$ is the length (in miles) of each link.

Note that in the above functions, the notations $|\cdot|_1$, $|\cdot|_2$ and $|\cdot|_\infty$ denote the standard ℓ_1-norm, ℓ_2-norm and ℓ_∞-norm of vector variables, respectively. More specifically,

$$|(a_1, a_2, \dots, a_r)|_1 = \sum_{i=1}^{r} |a_i|$$

$$|(a_1, a_2, \dots, a_r)|_2 = \sqrt{\sum_{i=1}^{r} a_i^2}$$

$$|(a_1, a_2, \dots, a_r)|_\infty = \max_{1 \leq i \leq r} |a_i|$$

In our analysis and solution to the **CritMat** problem, we will just deal with some instance of the distance function $||\vec{y}, X||$ and treat the other forms of the distance function as performance metrics.

2.2 Complexity Analysis

In this subsection, we will analyze the complexity of the critical traffic matrices selection problem. The decision version of the optimization problem can be stated as follows.

- PROBLEM INSTANCE: A finite set $X = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n\}$; a real number δ; an integer number m.
- QUESTION: Is there a set of vectors $Y = \{\vec{y}_1, \vec{y}_2, \dots, \vec{y}_m\}$ such that (i) $|Y| = m$, (ii) $\forall \vec{y} \in Y, ||\vec{y}, X|| \leq \delta$ and (iii) $\forall \vec{x} \in X, \exists a_1, a_2, \dots, a_m, \sum_{i=1}^{m} a_i = 1$ and $\vec{x} \leq_d \sum_{i=1}^{m} a_i \vec{y}_i$

We now show that the **CritMat** problem is NP-hard in the case where $||\vec{y}, X|| = \min_{\vec{x} \in X} |\vec{y} - \vec{x}|_\infty$ and dimension K is unbounded. We show that **3-SAT** is polynomial-time-reducible to **CritMat**.

Proof: Given an instance of a **3-SAT** problem, $C_1 \wedge C_2 \wedge \dots \wedge C_M$, where clause C_i is the disjunction of 3 literals (a literal is a variable or a negated variable), we construct a **CritMat** problem as follows.
For each variable x_i ($1 \leq i \leq N$) in the **3-SAT** problem, we create two vectors: $\vec{v}_{2i}$ and $\vec{v}_{2i-1}$ which have value 0 in all dimensions except for the $2i$-th and $(2i-1)$-th dimension. $\vec{v}_{2i}$, corresponding to x_i, has value 3 in the $2i$-th dimension and value 2 in the $(2i-1)$-th dimension and $\vec{v}_{2i-1}$, corresponding to $\neg x_i$, has value 2 in the $2i$-th dimension and value 3 in the $(2i-1)$-th dimension. For each

clause C_i, $1 \leq j \leq M$, we create another vector $\vec{c}_j$, which has value 1.75 in the three dimensions that corresponds to its three literals and has value 0 in the other dimensions. For example, the vector for clause $x_i \vee \neg x_j \vee x_k$ has value 1.75 in the $2i$-th, the $(2j-1)$-th and the $2k$-th dimension, and has value 0 in the other dimensions. We set X be the collection of *variable vectors* and *clause vectors*, i.e., $X = \{\vec{v}_1, \ldots, \vec{v}_{2N}, \vec{c}_1, \ldots, \vec{c}_M\}$ and let δ be 1 and m be the number of variables, N.

We now show that the **3-SAT** problem has a solution (a truth assignment of variables that satisfies all clauses) if and only if the constructed **CritMat** problem has a solution (a set of vectors that satisfies the problem constraint).

- $\Rightarrow$ If the original **3-SAT** problem has a solution, the set $Y = \{\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_N\}$ defined as follows is a solution of the constructed **CritMat** problem. For each variable, x_i, we define a vector $\vec{y}_i$: $\vec{y}_i$ has value 4 in the $2i$-th dimension, value 3 in the $(2i-1)$-th dimension and value 1 in the other dimensions if x_i is **T**; $\vec{y}_i$ has value 3 in the $2i$-th dimension, value 4 in the $(2i-1)$-th dimension and value 1 in the other dimensions if x_i is **F**. It is straight forward to see that $|Y| = N$ and $\forall \vec{y} \in Y, \exists \vec{x} \in X |\vec{y}, \vec{x}|_\infty \leq 1$. Furthermore, it can be verified that each pair of variable vectors, $\vec{v}_{2i}$ and $\vec{v}_{2i-1}$, are dominated by $\vec{y}_i$, and each clause vector, $\vec{c}_i$, is dominated by a linear combination of the vectors that corresponds to the variables in the clause, *e.g.*, the clause vector for $x_i \vee \neg x_j \vee x_k$ is dominated by $\frac{1}{4}\vec{y}_i + \frac{3}{8}\vec{y}_j + \frac{3}{8}\vec{y}_k$ if $x_i = x_j = \mathbf{T}$ and $x_k = \mathbf{F}$. Thus, Y is a solution to the **CritMat** problem.
- $\Leftarrow$ If the constructed **CritMat** problem has a solution: $Y = \{\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_N\}$, we know that 1) for each of the $2N$ dimensions, there exists at least one $\vec{y}_i$ that has value no less than 3 in this dimension (otherwise the corresponding variable vector is not dominated); 2) each $\vec{y}_i$ can have value no less than 3 in at most two dimensions: the $2j$-th and $(2j-1)$-th dimension for some j (otherwise $||\vec{y}_i, X|| > 1$). Combining 1) and 2), we have that each $\vec{y}_i$ has value no less than 3 in exactly two dimensions that corresponds to a pair of variable vectors. We rearrange i such that $\vec{y}_i$ dominates $\vec{v}_{2i}$ and $\vec{v}_{2i-1}$. Furthermore, since $||\vec{y}_i, X|| \leq 1$, each $\vec{y}_i$ should have values between 3 and 4 in the $2i$-th and $(2i-1)$-th dimension with at least one of them being exact 3, and have values between 0 and 1 in the other dimensions. Thus, if $\vec{y}_i$ has value greater than 3 in the $2i$-th dimension, we let $x_i = \mathbf{T}$; otherwise we let $x_i = \mathbf{F}$.

