Dynamic Optimization of OSPF Weights using Online Simulation

Tao Ye, Hema Tahilramani Kaur, Shivkumar Kalyanaraman, Kenneth S. Vastola and Saroj Yadav
Electrical Computer and Systems Engineering Department,
Rensselaer Polytechnic Institute, Troy, NY-12180.
{yet3,hema,shivkuma,vastola}@networks.ecse.rpi.edu, yadavs@rpi.edu

Abstract—**In this paper, we address the issues associated with the dynamic optimization of OSPF weights. For this work we have chosen the packet loss rate in the network as the optimization metric as it is a good indicator of congestion in the network and also impacts the performance of the underlying applications. The packet loss rate has been formulated in terms of the link parameters, such as bandwidth and buffer space, and the parameters of the traffic demands. A GI/M/1/K queueing model has been used to compute the packet drop probability on a given link in the network. We have developed a fast adaptive random search algorithm to address the issue of speed in the dynamic optimization problem. The proposed search algorithm has been compared to the local search algorithm of [6] in terms of the number of iterations needed to obtain a "good" link weight setting. Both the piece-wise linear optimization function from [6] and the packet loss rate in the network have been considered as the performance metrics.**

Our results demonstrate that the adaptive random search takes 50-90% fewer iterations as compared to the local search [6]. The amount of improvement depends on the network topology, traffic and the optimization metric. We have also demonstrated improvements of the order of 30-60% in the total packet loss rate in the network by dynamically optimizing OSPF weights.

Keywords—**Traffic Engineering, OSPF, Optimization**

I. Introduction

In this paper, we address the problem of traffic engineering in a network of OSPF routers. Traffic engineering is defined as the task of mapping traffic flows onto an existing physical topology. By evenly balancing the traffic across the network, congestion caused by uneven distribution of traffic can be avoided. Traffic engineering is becoming essential for ISPs due to an ever-increasing need to provide a good quality of service to customers and to sustain large growth in traffic. Two main approaches have been taken to solve the traffic engineering problem in the Internet. The first approach is to optimize the link weights of the existing network (running OSPF) such that the OSPF routing with these link weights leads to desired routes [6]. Another approach is to deploy the emerging MPLS technology which is not constrained by the shortest path nature of routing. Constraint-based routing can be used to compute routes in an MPLS network subject to QoS and policy constraints. However, increased communication and computation overhead, increased routing table size and potential routing instability are some of the cons of constraint based routing. Instead of using constraint based routing, an overlay approach may be used for establishing logical connections between every source and destination.

The main issue with using existing OSPF routing for traffic engineering is the shortest path nature of OSPF. OSPF routes traffic on shortest paths based on the advertised link weights. As a result, the link along the shortest path between the two nodes may become congested while the links on longer paths may remain idle. OSPF also allows for Equal Cost Multi Path(ECMP) where the traffic is distributed equally among various next hops of the equal cost paths between a source and a destination [14]. This is useful in distributing the load to several shortest paths. However, the splitting of load by ECMP is not optimal as shown in [7]. Various methods have been proposed in literature to balance the traffic across the network in OSPF routing framework. One of the earlier approaches was to adapt link weights to reflect the local traffic conditions on a link or to avoid congestion ([11], [8], [13]). This is called adaptive routing or traffic-sensitive routing. However, adapting link weights to traffic conditions leads to frequent route changes and is unstable (see [2], [17] for stability analysis). The above schemes were based on the local information and independent local decisions were made by the routers to change the link weights. Routers generally do not have any knowledge of the traffic load on distant links and therefore cannot optimize traffic allocation. Though adaptive routing approach is fully distributed in nature, it suffers from the lack of traffic information. These drawbacks are alleviated in [6] when optimum OSPF link weight settings are computed based on an estimate of the traffic demands. A local search algo-

This work was supported by the DARPA Contract No. F30602-00-2-0537

rithm has been used to obtain the optimum link weights for all the links in the network. These link weights, when deployed in the network, lead to the desired routing under the shortest path framework of OSPF. In [6], authors demonstrate the concept of optimizing OSPF weights for a piece-wise linear objective function. They show that for the proposed AT&T WorldNet backbone, OSPF can yield a routing that is within few percent of the optimal general routing. It may be noted that optimal general routing is the best that can be achieved by carefully setting up multiple Label Switched Paths (LSPs) in MPLS. Hence the question is: Is the OSPF performance close enough to the optimal general routing? This question has been answered in [6]. The answer is "yes", though the difference in performance depends on the network topology and the traffic demand. In [6], authors have shown that even for the randomly generated network topologies, 50%-110% more demand can be supported by using optimal OSPF weight setting as compared to setting link weights based on some standard heuristic.

The optimization problem considered in [6] is, what we call a *static optimization* problem. They have chosen a link penalty function which reflects the idea that it is cheap to send a flow over a link with small utilization, whereas, it is expensive to send a flow over heavily loaded links. Hence, minimization of the penalty function would avoid any links from being overloaded. The penalty function is chosen such that for offered load greater than 100%, a heavy penalty is paid and for loads greater than 110%, the penalty is so high that this should never occur. In loose terms, their goal was to avoid any link in the network from being overloaded. The goal of their work was to demonstrate the concept of optimization of OSPF weights for balancing traffic when the network demands are known. In the best case, their scheme may be used a few times during the day for optimizing the weights.

In this paper, we consider a *dynamic optimization* problem where periodic information about the traffic demands is available, either by monitoring the traffic at the edge-routers or by the information about service level agreements (along with some policing at the ingress routers). This information about the traffic demands is used to obtain the optimal link weight setting for the current traffic conditions. Issues related to estimating the demands (on a source-destination level) by monitoring the traffic at the ingress routers has been studied in [5]. It is more sensible to collect the flow-level statistics instead of the packet-level statistics. The flow-level statistics of aggregate traffic may be monitored at the ingress [3], [9]. In [19] also, authors discuss the issues of online traffic modeling and workload generation for simulations.Traffic monitoring is an important component for solving the *dynamic optimization* problem. However, issues associated with monitoring traffic to obtain demand estimates is not a focus of this paper and we do not discuss it further. We have assumed that information about the mean and variance of the aggregate traffic from every source to every destination router is available to us periodically. The underlying assumption in any dynamic optimization problem is that the traffic is quasi-stationary i.e. the traffic parameters vary on longer time scales as compared to the time required for optimization.

