Optimization of OSPF Weights Using On-line 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 present an OSPF weights optimization scheme using a general automatic network management framework proposed in [1], i.e., on-line simulation framework. We have chosen the packet drop 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 drop 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 queuing model has been used to compute the packet drop probability on a given link. We have also developed a fast recursive random search algorithm to address the issues associated with network optimization problems. The search algorithm has been compared to the local search heuristic of [4] in terms of the number of function evaluations needed to obtain a "good" OSPF link weight setting. Our results demonstrate that the recursive random search takes 50-90% fewer function evaluations to find a "good" setting. The amount of improvement depends on the network topology, traffic conditions and optimization metric. We have simulated the proposed OSPF optimization scheme in ns[19] and the results indicated improvements of the order of 30-60% in the total packet drop rate.

Index Terms

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 to meet the objectives of network operators. Two main approaches have been taken to solve the traffic engineering problem in the Internet. One 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. Another approach is to adjust the link weights of the existing network (running OSPF) such that the OSPF routing with these link weights leads to desired routes [4].

The main issue with using existing OSPF routing for traffic engineering is its shortest path nature. 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 [5]. 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], [6], [13]). This is called adaptive routing or traffic-sensitive routing. However, adapting link weights to local traffic conditions leads to frequent route changes and is unstable (see [2], [17] for stability analysis). Additionally, adaptive routing is based on the local information and therefore cannot optimize traffic allocation from the viewpoint of the overall network. These drawbacks are alleviated in [4] where the traffic demand of the network is used to estimate the offered load for each link and then a local search heuristic is deployed to find “good” OSPF link weight settings which optimize the traffic load allocation across the network.

Basically, authors in [4] have modeled the optimum setting of OSPF weights as a global optimization problem. They have chosen a heuristic cost function which is piecewise linear with offered load. By using such a cost function, they can model the optimal general routing as a linear programming problem and solve for the exact solution. Here the optimal general routing represents routing where there is no limitation on the way a flow is split among multiple paths available between a source and destination. The optimal general routing is the best that can be achieved by carefully setting up multiple Label Switched Paths (LSPs) in MPLS. Authors in [4] have shown that for the proposed AT&T WorldNet backbone, OSPF with optimized link weights can yield a routing with a performance within few percent of the optimal general routing and even for the randomly generated network topologies, 50%-110% more demand can be supported than link weight setting based on some standard heuristic.

Essentially, [4] has demonstrated the conception of optimization of OSPF weights for balancing the traffic. In fact, as long as the performance of a network protocol is sensitive to its parameter setting, we can always optimize it for a certain performance objective. In [1], we have proposed an on-line simulation framework(OLS) for general automatic network management by tuning the parameters of network protocols. The basic idea is illustrated in Figure 1. The on-line simulation architecture continuously collects concerned network information, such as, traffic conditions and topology. Based upon this information, the OLS can evaluate the network performance for a given set of protocol parameters and use a search algorithm to optimize its parameter setting to current network conditions. The underlying assumption is that the network is quasi-stationary, i.e., the network conditions vary on longer time scales as compared to the time required for optimization.

In the OLS scheme, a "black-box" approach has been adopted, i.e., the optimization of network protocols is modeled as a general "black box" problem where the objective function is unknown but can be evaluated through simulations. The advantage of this approach is that it makes the OLS a very *flexible* system whose use is *not restricted in one specific protocol or one performance objective*.

In [1], we have demonstrates the use of OLS to optimize the parameters of RED. In this paper, we will apply it to the optimization of OSPF link weights. We have chosen the total packet drop rate in the network as the optimization metric since it is a more accurate to indicate the congestion in the network than