  We claim that such a truth assignment is a solution to the **3-SAT** problem. This can be shown by contradiction. Without loss of generality, we assume clause $x_i \vee \neg x_j \vee x_k$ is not satisfied, *i.e.*, $x_i = \mathbf{F}$, $x_j = \mathbf{T}$, $x_k = \mathbf{F}$, which implies that $\vec{y}_i$ has value 3 in the $2i$-th dimension and value no greater than 1 in the $(2j-1)$-th and $2k$-th dimension, $\vec{y}_j$ has value 3 in the $(2j-1)$-th dimension and value no greater than 1 in the $2i$-th and $2k$-th dimension, and $\vec{y}_j$ has value 3 in the $2k$-th dimension and value no greater than 1 in the $2i$-th and $(2j-1)$-th dimension. If we focus on the $2i$-th, $(2j-1)$-th and $2k$-th dimension, in the best case where $\vec{y}_i, \vec{y}_j$ and $\vec{y}_k$ take the upper bound value 1 in the corresponding dimensions, they can dominate a vector with value no larger than $\frac{3+1+1}{3} = \frac{5}{3}$ in all of these three dimensions. Therefore, the corresponding clause vector, which has 1.75 in these three dimensions, cannot be dominated by any linear combination of vectors in Y — contradicting to Y being a solution. Thus, all clauses are satisfied and the truth assignment is a solution to the **3-SAT** problem.

This completes the proof.

Now we have established the NP-hard-ness of **CritMat** when the objective distance function is represented by the maximum difference (in all dimensions) to the closest input traffic matrix. We conjecture that for other distance functions, the **CritMat** problem remains NP-hard due to the combinatorial nature of the solution space. In the next section, we will look into approximation approaches to critical traffic matrices selection.

3 Algorithms

In this section, we describe our approximation algorithms for **CritMat**. By looking at the structure of the problem, we find that **CritMat** shares many similarities to the traditional clustering problem for high-dimensional data set – both require finding a set of representatives that can represent the whole data set and are individually close to some member in the data set. Therefore, we will first look at some algorithms in the context of high-dimensional data clustering. We will briefly review two well-known techniques for data clustering [11, 5], *K-means* and *hierarchical agglomeration*, the latter of which has motivated our design of the *Critical-ness Aware Clustering* algorithm.

K-means starts with a randomly chosen m vectors as the center of the clusters, where m is the desired number of clusters. Each vector computes its distance to each of the m clusters, and joins the one with the minimum distance. Once a vector joins a cluster, the center of the cluster is recomputed. This step repeats until no vector can find a smaller distance to another cluster than the one that it belongs to.

Hierarchical agglomeration starts with n (singleton) clusters centered at each of the n vectors to be clustered. At each iteration, the pair of clusters with the minimum distance to each other are agglomerated into one cluster. Once agglomerated, the center of the cluster is recomputed. The process stops after $n - m$ iterations when there are m clusters left.

While the K-means and the hierarchical agglomeration algorithms generate m-partitions of the vector set such that the vectors within one partition are close to each together, in the **CritMat** problem, we are seeking a set of vectors that

```
1  initialization:
2      for each $x_i$
3          create cluster $C_i = \{x_i\}$
4          define cluster head $h_i = x_i$
5          let $m_i$ = volume of $x_i$ // max volume in $C_i$
6      for each $C_i$, $C_j$
7          compute cost$(i, j)$ =
8            volume(head$(h_i, h_j)$)-$\max(m_i, m_j)$
9      sort cost$(i, j)$ in ascending order
10 for step = 1 to $n - m$
11     merge the clusters $C_i$, $C_j$ with min cost$(i, j)$:
12     $h_i$ = head$(h_i, h_j)$
13     $m_i = \max(m_i, m_j)$
14     remove cluster $C_j$
15     for each remaining cluster $C_k$
16         cost$(i, k)$ = cost$(k, i)$ =
17           volume(head$(h_i, h_k)$) - $\max(m_i, m_k)$
18         insert cost$(i, k)$ in sorted list
19 return $\{h_i\}$ for the remaining $m$ clusters
```