Primarily, following components are needed in order to solve the problem of *dynamic optimization* of OSPF weights. Firstly, an automatic network management tool is needed to both monitor the traffic to obtain the demand estimates and to deploy the optimal link weight settings in the network (this may be done using SNMP). Secondly, a scheme needs to be developed to search large parameter spaces to obtain link weight settings for improved performance. It has been pointed out in [16], [10] that local search schemes are not efficient in high-dimensional optimization problems and are also *susceptible to noise in the objective function*. Hence, the local search scheme proposed in [6] cannot be used for dynamic optimization because of its *inefficiency*.

The main contribution of this paper is to demonstrate substantial improvement in performance in terms of the packet loss rate by dynamic optimization of OSPF weights. Another contribution is to propose a fast search scheme and to demonstrate substantial improvement in the speed of optimization by using the proposed scheme. We formulate the optimization metric as the packets dropped in the network in terms of the estimated mean and variance of the traffic demands. The packet drop probability for each link is computed using a GI/M/1/K queueing model. We have used a Generalized Exponential(GE) distribution to model the general inter-arrival process. We justify the choice of GE distribution in Section III. Also, we have developed a fast adaptive random search algorithm for global optimization problems with large number of parameters. We have compared the performance of our adaptive random search algorithm with the local search algorithm in [6]. Throughout this paper, we refer to the search scheme proposed in [6] as the local search scheme. We have demonstrated that adaptive random search is more suitable for finding improved OSPF weights for *dynamic optimization* as it can find a "good" parameter setting much faster than the local search scheme. Improved OSPF weights can be deployed in the real network dynamically by using the Online Simulation (OLS) framework. OLS framework is a general framework for automatic network management

that can be used for tuning the parameters of network protocols and is not specific to OSPF. The OLS framework has been further described in Section IV-A.

This paper is organized as follows. Section II derives the objective function to minimize the packets dropped in the network. Section III formulates the optimal routing as a non-linear programming problem for the general routing and discusses the equivalent problem for the OSPF routing. Throughout this paper, we use the term general optimal routing to represent routing where there is no limitation on the way a flow is split among multiple paths available between a source and destination. Section IV highlights the issues associated with dynamic optimization and describes our solutions. Section V presents the simulation results and finally, Section VI presents the conclusions and future work.

II. Objective Function

Our goal is to minimize the packets dropped in the network for a given mean and variance of the aggregate demands between each source and destination routers.

Let us consider a network represented by a directed graph $\mathcal{G}$=($\mathcal{N}$,$\mathcal{L}$), where $\mathcal{N}$ and $\mathcal{L}$ represent respectively the set of routers and links in the network. Each link $l \in \mathcal{L}$ has bandwidth denoted by B_l and a buffer space of K_l packets. We assume that packets arriving when the buffer space at a link is full are dropped and there is no other active queue management algorithm running at the routers. In addition to the knowledge of bandwidth and buffers at all the links, we assume that an estimate of the mean and variance of the aggregate demand from each source s to destination t is known. Let $\mathcal{D}$, $\mathcal{V}$ denote the mean and variance matrix of the estimated aggregate demands.

Let P_l denote the packet drop probability and λ_l, σ_l^2 denote the mean, variance of the offered load to a link $l \in \mathcal{L}$ in packets per second. The objective to minimize packets dropped in the network may formulated as a non-linear programming problem as follows.

$$\min\ \Phi = \sum_{l \in \mathcal{L}} \lambda_l P_l \tag{1}$$

In order to find a closed-form expression for the packet drop probability on link l, P_l, let us assume an exponentially distributed packet size with mean $\bar{X}$. However, we consider a general arrival process. We compute the packet drop probability at a link l using a GI/M/1/K queueing model. The drop probability of a finite GI/M/1/K has been approximated by an infinite buffer GI/M/1 queue [12] using the following equation.

$$P(N_K = K) = \frac{P(N_\infty = K)}{P(N_\infty \leq K)} \tag{2}$$

N_K denotes the number of packets in the finite buffered queue, whereas, N_∞ denotes number of packets in the infinite buffer GI/M/1 queue. The queue length distribution of GI/M/1 queue is given by [4]:

$$P(N_\infty = j) = A\omega^{j-1} \qquad (j \ \geq \ 0) \tag{3}$$

ω can be obtained by solving the following equation for ω and A is the normalization constant.

$$\omega = \gamma((1-\omega)\mu) \tag{4}$$

$\gamma(s)$ is the Laplace transform of the arrival process and μ is the service rate. For a link $l \in \mathcal{L}$, the service rate μ_l is given by $\frac{B_l}{\bar{X}}$. In order to solve (4) for ω, we need to assume a distribution for the arrival process. Let us consider the Generalized Exponential (GE) distribution for modeling the arrival process to first two moments. We discuss below the reason for choice of GE distribution. The pdf of GE distribution is given by

$$g(x) = (1-p)\delta(0) + pae^{-ax} \tag{5}$$

Taking the Laplace transform, we get,

$$G(s) = 1 - p + \frac{pa}{s+a} \tag{6}$$

As can be seen from (5), a GE process is characterized by two parameters, p and a. GE distribution is a special case of H_2 distribution and can be used to model general inter-arrival processes that are more bursty as compared to the Poisson process. For a Poisson process the variance is equal to the square of mean. Hence, GE distribution may be used to model the first two moments of processes with variance greater than the square of mean. If the arrival process is represented by a GE distribution, then, with probability p the inter-arrival time is exponentially distributed with mean a and with probability $1-p$, the inter-arrival time is zero. Hence, this distribution represents a batch arrival process with geometrically distributed batch size and exponentially distributed inter-batch arrival times. For a link $l \in \mathcal{L}$, $p_l = \frac{2\lambda_l^2}{\sigma_l^2+\lambda_l^2}$, $a_l = p_l\lambda_l$ represent the parameters of the GE distribution representing the arrival process. The merging of N independent GE(p_i,a_i) processes is a bulk-arrival Poisson process with mean arrival rate a equal to $\sum_{i=1}^{N} a_i$ and p equal to $a/\sum \frac{a_i}{p_i}$. Similarly, splitting of a GE(p,a) process into N streams according to a

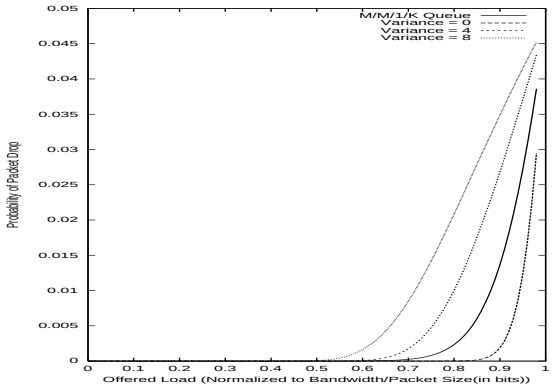


Fig. 1. Figure showing packet drop probability as a function of offered load for a GE/M/1/20 queue for different values of variance

Bernoulli filter $r_1, r_2, ...r_N$, the parameters of the i^{th} process are $p_i = \frac{p}{p(1-r_i)+r_i}$, $a_i = r_i a$. Reader may refer to [15], Section 1.4 for more details.