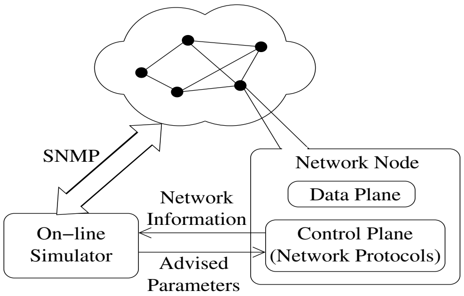


Fig. 1. On-line simulation architecture for automatic network management

the heuristic metric in [4] and it also has great impacts on the performance of some underlying protocols, such as TCP. The packet drop rate for one set of link weights can be estimated using packet-level or flow level simulation. However, in this paper, we use an analytic approach to calculate the packet drop rate by using a GI/M/1/K queuing model. This is considerably faster than the simulation approach. Also, we have developed a fast recursive random search algorithm(RRS) for "black-box" optimization problems with large number of parameters. As pointed out in [10], [16], local search schemes are lacking in efficiency for high-dimensional optimization problems and are also susceptible to noises in the objective function since they are biased too much on the local features of the objective function. Without using any traditional local search methods, our search scheme is more suitable than the local heuristic proposed in [4] for network optimization, where there may exist hundreds of parameters and inaccuracies in measurement and modeling may also introduce random noises into the objective function. Our simulation results demonstrated that that recursive random search can find a "good" parameter setting much faster. We have also simulated the overall OSPF optimization scheme with ns simulator[19] and found that the network packet drop rate can be lowered significantly.

The rest of the paper is organized as follows. Section II derives the link packet drop rate from the offered load and formulates our optimization problem. Section III describes our approach of using on-line simulation framework for OSPF optimization. Section IV presents the simulation results and finally, Section V presents the conclusions and future work.

II. THE OBJECTIVE FUNCTION

Our goal is to minimize the packet drop rate 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 demand. In practice, all such information can be obtained using the tools described in [8], [9].

In the following, we will first show how to derive the drop probability for one link based on the offered load. Then we will formulate the optimal general routing problem which aims to optimize the overall packet drop rate for the network. Note that the OSPF optimization problem is just the optimal general routing subject to the shortest path constraint.

A. Link Drop Probability

Let P denote the packet drop probability on a link, λ, σ^2 denote the mean, variance of the offered load to this link in packets per second, and B, K denote its bandwidth and buffer space respectively. In order to find a closed-form expression for the packet drop probability P, 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 the link using a GI/M/1/K queuing 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.

$$P(N_K = K) = \frac{P(N_\infty = K)}{P(N_\infty \leq K)} \qquad (1)$$

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 [3]:

$$P(N_\infty = j) = A\omega^{j-1} \qquad (j \geq 0) \tag{2}$$

where A is the normalization constant and ω is a constant depending on the arrival process and service rate. ω can be obtained by solving the following equation:

$$\omega = \gamma\left((1-\omega)\mu\right) \tag{3}$$

where $\gamma(s)$ is the Laplace transform of the arrival process and μ is the service rate which is given by $\frac{B}{X}$. In order to solve (3) for ω, we need to assume a inter-arrival time 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

$$g(x) = (1-p)\delta(x) + pae^{-ax} \tag{4}$$

where $\delta(x)$ is the delta function, p and a two constant parameters. As can be seen from (4), 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 than 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 with λ, σ as its mean and variance of the offered load, we can have the parameters of the GE distribution representing the arrival process:

$$p = \frac{2\lambda^2}{\sigma^2 + \lambda^2} \text{ and } a = p\lambda \tag{5}$$

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 Bernoulli filter $r_1, r_2, ... r_N$, the parameters of the i^{th} process are

$$p_i = \frac{p}{p(1-r_i)+r_i} \text{ and } a_i = r_i a. \tag{6}$$

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.

Taking the Laplace transform of (4), we get,

$$G(s) = 1 - p + \frac{pa}{s+a} \tag{7}$$

Then substitute it into (3) and solve it for ω for the GE arrival process gives

$$\omega = \rho + (1-p) \tag{8}$$

where,

$$\rho = \frac{a}{\mu} = \frac{a\bar{X}}{B}. \tag{9}$$

Finally, using (1), (2), (3) and (7), we get the packet drop probability

$$P = \frac{(p-\rho)(\rho+1-p)^K}{1-(\rho+1-p)^{K+1}} \tag{10}$$

In summary, (10) represents the closed form expression of packet drop probability, P, on a single link as a function of mean, variance λ, σ^2 of the arrival process, mean packet size $\bar{X}$, link bandwidth B and buffer space K. Figure 2 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.