Figure 2. Critical-ness Aware Clustering

are close to the original vector set and are able to dominate all vectors in the set by linear combination. One direct approach is to utilize the result from clustering algorithm by generating one "head" vector from each of the cluster such that the head vector dominates the vectors within the cluster. This can be achieved by setting the value of the head vector to the maximum value of all cluster members in each dimension, i.e., head$(\{\vec{x}_i\}) = (\max_i x_{i,1}, \max_i x_{i,2}, \ldots)$ where $\vec{x}_i = (x_{i,1}, x_{i,2}, \ldots)$. Since the cluster members are close to each other, the head vector should also be close to the cluster members, satisfying the requirement for **CritMat**. We use the result from the above method as our baseline algorithms. In the rest of the paper, we will refer to these algorithms as **K-means Head** and **Hierarchical Head** respectively, depending on which algorithm is used for clustering.

A potential problem with the above clustering-based algorithms is that clustering treats each vector in the original data set equally while **CritMat** gives more importance to the vectors with high volume. To correct this effect, we proposed a **Critical-ness Aware Clustering** (**CritAC**) algorithm by adapting the hierarchical agglomeration method. The key idea is to define a dominating cluster head, as opposed to a cluster center, and explicitly consider the oversizing cost function in every agglomeration step. **CritAC** starts with n clusters headed by each of the original n vectors. At each iteration, the pair of clusters with the minimum "merging cost" are agglomerated and the vector that has value in every dimension equal to that of the maximum among vectors from both clusters becomes the new cluster head. This process stops after $n - m$ iterations when there are m clusters left. Figure 2 presents the pseudo-code of the algorithm.

The runtime of **CritAC** can be evaluated as follows. The most expensive computation in initialization is to calculate cost(i, j) for all $n(n-1)/2$ pairs of clusters. Each pair requires $O(K)$ computations (to obtain head(C_i, C_j)), where K is the number of dimensions. Thus the runtime for line 6 to line 8 is $O(n^2 K)$. In line 9, sorting $n(n-1)/2$ cost values requires $O(n^2 \log(n))$ computation. Thus the overall runtime for initialization is $O(n^2 K + n^2 \log(n))$. For agglomeration steps, each iteration requires $O(nK + n \log(n))$ computation for calculating cost(i, k) and inserting the result into sorted list (line 16 to 19). There are $n - m$ iterations. Thus the overall runtime for agglomeration steps is $O(n(n-m)K + n(n-m)\log(n))$ and the total runtime of **CritAC** is dominated by that of the initialization steps – $O(n^2 K + n^2 \log(n))$.

We should note that it is possible to use other forms of the cost function in **CritAC**. For example, we can associate a weight for each dimension in the traffic matrices that captures the distance between the corresponding flow's origin and destination (similar to the distance function in Section 2.1). We have explored these variations of the algorithm. However, for the data set that we evaluated in Section 5, we found little difference in their performance. Therefore, we omit these variations of the algorithm and the corresponding performance results for the interest of brevity.

4 Evaluation Methodology

In this section, we first discuss the performance metrics that we use for comparing different methods. Next, we summarize the baseline algorithms for comparing against **CritAC**. We then describe the traffic matrices, network topology, and routing configuration used in our evaluation.

4.1 Performance Metrics

We use two sets of metrics to compare the performance of different methods. The first set of metrics are more direct. They are simply the objective function of **CritMat** – the distance function $||\vec{y}, X||$ – in various forms as described in Section 2.1. The second set of performance metrics are more application specific. We take the set of critical traffic matrices produced by different algorithms and use them as input for two specific network engineering applications, namely OSPF route optimization and network survivability analysis. We then evaluate the results of these applications.

4.2 Baseline Algorithms

Category	Methods
Direct clustering	**K-means Head**, **Hierarchical Head**, **Peak-all-elements**
Total volume based	**TopN**, **TopConsecN**, **Top1**

Table 1. Baseline algorithms for evaluation.

In our evaluation, we compare **CritAC** with six alternatives (as summarized in Table 1). These methods can be classified into the following two categories:

- *Direct clustering based methods*, which apply standard clustering techniques to cluster the input traffic matrices and then return the head vectors of the resulted clusters as the critical traffic matrices. **K-means Head** and **Hierarchical Head**, as described in Section 3, belong to this class. Another method that belongs to this class is **Peak-all-elements**, which returns a critical traffic matrix that has the peak demand for each individual origin-destination flow. Note that **Peak-all-elements** can be viewed as a special case of **K-means Head** and **Hierarchical Head**, with all input traffic matrices forming a single cluster.