The packet arrival process of a single TCP flow is bursty in nature with a "bulk" of packets arriving every round-trip time. The model that we have considered implies that we have "bulk" arrivals (in form of bursts of packets from competing TCP sources) of varying sizes arriving into a queue. Our model does not capture the feedback effect of packet drops on TCP flows because we have considered the aggregate traffic arriving at an OSPF router as our demand estimate.

Solving (4) for ω for the GE arrival process given by (5) gives

$$\omega = \rho + (1 - p) \tag{7}$$

where, $\rho = \frac{a}{\mu}$. For a link $l \in \mathcal{L}$, $\rho_l = \frac{a\bar{X}}{B_l}$. Using (2), (3), (4) and (6), we get,

$$P_l = \frac{(p_l - \rho_l)(\rho_l + 1 - p_l)^{K_l}}{1 - (\rho_l + 1 - p_l)^{K_l+1}} \tag{8}$$

In summary, (8) represents the closed form expression of packet drop probability, P_l, on a single link l as a function of mean, variance λ_l, σ_l^2 of the arrival process, mean packet size $\bar{X}$, link bandwidth B_l and buffer space K_l. Figure 1 shows the drop probability as a function of the offered load for difference values of variance of the inter-arrival time for a buffer size of 20 packets. As expected, higher drop probability is observed when the arrival process has a high variance i.e. when the incoming traffic is more bursty.

III. Optimal Routing Problem

. In Section II, we had obtained the expression for packet drop probability given the arrival process, mean packet size and parameters associated with a link. Now let us consider a network of links with the matrices $\mathcal{D}$ and $\mathcal{V}$ denoting the mean and variance of the estimated demand. We now obtain expressions to compute the packet drop probability for a link $l \in \mathcal{L}$, given the demand parameters. We also formulate the optimal routing problem to minimize the total packets dropped in the network.

Using the expression for packet drop probability in (8), the objective function is given by

$$\min \Phi = \sum_{l \in \mathcal{L}} \lambda_l P_l \tag{9}$$

This is a constrained optimization problem with the flow constraints at each router j for each demand $\mathcal{D}(s,t)$ between source s and destination t. If $f_l^{(s,t)}$ denotes the fraction of the demand $\mathcal{D}(s,t)$ on link l, then the flow balance constraints are given by

$$\sum_{i:(i,j)\in\mathcal{L}} f_{(i,j)}^{(s,t)} - \sum_{i:(j,i)\in\mathcal{L}} f_{(j,i)}^{(s,t)} = \begin{cases} -\mathcal{D}(s,t) & \text{if } j = s \\ \mathcal{D}(s,t) & \text{if } j = t \\ 0 & \text{Otherwise} \end{cases} \tag{10}$$

The total mean packet arrival rate to a link l is given by

$$\lambda_l = \sum_{(s,t)\in\mathcal{N}\times\mathcal{N}} f_l^{(s,t)} \tag{11}$$

The parameter p for the GE process used to fit the demand $\mathcal{D}(s,t)$ is given by

$$p^{(s,t)} = \frac{2\mathcal{D}(s,t)^2}{\mathcal{D}(s,t)^2 + \mathcal{V}(s,t)} \tag{12}$$

Let $r_l^{(s,t)}$ denote the probability with which the demand $\mathcal{D}(s,t)$ is sent on link l. Then $r_l^{(s,t)}$ is given by

$$r_l^{(s,t)} = \frac{f_l^{(s,t)}}{\mathcal{D}(s,t)} \tag{13}$$

Let $p_l^{(s,t)}$ denote the parameter p of the GE process after splitting the demand $\mathcal{D}(s,t)$ with probability $r_l^{(s,t)}$. Then $p_l^{(s,t)}$ denotes the parameter p of the GE process representing the flow $f_l^{(s,t)}$. The parameter $p_l^{(s,t)}$ is given by

$$p_l^{(s,t)} = \frac{p^{(s,t)}}{p^{(s,t)}(1 - r_l^{(s,t)}) + r_l^{(s,t)}} \tag{14}$$

The total offered load on link l is given by λ_l (11), the parameter p of the associated GE distribution may be obtained by merging the flows $f_l^{(s,t)}$ going through l. If p_l

denotes the parameter p of the GE process associated with the aggregate traffic on link l, then p_l is given by

$$p_l = \lambda_l (\sum_{(s,t)\in\mathcal{N}\times\mathcal{N}} f_l^{(s,t)} p_l^{(s,t)})^{-1} \quad (15)$$

If ρ_l is equal to $\frac{\lambda_l p_l \bar{X}}{B_l}$, then, using (8), the probability of packet dropped at link l is given by

$$P_l = \frac{(p_l - \rho_l)(\rho_l + 1 - p_l)^{K_l}}{1 - (\rho_l + 1 - p_l)^{K_l+1}} \quad (16)$$

The optimal general routing problem is given by (9), subject to the constraints given by (11), (12), (13), (14), (15), (16). It may be noted that we are casting the traffic according to the routing in order to obtain the mean and variance of the total offered traffic to each $l \in \mathcal{L}$. However, we are not iterating to obtain the equilibrium traffic parameters. Essentially, we are using the upper bound on the packet drop probability in (9).

In order to obtain the value of Φ for a given OSPF weight setting, we run modified Floyd Warshall's algorithm (modified to obtain equal cost paths also) to obtain the routing. Then the traffic is cast to obtain parameters of the aggregate packet arrival process and drop probability for every link $l \in \mathcal{L}$ using (11), (12), (13), (14), (15) and (16).