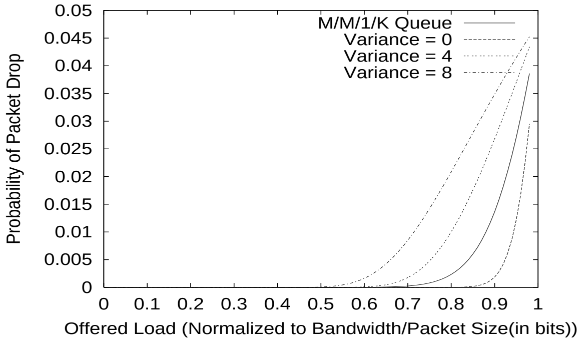


Fig. 2. Packet drop probability as a function of offered load for a GE/M/1/20 queue for different values of variance

B. The Optimal General Routing

Using link packet drop probabilities obtained from (10), we can formulate the optimal general routing problem as:

$$\Phi = \sum_{l \in \mathcal{L}} \lambda_l P_l \tag{11}$$

where λ_l is the arrival rate for link l and P_l is its drop rate calculated by (10). 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

$$\sum_{i:(i,j)\in\mathcal{L}} f_{(i,j)}^{(s,t)} - \sum_{i:(j,i)\in\mathcal{L}} f_{(j,i)}^{(s,t)} = \begin{cases} -\mathcal{D}(s,t) & \text{if } j = s \\ \mathcal{D}(s,t) & \text{if } j = t \\ 0 & \text{Otherwise} \end{cases} \tag{12}$$

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

$$\lambda_l = \sum_{(s,t)\in\mathcal{N}\times\mathcal{N}} f_l^{(s,t)} \tag{13}$$

The parameter $p^{(s,t)}$ for the GE process used to fit the demand $\mathcal{D}(s,t)$ is given according to (5):

$$p^{(s,t)} = \frac{2\mathcal{D}(s,t)^2}{\mathcal{D}(s,t)^2 + \mathcal{V}(s,t)} \tag{14}$$

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

$$r_l^{(s,t)} = \frac{f_l^{(s,t)}}{\mathcal{D}(s,t)} \tag{15}$$

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 according to (6):

$$p_l^{(s,t)} = \frac{p^{(s,t)}}{p^{(s,t)}(1 - r_l^{(s,t)}) + r_l^{(s,t)}} \tag{16}$$

The total offered load on link l is given by λ_l (13), 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

$$p_l = \lambda_l \Big(\sum_{(s,t) \in \mathcal{N} \times \mathcal{N}} f_l^{(s,t)} p_l^{(s,t)} \Big)^{-1} \tag{17}$$

If ρ_l is equal to $\frac{\lambda_l p_l \bar{X}}{B_l}$, then, using (10), the probability of packet dropped at link l is given by

$$P_l = \frac{(p_l - \rho_l)(\rho_l + 1 - p_l)^{K_l}}{1 - (\rho_l + 1 - p_l)^{K_l + 1}} \tag{18}$$

The optimal general routing problem is given by (11), subject to the constraints given by (13), (14), (15), (16), (17), (18). 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 (11).

III. Optimization of OSPF weights Using On-line Simulation

The general optimal routing problem, where the objective function is completely defined by (11)-(18), may possibly be solved for $f_l^{(s,t)} \forall l \in \mathcal{L}$ by using some non-linear programming techniques. However, under constraints of OSPF routing, the relation between the link weights and optimization metric can no longer be analytically defined. Hence, the optimal routing in OSPF becomes a “black box” optimization problem which may be defined as:

$$\min \Phi(\mathbf{w}) \tag{19}$$

where $\mathbf{w}$ is the vector of network link weights and $\Phi(\cdot)$ the objective function, which is unknown. Basically, 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 (13), (14), (15), (16), (17) and (18). Finally the value of Φ may be calculated by (11). In [5], authors have proved that it is NP-hard to find OSPF link weight settings for an optimization metric piecewise linear in offered load. It is straightforward to show, by proceeding along the same lines, that our problem, i.e., minimize the packet drop rate given by (11) is also NP-hard.

For such NP-hard problems, heuristic optimization algorithms are usually used to search for approximate solution. In the context of network optimization, we need a highly efficient search algorithm to find "good" OSPF link weight setting quickly since the network is a dynamic system and network conditions may change significantly from time to time. In other words, our emphasis is not on full optimization but *obtaining a practically usable solution within a limited time frame*. Furthermore the search algorithm should be *scalable to high-dimensional problems* since there may be hundreds of parameters in a network. Another issue that needs to be considered is that 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 inaccuracies in network modeling, simulation, etc. To address these issues, we have designed a recursive random search scheme. The details of our algorithm are presented in a separate paper[18]. We summarize the algorithm here for the purpose of completeness.