- *Total volume based methods*, which return a subset of input traffic matrices as the critical ones based on their volumes. Two methods that belong to this class are **TopN**, which returns N traffic matrices with the highest volumes, and **TopConsecN**, which returns a set of N consecutive traffic matrices with the highest total volume (among all possible consecutive traffic matrices). We also consider a third alternative **Top1**, which is a special case of **TopN** and **TopConsecN** with $N = 1$.

Note that the traffic matrices returned by total volume based methods are not exactly critical traffic matrices in that their linear combination may not dominate all input traffic matrices. As a result, we cannot apply the direct metrics mentioned above to evaluate these methods. So we will only evaluate them in the context of network survivability analysis, and OSPF route optimization.

4.3 Traffic Matrices

Our evaluations are based on real traffic matrices collected from a large operational IP network – AT&T's North American commercial backbone network. The network consists of tens of Point of Presence (PoPs), hundreds of routers, thousands of links, and carries over one petabyte of traffic per day.

The traffic matrices are estimated from SNMP link load measurements using the *tomo-gravity* method [15, 16], which has been shown to yield accurate estimates especially for large traffic matrix elements. We use hourly traffic matrices, as they are commonly used in network engineering applications. The data collection in our study contains more than five months of hourly traffic matrices (from January 2, 2004 to June 10, 2004), which provide us 3048 instances of traffic matrices in total. The traffic matrices in our original dataset are at the router level. For simplicity, we aggregate them into PoP-level traffic matrices (so that we don't have to deal with changes in the number of origin-destination flows due to router failures or newly added routers). Each PoP-level traffic matrix contains over 400 origin-destination flows at rates ranging from tens of Kbps to tens of Gbps.

4.4 Network Topology and Routing

We use the PoP-level network topology on June 10, 2004 in our evaluations. We first obtain the router-level topology using the methods by Feldmann *et al.* [7]. We then reduce the router-level topology into a PoP-level topology by collapsing all the router-level links between each PoP into a single PoP-level link. The capacity of the PoP-level link is computed as the sum of the capacities of all the underlying router-level links. We also compute the geographical distance (in miles) for each PoP-level link using the latitude/longitude information for its two end points.

In order to translate traffic matrices into link loads or utilization, we need to have the routing information. By default, Cisco routers [4] set the OSPF weight of each link to be inversely proportional to its capacity — we refer to this setting as the *InvCap* weight setting and use it as the default configuration for our PoP-level topology.

5 Results

In this section, we present the results of our evaluation of different methods for selecting critical traffic matrices. We first compare different clustering based methods based on a set of direct performance metrics (*i.e.*, various forms of the distance function $||\vec{y}, X||$ as described in Section 2.1). We then report on the performance of different methods in the context of network survivability analysis and OSPF route optimization.

5.1 Simple Metrics

In this section, we evaluate **CritAC** against direct clustering based methods, using various forms of $||\vec{y}, X||$ as our performance metrics. We cannot report the exact values of $||\vec{y}, X||$, as they are considered proprietary. So instead we only report on the normalized values. The normalized metrics are summarized below:

- the traffic demand oversizing ratio

$$\frac{\max\limits_{\vec{y}\in Y} \min\limits_{\vec{x}\in X, \vec{x}\leq_d \vec{y}} |\vec{y}-\vec{x}|_1}{\max\limits_{\vec{x}\in X} |\vec{x}|_1}$$

- the weighted demand oversizing ratio

$$\frac{\max\limits_{\vec{y}\in Y} \min\limits_{\vec{x}\in X, \vec{x}\leq_d \vec{y}} \vec{w}\cdot(\vec{y}-\vec{x})}{\max\limits_{\vec{x}\in X} \vec{w}\cdot\vec{x}}$$

  where $\vec{w}$ is the geographical distance (*i.e.*, air miles) between the flow source and destination of each flow.

- the link load oversizing ratio

$$\frac{\max\limits_{\vec{y}\in Y} \min\limits_{\vec{x}\in X, \vec{x}\leq_d \vec{y}} |A(\vec{y}-\vec{x})|_1}{\max\limits_{\vec{x}\in X} |A\vec{x}|_1}$$

  where A is the routing matrix obtained using the default *InvCap* weight setting.

- the weighted link load oversizing ratio

$$\frac{\max\limits_{\vec{y}\in Y} \min\limits_{\vec{x}\in X, \vec{x}\leq_d \vec{y}} \vec{d}\cdot(A(\vec{y}-\vec{x}))}{\max\limits_{x\in X} \vec{d}\cdot A\vec{x}}$$

where A is the routing matrix obtained using the default *InvCap* weight setting, and $\vec{d}$ is the length (in miles) of each link.

We compute the above oversizing ratios for the three clustering based methods using either 500 or 1000 consecutive traffic matrices, which correspond to 22 days or 44 days of measurement, at different times during the 5-month period of our data collection. For the interest of brevity, we only present the results for the first 1000 traffic matrices. The results for other periods of time are qualitatively similar.

Figure 3 shows the four different kinds of oversizing ratio as a function of the number of critical traffic matrices desired. We observe that different performance metrics have given consistent result. If we look at the case at which the number of critical traffic matrices is 1, which corresponds to **Peak-all-elements**, we find that the oversizing ratio is more than 100% in all graphs (the left most point in each graph). This confirms our intuition that **Peak-all-elements** is an overly conservative method.