IV. Dynamic Optimization of OSPF weights

The general optimal routing problem, defined by (9)-(16), can be solved for $f_l^{(s,t)} \forall l \in \mathcal{L}$ by using non-linear programming techniques. However, due to the shortest path nature of OSPF and the equal cost multi-path, finding link weight settings that minimize the packet drop probability given by (8) is a NP-hard problem. In [7], authors have proved the result for a linear objective function. It is straightforward to show, by proceeding along the same lines, that our problem is also NP-hard. Moreover, the objective function is unknown in the sense that we do not know the packet drop probability in terms of the OSPF link weight settings. Hence, we use a fast global search algorithm to find "good" OSPF link weight setting.

Dynamic optimization of OSPF weights demands fast search algorithms and automatic network management tools. Since the objective function is non-linear and multimodal, this is a global optimization problem, rather than a local optimization problem. However, when we consider the dynamic optimization problem, issues of scalability and number of iterations needed to obtain a "good" operating point gain foremost importance. In the next two sub-sections we describe our solution to the problem of dynamic global optimization problem of optimizing OSPF weights to minimize the packets dropped in the network. We use Online Simulation (OLS) framework for automatic network management. Section IV-A describes the OLS framework. In IV-B we discuss the issues governing the speed of search algorithms for large-dimensional parameter spaces and describe a new search algorithm for obtaining "good" parameter setting fast.

A. Online Simulation Framework

The online simulation architecture is an automatic network management framework. In particular, OLS does not interfere with the packet-by-packet data of the network. We term this as *"second-order" control* over network functions. The OLS architecture is mainly composed of autonomous online simulators which continuously monitor and model the network conditions and topology. Based upon the online model of traffic and topology, the simulators can execute simulations or other programs to evaluate the performance of the network for a given set of protocol parameters. For the problem of dynamic optimization of OSPF weights, the protocol parameters that we consider are link weights and the performance metric is the total packets dropped per second. However, online simulation framework may be used to "tune" parameters of any network protocol. The underlying assumption is that the performance is sensitive to the parameter settings and a "good" parameter setting depends on the network topology and current traffic conditions. [18] demonstrates the use of OLS to improve the end-to-end performance by "tuning" the parameters of RED and adaptive routing. The OLS system uses fast search schemes to obtain better parameter settings for the current traffic and topology conditions. In other words, the simulation system can support continuous "tuning" of the network based upon the online traffic modeling, parameter search and simulation capabilities. The online simulation scheme uses a best-effort parameter search strategy whose emphasis is not on "full" optimization, but on continuously and increasingly moving the system towards a "better" operating point. The OLS may be triggered either periodically or by a substantial change in the network load. However, a limit may be imposed on how frequently OLS is triggered for the optimization of the OSPF weights.

Figure 2 shows the OLS architecture for automatic network management. The reader may refer to [18] for more details on the OLS architecture.

Speed is an important issue for the effectiveness of OLS as it decides the response time of OLS. The OLS can only respond to changes in traffic which occur at a rate slower than the OLS response time. Hence, in some cases it may be important to obtain "good" results as soon as possible,

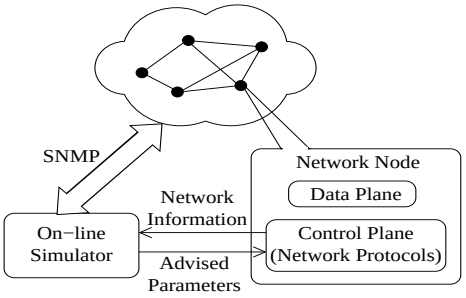


Fig. 2. Online Simulation architecture for automatic network management

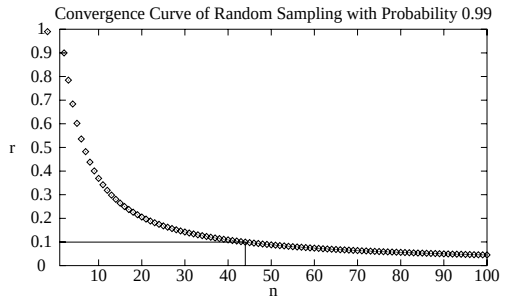


Fig. 3. Convergence curve of random sampling with probability 0.99

rather than to obtain "optimal" results after a long delay. The methods described above were designed to speed-up each experiment (iteration). However, we also need to minimize the number of iterations needed to find a good solution. Therefore, we designed a new random search algorithm. IV-B describes the new scheme and argues why random search schemes are more effective in optimization problems with large parameter space and analytically unknown objective function. We also analyze the convergence properties of the proposed adaptive random search scheme.

B. Adaptive Random Search Algorithm

As discussed earlier, the design goal of the search scheme is not to search for the optimum parameter setting, but to find a better parameter setting within a limited time frame. The high efficiency requirement is also due to the fact that the simulation of a complex network is often very time-consuming and therefore it is necessary to minimize the number of function evaluations. Second requirement is that the search algorithm must be able to handle a large number of parameters. This is because a network would often have a large parameter space, e.g. in our case, the size of the parameter space is same as the number of links in the network. Another issue that needs to be considered for designing a search algorithm is that the network simulation only provides us with approximate estimation of network performance. This means that the objective function in our problem is superimposed with small random noises due to inaccuracy in network modeling, simulation, etc.

In [6], [7], authors have proposed a local search scheme to optimize the OSPF weights. If vector $\bar{w}$ denotes the vector of link weights $w_l, \forall l \in \mathcal{L}$, then they find a neighbor $\bar{w}'$ of $\bar{w}$ by using one of the two operations. First operation is *single weight change*, where the link weight of only one link is changed to a value that has not been evaluated yet. Second operation is *evenly balancing flows*, where they try to generate equal cost paths in order to balance the load. More precisely, weights of a subset of the links leaving a router r are changed to create equal cost paths for some destination t. In order to avoid the search from *exploiting* a local minima, if a better solution is not obtained in 300 iterations, the current solution $\bar{w}$ is randomly perturbed uniformly between (-10%,10%) of maximum link weights. In their search method, tabu technique [1] has been used to prevent the search from revisiting the previous link weight setting. Basically, they use a hash table to store the setting already visited and make this table a tabu list to avoid revisits. However, the problem with using tabu lists with hashing is that some link weight setting which is good and has not been visited before may also be prevented from visiting. Therefore, in [7], authors have stated that the hashing method must be selected carefully to minimize the effect of this problem.