Basically, the algorithm is based on the high-efficiency feature of random sampling at initial steps. To illustrate this property, we first define a goodness measure $m_D(\phi)$ for a given function value ϕ when given an objective function as (19) and a parameter space D:

$$m_D(\phi) = \frac{V(\{\, \mathbf{w} \in D \mid \Phi(\mathbf{w}) \leq \phi \,\})}{V(D)} \tag{20}$$

where $V(\cdot)$ is *Lebesgue measure* of a set. Note that $m_D(\phi)$ is a monotonously increasing function with ϕ. When ϕ approaches the minimum function value, $m_D(\phi)$ tends to 0. With a probability p, n random samples

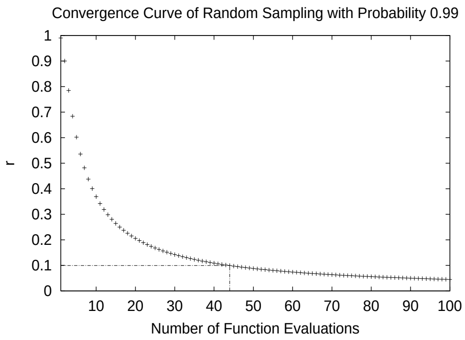


Fig. 3. Convergence curve of random sampling with probability 0.99

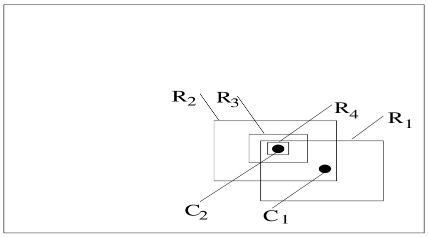


Fig. 4. Shrink and re-align procedure of Recursive Random Search

can find a solution whose m_D is:

$$1 - (1 - p)^{1/n} \tag{21}$$

For any probability $p < 1$, the above value will tend to 0 with increasing n. This is the global convergence property of random sampling. Figure 3 shows a convergence curve of random sampling with a probability of 0.99. We can see that at initial iterations of random sampling, the m_D of the solution decreases rapidly, and the inefficiency of random sampling is from later samples. For example, in Figure 3 it only takes only 44 random samples to improve the m_D of the solution from 0.99 to 0.1 whereas all samples after that can only improve it at most by 0.1. The basic idea of our algorithm is to use the first part of high-efficiency samples to identify promising areas then start recursive random sampling processes in these areas which shrink and re-align the sample space to local optima.

An example search process of RRS is illustrated in Figure 4. First we take a number, say n, of random samples from the parameter space D, and take the best point as the center C_1 of the promising region R_1 which will be further explored. The size of R_1 is taken to be the m_D value of C_1. 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.

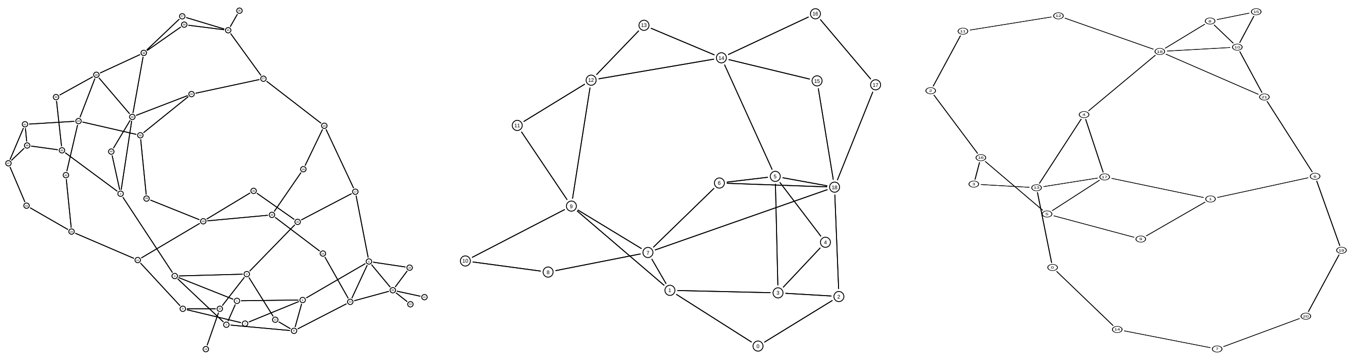
We have tested this algorithm on a suite of difficult benchmark functions. The results have shown that our algorithm is consistently more efficient than other search algorithms using traditional local search techniques, such as multi-start hillclimbing. The reader may refer to [18] for more details and results.