Comparing **CritAC** with **Hierarchical Heads** and **K-means Heads**, we find that **CritAC** achieves much lower oversizing ratio. With **CritAC**, the oversizing ratio decays rapidly to below 20% with only 24 critical traffic matrices. In contrast, with **Hierarchical Heads** and **K-means Heads**, the oversize ratio undergoes a quick decay to about 40% and then decays very slowly – even with 200 critical traffic matrices, the oversizing ratio is still close to the oversizing ratio achieved by 24 critical traffic matrices produced by **CritAC**. This demonstrates the advantage of being criticalness aware.

5.2 OSPF Route Optimization

Open Shortest Path First (OSPF) [13] and Intermediate System-Intermediate System (IS-IS) [3] are the two most commonly used intra-domain routing protocols today. In these protocols, which are functionally identical, each link is associated with a positive weight, and the length of a path is defined as the sum of the weights of all links on that path. Traffic is routed along the shortest path(s). In case of ties where several outgoing links are on shortest paths to the destination, the flow is split roughly evenly.

Over the years, many methods [8, 14, 10, 6, 2] have been proposed to obtain a set of link weights that minimizes the level of congestion in the resulting shortest-path-routed network. We refer to such a method as an *OSPF optimizer*, though the resulted weight setting is equally applicable to IS-IS routing.

In our evaluation, we have used the approach in [8, 9], which is based on a so-called local search technique [1]. The method uses heuristics to iteratively improve the weight setting, changing one or a few weights in each iteration. We ran each problem configuration for 5000 iterations – each taking a few minutes on average. Since the problem of finding an optimal weight setting is NP-hard [8], we cannot guarantee finding the true optimum. The quality of the final weight setting is affected by random choices made through the iterations, causing some variance in the quality of the outcome.

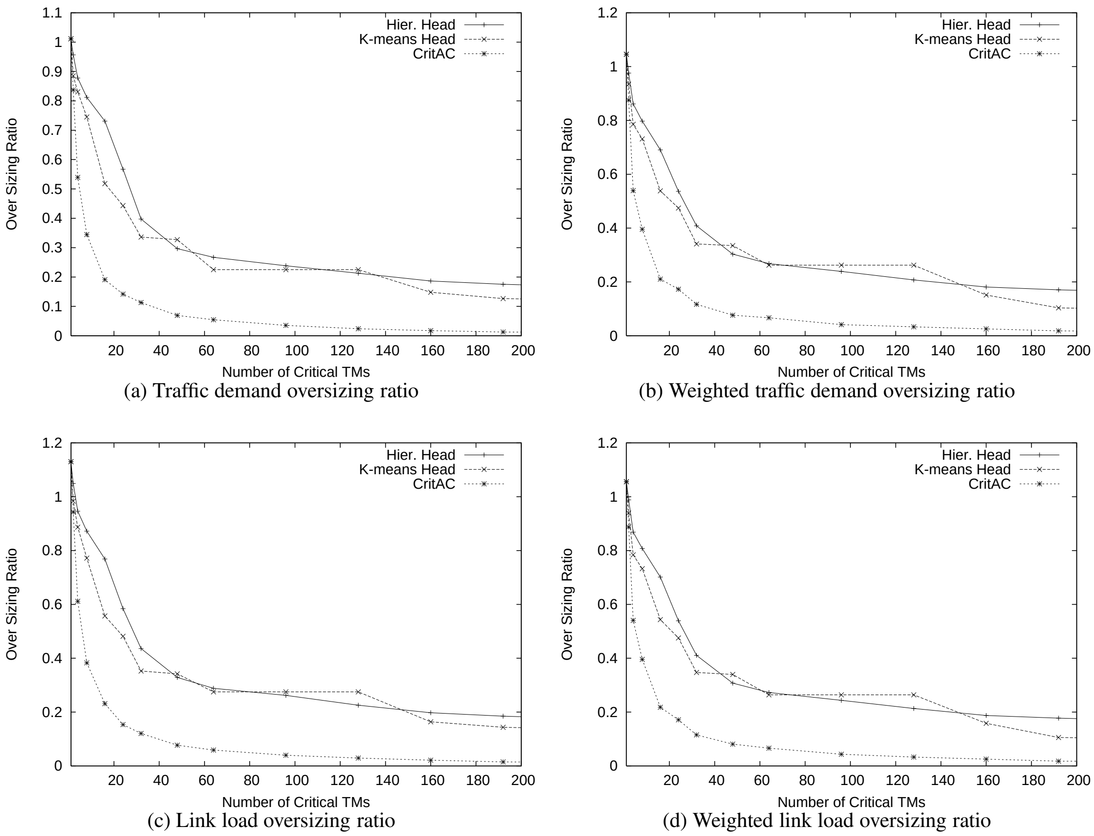
To evaluate the performance of different methods in the context of OSPF route optimization, we have conducted a number of experiments in both offline and online settings. In the offline setting, we first obtain a set of critical traffic matrices from 1000 input traffic matrices using one of our algorithms; then we feed the obtained critical traffic matrices to the OSPF optimizer and compute for a set of OSPF link weights; we test the performance of these weights on the same 1000 input traffic matrices. In the online setting, we optimize routing using critical traffic matrices derived from the first 500 traffic matrices and then test the performance of the obtained routing configuration on the next 500 traffic matrices.