The above local search method has been used in [6], [7] to demonstrate that OSPF weight settings can be found which yield performance very close to the optimum achieved by the general optimal routing. However, this scheme cannot be used for the dynamic optimization. The main reason for this is that local search schemes are very inefficient for high-dimensional optimization problems [16], [10]. Also, they are susceptible to the effect of noise in the objective function. We have designed an adaptive random search scheme for use in the OLS architecture. Our algorithm is based on the high efficiency of random sampling at initial steps and it does not use any traditional local search method, such as pattern search, hill climbing, etc.. Therefore, it is more robust to noises in function evaluations and more scalable to high dimensional problems without sacrificing efficiency.

The basic idea of our algorithm is to first do a random sampling of the entire parameter space to identify a promising region, then start another random sampling in the promising region. The sample space is shrunk and realigned based on these samples. The shrink-and-re-align process continues until it finally converges to a local optimum. Then the whole process is repeated until the stop-

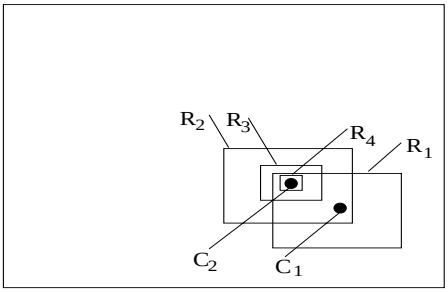


Fig. 4. Shrink and re-align procedure of Adaptive Random Search

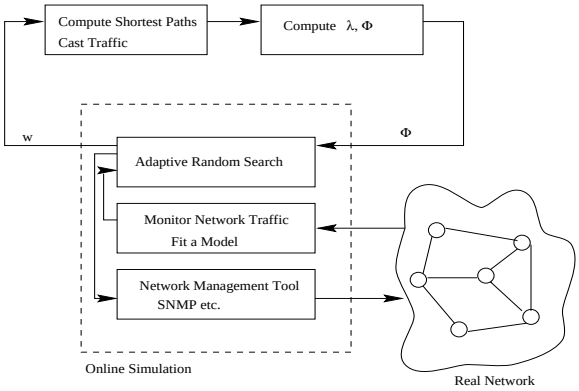


Fig. 5. Overall dynamic optimization setup using the Online Simulation architecture

ping criteria is satisfied. Random sampling is very efficient in identifying a good point close to the promising areas in its initial steps. Let $f(x)$ denote the objective function and D denote the parameter space. To illustrate the efficiency of random search in initial steps, we first define a measure $\phi_D(y_r)$ given by

$$\phi_D(y_r) = \frac{m(\{\, \mathbf{x} \in D \mid f(\mathbf{x}) \leq y_r \,\})}{m(D)} \tag{17}$$

where $m(\cdot)$ is *Lebesgue measure*. Then we can define a *r-percentile* region in the parameter space D:

$$A_D(r) = \{\, \mathbf{x} \in D \mid f(\mathbf{x}) \leq y_r \,\} \tag{18}$$

where

$$\phi_D(y_r) = r, \quad r \in [0, 1]$$

y_r is called *r-percentile* of the objective function value. Note that $A_D(1)$ is just the whole parameter space D and $\lim_{\epsilon \to 0} A_D(\epsilon)$ will converge to the global optimum. Suppose the sample sequence generated by n steps of random sampling is given by $\mathbf{x}_i, i = 1 \ldots n$ and $\mathbf{x}^n_{(1)} = \arg\min_{1 \leq i \leq n} f(\mathbf{x}_i)$, then the probability of $\mathbf{x}^n_{(1)}$ in $A_D(r)$ is:

$$P(\mathbf{x}^n_{(1)} \in A_D(r)) = 1 - (1 - r)^n \tag{19}$$

We can have the value of r indicating the *r-percentile* region that $\mathbf{x}^n_{(1)}$ will reach with probability p

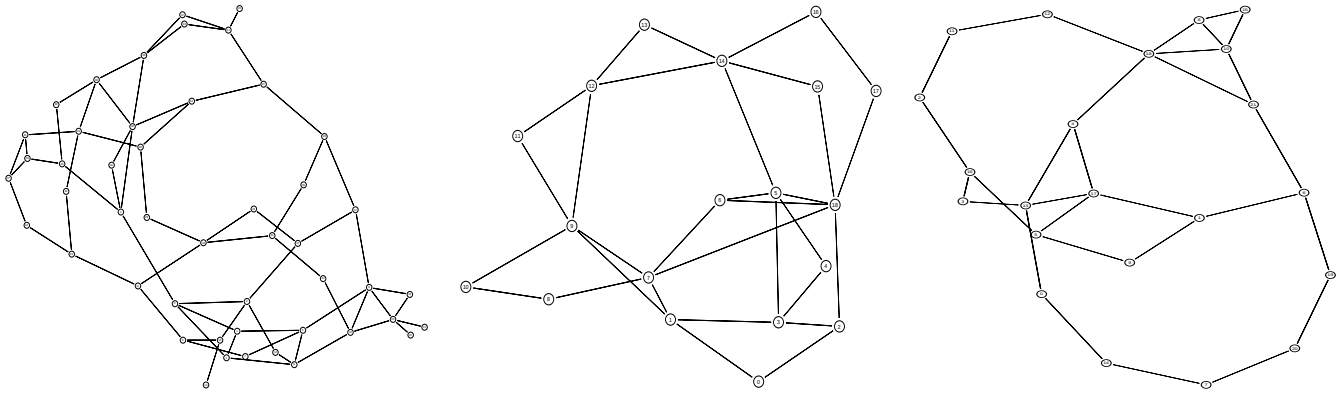
$$r = 1 - (1 - p)^{1/n} \tag{20}$$

For any probability $p < 1$, r will tend to 0 with increasing n. This is the global convergence guarantee of random sampling scheme. From (20), we can also see that random sampling is highly efficient at initial steps since r decreases exponentially with increasing n, and its inefficiency is from later samples. Figure 3 shows the convergence curve of random sampling with a probability of 0.99. We can see that it takes only 44 samples to reach a point in $A_D(0.1)$ area whereas all samples after the first 44 can only improve r value of $\mathbf{x}^n_{(1)}$ at most by 0.1. We just make use of these first part of high-efficiency samples to identify promising areas and shrink and re-align sample space.