IV. Simulation Results

In this section we present two sets of simulation results. One is to demonstrate that the recursive random search scheme obtains better OSPF link weight settings with fewer function evaluations than the algorithm proposed in [4]. Another set of results demonstrate the improvement in end-to-end performance (in terms of the drop rate) by dynamic optimization of OSPF weights.

We have considered three network topologies, shown in Figure 5, to demonstrate our results. Two are well-known ARPANET topology and MCI topology. We couldn't include AT&T topology used in [4] since it is not publicly available. 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 [4]. For each node u, two random numbers are generated



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

Fig. 5. Figure showing the network topologies used in simulation

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. 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 IV-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 Search Schemes

In this section, we present the results of comparison of recursive random search scheme with the local search scheme proposed in [4]. In optimization literatures, the comparison between algorithms is usually done in terms of the number of function evaluation instead of the absolute time taken to find a "good" parameter setting. This is because the computation time is considerably dependent on many other factors, such as, implementation efficiency, testing platform, etc.. Considering the main computation time is for function evaluations, the number of function evaluation is a more appropriate performance metric under the assumption that the computation time per function evaluation is approximately the same for both schemes. Note this assumption is not exactly true in the context of our problem, where one function evaluation represents one optimization metric computation for a specific set of link weights. In [4], authors have used incremental shortest path computations to improve the speed of search as very few link weights change from one iteration to the next which is reported to have 15% improvements on an average. In spite of this, we still use the number of function evaluations as our algorithm performance metric for the reasons mentioned above and the consideration that our algorithm is designed to be a general "black-box" search algorithm where no problem-specific is available. It should be noted that even if taking 15% improvement for the local search scheme of [5] into consideration, the test results still show that our algorithm is significantly faster.

Loosely, we refer to the number of function evaluations required to obtain a "good" parameter setting as the speed of convergence. A "good" parameter setting has been defined 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.

1) Heuristic Piecewise Linear Metric: In order to compare the speed of convergence of our search scheme with the local search scheme proposed in [4], we use the same metric used in [4], which is piecewise linear with the link offered load. Figure 6 shows the cost for one link as a function of offered load. The optimization objective is to minimize the sum of link costs, summed over all $l \in \mathcal{L}$.

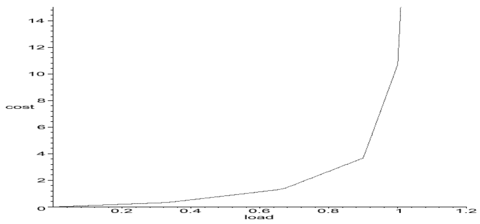
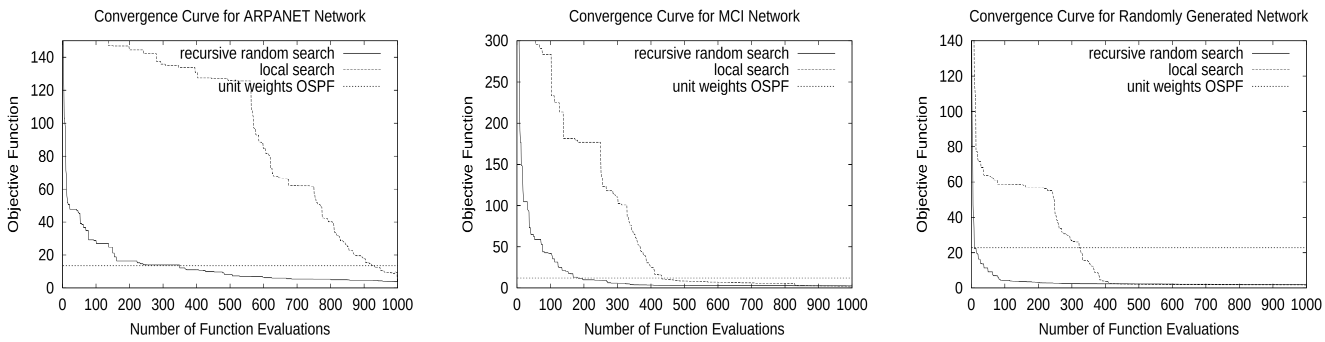


Fig. 6. Figure showing the link cost as a function of offered load

Figure 7 shows the optimization convergence curves 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. It can be seen that the recursive random search scheme outperforms the local search scheme in terms of the number of function evaluations needed to find a "good" parameter setting for all three network topologies. These results have been tabulated in Table I.