As in [10], we pick max-utilization as the performance metric for route optimization in our evaluation. The *utilization* of a link is defined as the ratio of its load over its capacity, and the *max-utilization* is the maximum utilization of all links in the network. Other works have used more sophisticated metrics, such as the total utilization *cost* of all links in the network (*e.g.*, [14, 8]). However they are difficult to interpret and do not provide additional insights for our purpose, thus are not used. Furthermore, since the exact value of link utilization is considered proprietary, we normalize the max-utilization by the maximum link utilization achieved by the default weight setting – *InvCap*. For example, a normalized max-utilization of 0.7 means that the corresponding weight setting performs 30% $(1 - 0.7)$ better than *InvCap*.

Figure 4(a) shows the results for the offline setting when we take the first 1000 traffic matrices as input. We observe that when the number of critical traffic matrices allowed is big enough (*e.g.*, 12 or 24), **CritAC** significantly outperforms **Hierarchical Head**, **TopN**, and **TopConsecN**. In addition, compared to the default weight setting (*InvCap*), **CritAC** is able to reduce max-utilization by over 35%. Interestingly, **K-means Head** performs slightly better than **CritAC**, although the difference is very small (below 2%). This small discrepancy rises because the OSPF optimizer [9] in our evaluation optimizes the average max-utilization for the input traffic matrices, which usually produces good, however not necessarily the best maximum max-utilization over all traffic matrices.

Figure 4(b) shows the results for the online setting when we take the first 500 traffic matrices as input. Again, we see that **CritAC** consistently achieves good performance – the results are either significantly better than or close to that of alternative methods. Since the results are consistent with that of offline setting, it demonstrates that the critical traffic matrices identified by **CritAC** are robust enough for on-line analysis in the context of route optimization.

Overall, the results in this subsection have suggested that



(a) Traffic demand oversizing ratio

(b) Weighted traffic demand oversizing ratio

(c) Link load oversizing ratio

(d) Weighted link load oversizing ratio

Figure 3. Oversizing ratios for different clustering based methods.

it is sufficient to only use a small number (*e.g.*, 12) of critical traffic matrices for OSPF route optimization and **CritAC** can provide good input for this purpose.

5.3 Network Survivability Analysis

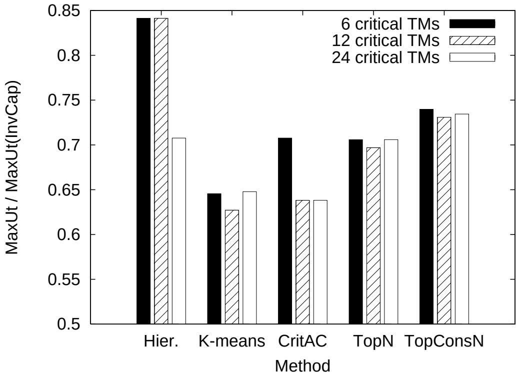
A well engineered network should work well not only under normal conditions, but also under common failures. Network survivability analysis is the key network engineering application that makes this possible. The basic task of network survivability analysis is to simulate the network under all single or dual shared-risk-device-group failures (*e.g.*, link, router, or fiber span failures). For a large network, the possible failure scenarios can be very large. Thus it maybe either infeasible or computationally very expensive to test all failure scenarios on a large number of input traffic matrices.

In this section, we examine the performance of using only the critical traffic matrices for network survivability analysis. Our basic evaluation methodology is as follows. We first use one of our algorithms to obtain a set of critical traffic matrices from an input set of measured traffic matrices; we then perform survivability analysis based on the output set of the critical traffic matrices and compare the predicted performance to the "true" performance when survivability analysis is based on all measured traffic matrices.

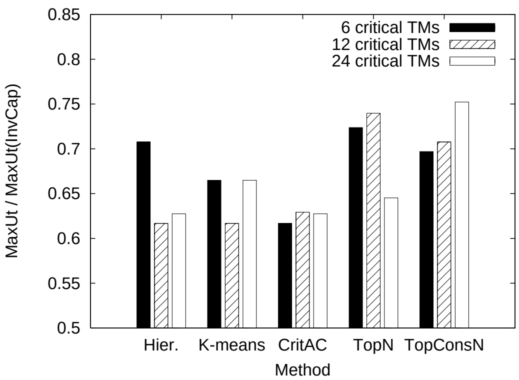
In our evaluation, we use *InvCap* as the routing configuration, and the max-utilization as our performance metric. The survivability analysis we perform consists of simulating all possible single link failures and computing the max-utilization under each failure scenario. Given a failure scenario f, we use $\text{predictedMaxUt}(f)$ and $\text{trueMaxUt}(f)$ to denote the max-utilization computed using the critical traffic matrices and original traffic matrices, respectively. We then compute the empirical Cumulative Distribution Function (CDF) for $\text{predictedMaxUt}(f)/\max_f\{\text{trueMaxUt}(f)\}$ (again because the exact value of $\text{predictedMaxUt}(f)$ is considered proprietary).