The complete search process of the adaptive random search scheme has been illustrated in Figure 4. First we take a number, say n, of random samples from the parameter space S, and take the best point as the center C_1 of the promising region R_1 which is further explored. The point C_1 will fall in $A_D(r), r = 1 - (1 - p)^{1/n}$ with probability p. The size of the promising region is taken to be the size of $A_D(r)$ so as to to cover at least one local optimum in $A_D(r)$ with a high probability. Then we take another l random samples from R_1. Here l should be much less than n since now we are in a promising area and expecting to find better points quickly. If we find a better point within these l samples, we move the center of the sample space to this point and keep the size unchanged. As shown in Figure 4, we move the center to C_2, use region R_2 as the next sample space. If we cannot find any better point in l samples, it suggests that the center is close to the local optimum and most points in the region are not as good as the center. Therefore we reduce the size of sample space by half and keep the center unchanged to explore the neighborhood of the center more carefully. As shown in Figure 4, we use R_3 as next sample space after l unsuccessful samples in R_2, and keep the center C_2 unchanged. This shrink-and-re-align procedure is repeated until the size of the region is reduced below a threshold, then we restart the above search process.

V. Simulation Results

Figure 5 shows the functional block diagram of the overall dynamic optimization setup using OLS. The OLS monitors the traffic and models it to provide mean and variance of the demand estimates to the adaptive random search. Adaptive random search obtains a link weight setting for



(a) ARPANET Topology (b) MCI Topology (c) Randomly Generated Topology

Fig. 6. Figure showing the network topologies used in simulation

all links in the network, the performance of this link weight setting is computed using the analytic results obtained in section II. Based on the packet drop rate obtained from the analysis, the search scheme comes up with another link weight setting. This process is repeated until the stopping criteria is met or a timeout occurs. This timer may be used to put an upper bound on the response time of the OLS. The new link weight settings may be deployed in the real network only if it results in substantial improvement in the performance. It may be noted that we do not use packet-level or flow level simulation to estimate the packet loss rate in the network. Instead we use an analytic approach, outlined in Section III, that is considerably faster. Also, our search scheme has been designed to be robust to the noise in the objective function.

In this section we present two sets of simulation results. One is to demonstrate that the adaptive random search scheme obtains better OSPF link weight settings in fewer iterations than the algorithm proposed in [6]. Another set of results demonstrate the improvement in end-to-end performance (in terms of the loss rate) by dynamic optimization of OSPF weights.

We have considered three network topologies, shown in Figure 6, to demonstrate our results. Two are well-known ARPANET topology and MCI topology. The ARPANET topology consists of 48 routers and 140 simplex links Each link in the network is assumed to consist of two simplex link whose weights may be set independently. MCI topology consists of 19 routers and 62 simplex links. We have also considered a randomly generated topology with 22 routers and 60 simplex links.

Random amount of traffic was sent from every node to every other node in the network. This random traffic was generated using the method outlined in [6]. For each node u, two random numbers are generated O_u, $D_u \in [0, 1]$. For each pair of nodes (u, v) another random number $C_{(u,v)} \in [0, 1]$ was generated. If Δ denotes the largest Eucledian distance between any pair of nodes and if α denotes a constant, the average demand between u and v is given by

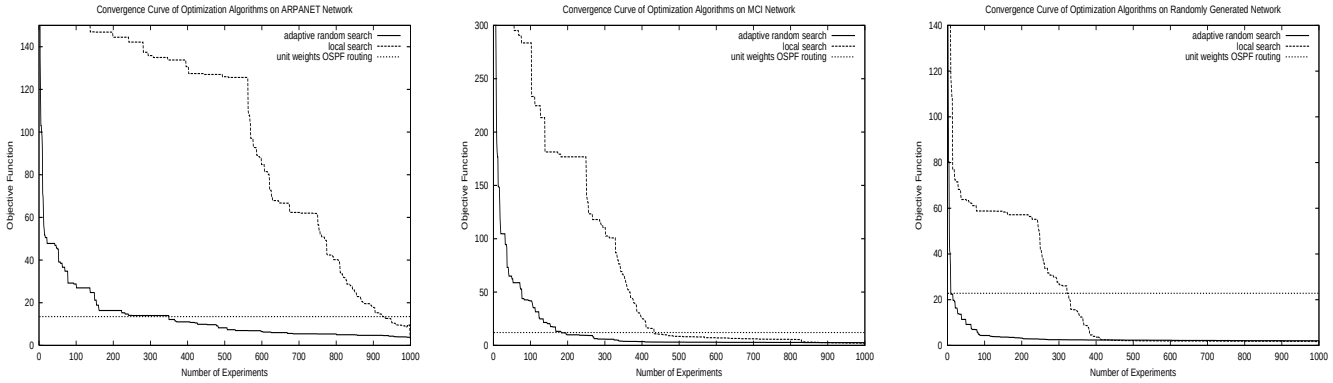
$$\mathcal{D}(u, v) = \alpha O_u D_v C_{(u,v)} e^{\frac{-\delta(u,v)}{2\Delta}}$$

where, $\delta(u, v)$ denotes the Eucledian distance between the nodes u and v. This method of generating random traffic (the term $e^{\frac{-\delta(u,v)}{2\Delta}}$) ensures more traffic for source destination pairs that are closer to each other. Since a product of three random variables is taken to generate the demands, there is actually a large variation in the traffic demands. The ratio of square of mean to the variance was assumed to be a uniformly distributed random variable in $[0, 1]$. The mean and variance of the traffic demands are generated using the above procedure. A traffic generator was implemented over UDP, in *ns*, to generate bursty traffic with exponentially distributed mean burst inter-arrival time and geometric burst size distribution. Essentially, the traffic was generated at according to the GE arrival process with a given mean and variance. All the links in the network have 1Mbps bandwidth with a buffer size of 50 packets. The packet size was chosen to be exponentially distributed with mean packet size of 200 bytes.

In the simulation results presented in this paper, we do not verify the traffic modeling assumptions as this is not a focus of this paper. The performance results shown in V-A are the average results from ten simulation runs. Average of multiple simulation runs is presented as we compare the performance of two stochastic search algorithms.