The results are summarized in Figure 5. To avoid putting too many curves into the same plot, we partition the comparison into two parts. The left column of Figure 5 compares **CritAC** against direct clustering based methods: **K-means Head**, **Hierarchical Head**, and **peak-all-element**; the right column compares **CritAC** against total volume based methods: **TopN**, **TopConsecN**, and **Top1**. We also vary the number of critical traffic matrices. In Figure 5, the two plots in each row shares the same number of critical traffic matrices.

From Figure 5, it is evident that the total volume based methods tend to underestimate the max-utilization for a considerable fraction of failure scenarios. Such underestimation can be quite undesirable in network reliability analysis in that it may cause serious problems to go undetected during the analysis process (and become evi-



(a) offline performance



(b) online performance

Figure 4. Performance of route optimization.

dent later during actual operations). In contrast, the clustering based methods are guaranteed to never underestimate. Among the clustering based methods, **CritAC** clearly performs the best in that it leads to the least amount of overestimation. Finally, compared with the real max-utilization, we see that for **CritAC** with 12 or 24 critical traffic matrices, $\max_f \text{predictedMaxUt}(f)$ is very close to $\max_f \text{trueMaxUt}(f)$ – the difference is always below 3%. These results suggest that **CritAC** can be used to significantly reduce the number of traffic matrices that need to be tested for reliability analysis.

6 Conclusion

In this paper, we defined the critical traffic matrices selection (**CritMat**) problem. We identified its properties and requirements in the context of network design and analysis. Based on these properties, we developed a mathematical formalization of the problem. We conducted complexity analysis of **CritMat** and showed that it is NP-hard. Furthermore, we proposed several clustering-based approximation algorithms to solve the problem and evaluated and compared their performance using application independent metrics. Finally, we applied our techniques to two network engineering applications – OSPF route optimization and network survivability analysis. Our results using real network data demonstrated that a small number (*e.g.*, 12) of critical traffic matrices suffice to cover all the worst-case scenarios without being overly conservative.

References

[1] E. H. Aarts and J. K. Lenstra. *Local Search in Combinatorial Optimization*. John Wiley & Sons, Inc., 1997.

[2] L. S. Buriol, M. G. C. Resende, C. C. Ribeiro, and M. Thorup. A memetic algorithms for OSPF routing. In *Proceedings of 6th INFORMS Telecom*, pages 187–188, 2002.

[3] R. Callon. Use of OSI IS-IS for routing in TCP/IP and dual environments. IETF RFC 1195, Dec. 1990. `http://search.ietf.org/rfc/rfc1195.txt`.

[4] Cisco. Configuring OSPF, 2001. Documentation at `http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/ip_c/ipcprt2/1cdospf.htm`.

[5] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., 2000.

[6] M. Ericsson, M. Resende, and P. Pardalos. A genetic algorithm for the weight setting problem in OSPF routing. *J. Combinatorial Optimization*, 6(3):299–333, 2002.

[7] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. *IEEE/ACM Transactions on Networking*, 9(3):265–279, 2001.

[8] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings of IEEE INFOCOM*, pages 519–528, 2000.

[9] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications (Special Issue on Recent Advances on Fundamentals of Network Management)*, 20(4):756–767, 2002.

[10] F. Lin and J. Wang. Minimax open shortest path first routing algorithms in networks supporting the SMDS services. In *Proceedings of IEEE International Conference on Communications (ICC)*, volume 2, pages 666–670, 1993.

[11] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

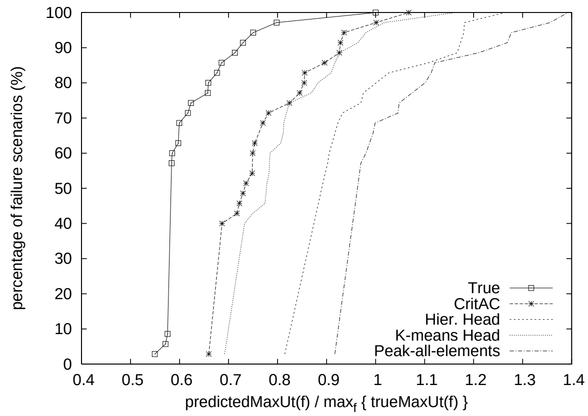
[12] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *Proceedings of ACM SIGCOMM*, Aug. 2002.

[13] J. T. Moy. OSPF version 2. IETF RFC 2328, Apr. 1998. `http://search.ietf.org/rfc/rfc2328.txt`.

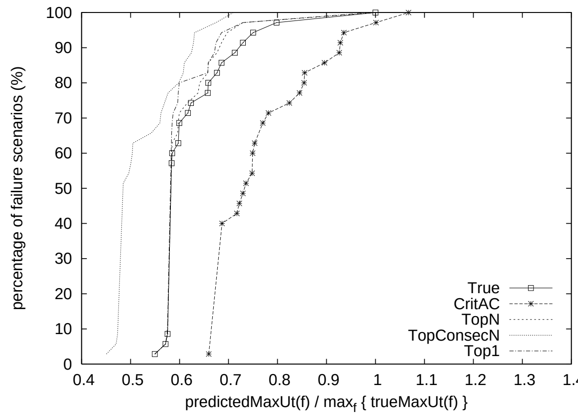
[14] K. Ramakrishnan and M. Rodriguez. Optimal routing in shortest-path data networks. *Lucent Bell Labs Technical Journal*, 6(1), 1994.

[15] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale IP traffic matrices from link loads. In *Proceedings of ACM SIGMETRICS*, 2003.

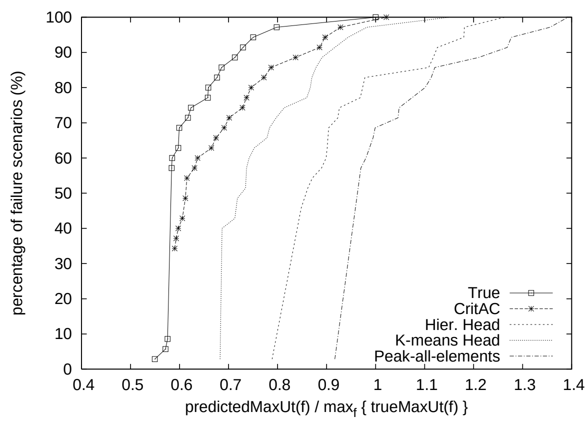
[16] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation. In *Proceedings of ACM SIGCOMM*, 2003.



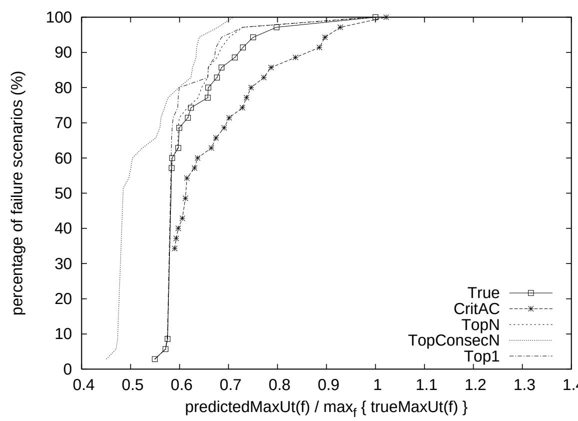
(a) **CritAC** vs. direct clustering (6 critical TMs)



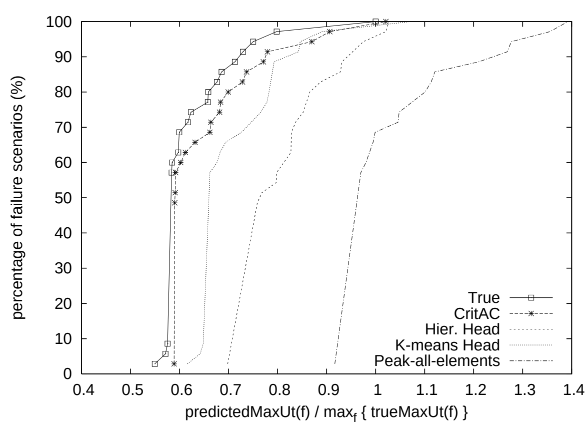
(b) **CritAC** vs. using high-volume TMs (6 critical TMs)



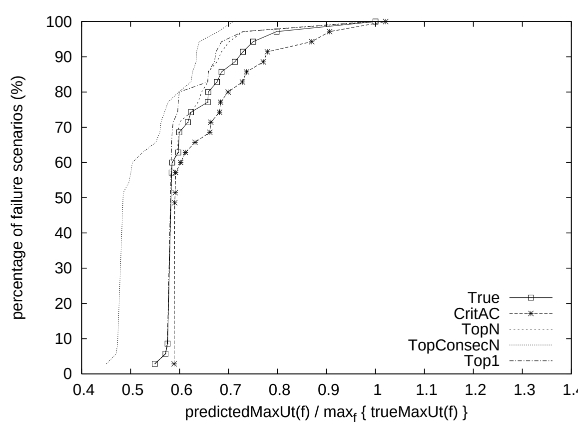
(c) **CritAC** vs. direct clustering (12 critical TMs)



(d) **CritAC** vs. using high-volume TMs (12 critical TMs)



(e) **CritAC** vs. direct clustering (24 critical TMs)



(f) **CritAC** vs. using high-volume TMs (24 critical TMs)

Figure 5. Performance comparison between CritAC and alternative methods in network survivability analysis.