A. Comparison of convergence

In this section, we present the results of comparison of adaptive random search scheme with the local search scheme proposed in [6], [7]. The comparison is done in



(a) (b) (c)

Fig. 7. Figure showing the piece-wise linear metric as a function of number of iterations for the (a) ARPANET (b) MCI (c) Randomly generated network topology

terms of the number of iterations because, as noticed earlier, the main computation time is to evaluate the optimization metric for a given link weight setting. Our assumption was that the computation time per iteration is approximately the same for both the schemes. This is not exactly true because in [6], authors have used lazy shortest path computations to improve the speed of search as very few link weights change from one iteration to the next in their local search scheme. However, adaptive local search is a general search scheme for large dimension parameter spaces. Hence, we cannot utilize the knowledge that faster function computations can be achieved by changing only a few link weights. In [7] authors have reported 15% improvements on an average by using the lazy Dijkstra's algorithm. However, their search scheme needs information about path weights to a destination for the *evenly balancing flows* step. This may need additional computational depending on the implementation. Hence, we do not compare the two search schemes in terms of the absolute time taken to find a "good" parameter setting. Instead, we compare the two schemes in terms of the number of iterations needed. Loosely, we refer to the number of iterations required to obtain a "good" parameter setting as the speed of convergence (to the "good" parameter setting). A 'good" parameter setting has been defined earlier as the OSPF link weight setting that give metric value lower than that by setting all link weights equal to unity (called unit OSPF). This definition is just for the purpose of comparison. A "good" parameter setting may have been defined alternatively as the link weight setting to achieve performance metric equal to, say, 80% of the unit OSPF.

A.1 Heuristic Piece-Wise Linear Metric

In order to compare the speed of convergence of our search scheme with the local search scheme proposed in [6], we use the metric used in [6]. In this subsection, we present the comparison results for the piece-wise linear optimization metric used in [6]. Figure 8 shows the link cost as a function of offered load, as used in [6]. The optimization function is to minimize the sum of link costs, summed over all $l \in \mathcal{L}$.

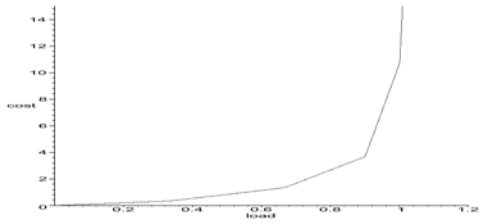
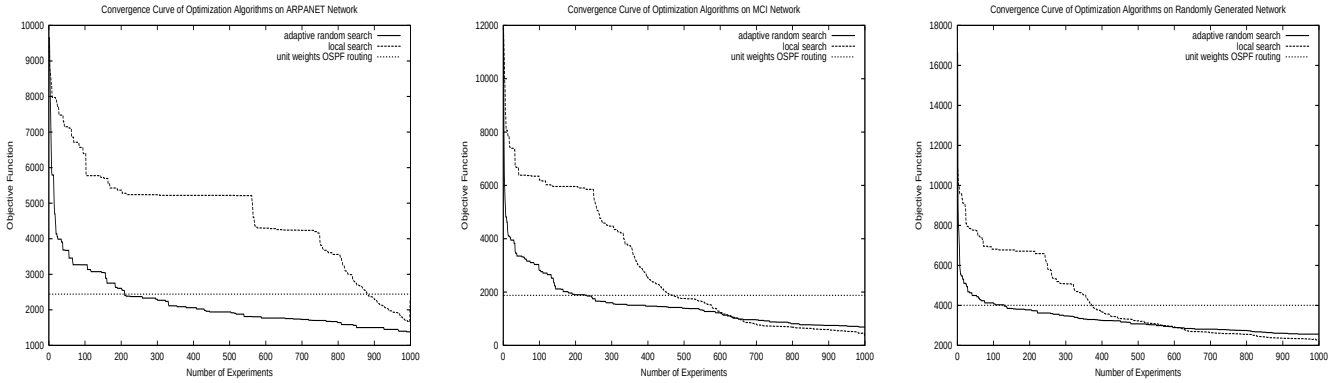


Fig. 8. Figure showing the link cost as a function of offered load

Figure 7 shows the piece-wise linear optimization metric value as a function of iteration number for the ARPANET, MCI and Randomly generated network topologies respectively. For the sake of comparison, these graphs also show the optimization metric value when all the links' weights are set to unity. Figure 7 shows that the adaptive random search scheme out-performs the local search scheme in terms of the number of iterations needed to find a "good" parameter setting for all the three network topologies. These results have been tabulated in Table I.

It may also be noted that maximum improvement is observed for the case of ARPANET topology. The reason for this, we believe, is that ARPANET has the maximum number of links among the three network topologies for which the results have been demonstrated. The gain by using adaptive random search increases with the number of dimensions of the search space (number of links in the network).



(a) (b) (c)

Fig. 9. Figure showing total packet drop rate as a function of number of iterations for the (a) ARPANET (b) MCI (c) Randomly generated network topology

Scheme	ARPANET	MCI	Random
Local Search	932	433	322
ARS	350	183	9
Improvement	62.4%	57.7%	97.2%

TABLE I

TABLE COMPARING THE NUMBER OF ITERATIONS NEEDED TO OBTAIN A "GOOD" PARAMETER SETTING FOR PIECE-WISE LINEAR OBJECTIVE FUNCTION

A.2 Packet Loss Rate Metric

In this section we present the comparative results for the packet loss metric (16). Figure 9 shows the comparison results of metric value as a function of iteration number. The results clearly show that the adaptive random algorithm significantly out-performs the local search algorithm. Table II shows that for the packet drop rate metric also, our adaptive random search scheme took 50% or fewer iterations to obtain a "good" OSPF link weight setting. Again, more improvement is observed for the ARPANET topology due to the large size of the search space.

Scheme	ARPANET	MCI	Random
Local Search	882	469	372
ARS	210	232	130
Improvement	76.1%	50.5%	64.5%

TABLE II

TABLE COMPARING THE NUMBER OF ITERATIONS NEEDED TO OBTAIN A "GOOD" PARAMETER SETTING PACKET DROP RATE METRIC

B. Impact of dynamic optimization on end-to-end performance

Figure 10 shows total packet drop rate in the network as a function of time. The traffic was generated in the same way as outlined in the beginning of section V. However, every 200 seconds the traffic pattern was changed in order to create a dynamic scenario. The results of the adaptive random search were deployed after 100seconds of the traffic change. This was done to clearly demonstrate the improvement in the performance by dynamic optimization of OSPF weights using OLS. In these results, the traffic conditions are assumed to be known to the OLS. Table III summarizes the maximum improvement in packet loss rates for different topologies. These results are only for the results presented in Figure 10. More improvements or less improvements may result depending on the topology and traffic conditions.

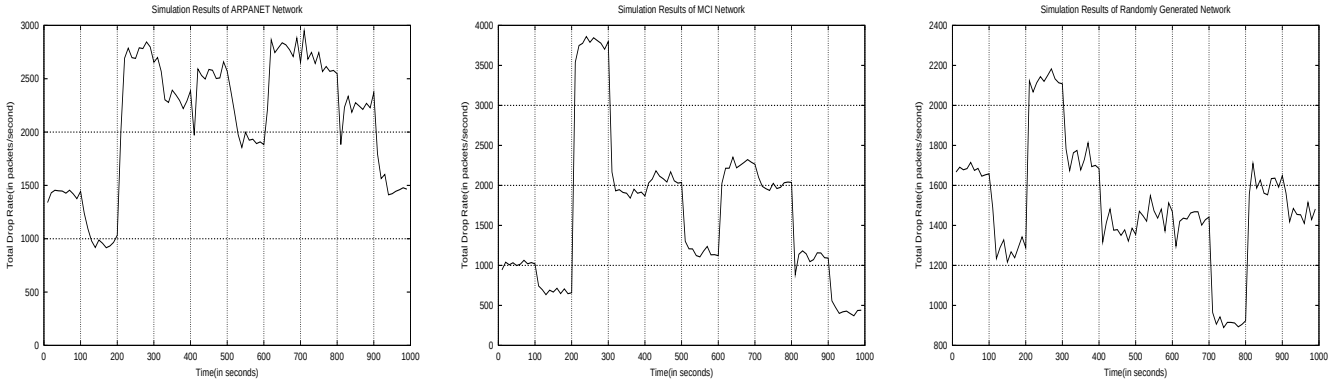
	ARPANET	MCI	Random
Max. Improvement	31.8%	60.2%	35.7%

TABLE III

TABLE SUMMARIZING THE MAXIMUM PERCENTAGE IMPROVEMENT IN THE PACKET LOSS RATES OBTAINED FOR DIFFERENT TOPOLOGIES FOR THE RESULTS SHOWN IN FIGURE 10

VI. CONCLUSIONS AND FUTURE WORK

In this paper we have solved some of the problems associated with the dynamic optimization of OSPF weights. Total packet drop rate in the network was analytically computed using GI/M/1/K queueing model. Dynamic optimization problem was formulated where total packet drop rate was chosen as the optimization criteria. Fast adap-



(a) (b) (c)

Fig. 10. Figure showing total packet drop rate as a function time for the (a) ARPANET (b) MCI (c) Randomly generated network topology. Traffic pattern was changed at times 0, 200, 400..., the optimized OSPF weights were deployed at times 100, 300,...

tive random search algorithm was developed to address the issue of speed associated with the dynamic optimization. Performance results demonstrate that our search algorithm took 50-90% fewer iterations as compared to the local search algorithm of [6]. Our simulation results also demonstrate improvements of the order of 30-60% in the total loss rate in the network. Future work includes demonstration of dynamic optimization of OSPF weights in a real network. To investigate the issues associated with traffic monitoring and modeling and its impact on the performance of dynamic optimization can be another goal for future work.

REFERENCES

[1] R. Battiti, G. Tecchiollo, "The reactive Tabu Search", *ORSA Journal on Computing*, Vol. 6, No. 2, pp. 126-140, 1994.

[2] Dimitri P. Bertsekas, "Dynamic Models of Shortest Path Routing Algorithms for Communication Networks with Multiple Destinations," *Proceedings of 1979 IEEE Conference on Decision and Control*, Ft. Lauderdale, FL, pp. 127-133, Dec. 1979.

[3] Cisco::Netflow, *http://cisco.com/warp/public/732/netflow/index.html*

[4] Robert B. Cooper, "Introduction to Queueing Theory," Second Ed. New York : North Holland, 1981.

[5] Anja Feldmann, Albert G. Greenberg, Carsten Lund, Nick Reingold, Jennifer Rexford and Fred True, "Deriving traffic demands for operational IP networks: methodology and experience", In *Proceedings of SIGCOMM*, pp. 257-270, 2000.

[6] Bernard Fortz and Mikkel Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights, in *Proceedings of the INFOCOM 2000*, pp. 519-528, 2000.

[7] Bernard Fortz and Mikkel Thorup, "Increasing Internet Capacity Using Local Search," Preprint, *http://smg.ulb.ac.be/Preprints/Fortz00_21.html*, 2000.

[8] David W. Glazer and Carl Tropper, "A New Metric for Dynamic Routing Algorithms," *IEEE Transactions on Communications*, Vol. 38 No.3 March 1990.

[9] S. Handelman, S. Stibler, N. Brownlee, G. Ruth, "RTFM: New attributes for traffic flow management", *Request for Comments-2724*, October 1999.

[10] A. H. Kan and G. T. Timmer, "Stochastic Global Optimization Methods Part I: Clustering Methods," *Mathematical Programming*, Vol. 39, pp. 27-78, 1987.

[11] Atul Khanna and John Zinky, "The Revised ARPANET Routing Metric," *Proceedings of the ACM SIGCOMM*, pp. 45-56, 1989.

[12] Ramesh Nagarajan, James F. Kurose, and Don Towsley, "Approximation techniques for computing packet loss in finite-buffered voice multiplexers," IEEE J.Select.Areas Commun. , 9(3):368–377, April 1991.

[13] D. S. Lee, G. Ramamurthy, A. Sakamoto and B. Sengupta "Performance analysis of a threshold-based dynamic routing algorithm," *Proceedings of the Fourteenth International Teletraffic Congress*, ITC-14, 1994.

[14] J. Moy, "OSPF Version 2," RFC 2178, April 1998.

[15] Harry G. Perros Queueing Networks With Blocking, Exact and Approximate Solutions, *Oxford University Press*, 1994.

[16] Soraya Rana, L. Darrell Whitley and Ronald Cogswell, "Searching in the Presence of Noise", Parallel Problem Solving from Nature – PPSN-IV (Berlin, 1996), *Lecture Notes in Computer Science 1141*, Editors–H. Voigt, W. Ebeling, I. Rechenberg and Hans-Paul Schwefel, Springer, pp. 198-207, 1996.

[17] Zheng Wang, Jon Crowcroft, "Analysis of Shortest-Path Routing Algorithms in a Dynamic Network Environment," *Computer Communication Review*, Vol. 22, no. 2, pp.63-71, 1992.

[18] T. Ye, D. Harrison, B. Mo, B. Sikdar, H. T. Kaur, S. Kalyanaraman, B. Szymanski and K. S. Vastola, "Network Management and Control Using Collaborative On-line Simulation," *In Proceedings of IEEE ICC*, Helsinki, Finland, June 2001.

[19] M. Yuksel, B. Sikdar, K. S. Vastola and B. Szymanski, "Workload generation for ns Simulations of Wide Area Networks and the Internet," *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, pp 93-98, San Diego, CA, USA, 2000.