(a) (b) (c)

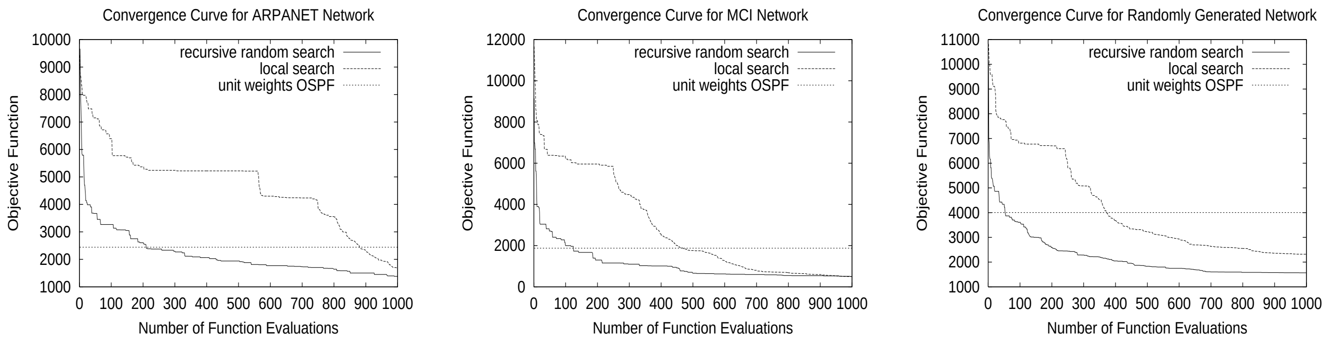
Fig. 7. Figure showing the convergence curves of piecewise linear metric for (a) ARPANET (b) MCI (c) Randomly generated network topology

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

TABLE I

TABLE COMPARING THE NUMBER OF FUNCTION EVALUATIONS NEEDED TO OBTAIN A "GOOD" PARAMETER SETTING FOR PIECEWISE LINEAR METRIC

2) Packet Drop Rate Metric: In this section we present the comparative results for the packet drop metric defined in (18). Figure 8 shows the comparison results of the optimization convergence speed. The results



(a) (b) (c)

Fig. 8. Figure showing the convergence curve of total packet drop rate for (a) ARPANET (b) MCI (c) Randomly generated network topology

clearly show that the recursive random algorithm significantly outperforms the local search algorithm. Table II shows that for the packet drop rate metric, our recursive random search scheme took 70% or fewer function evaluations to obtain a “good” OSPF link weight setting.

Scheme	ARPANET	MCI	Random
Local Search	882	469	372
RRS	210	125	54
Improvement	76.1%	73.3%	85.5%

TABLE II

TABLE COMPARING THE NUMBER OF FUNCTION EVALUATIONS NEEDED TO OBTAIN A “GOOD” PARAMETER SETTING PACKET DROP RATE METRIC

B. *Optimizing OSPF for improving packet drop rate*

Now we describe the simulation showing how the network performance can be improved by our OSPF optimization scheme. Figure 9 shows the functional block diagram of the overall setup of this simulation.

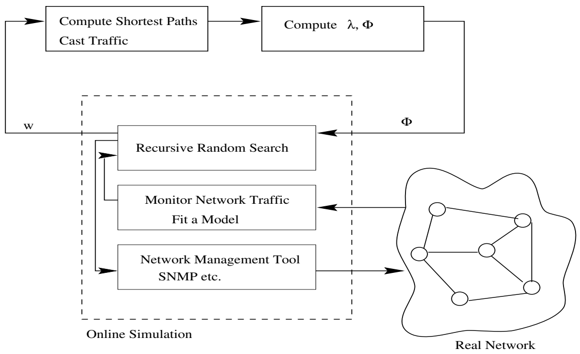
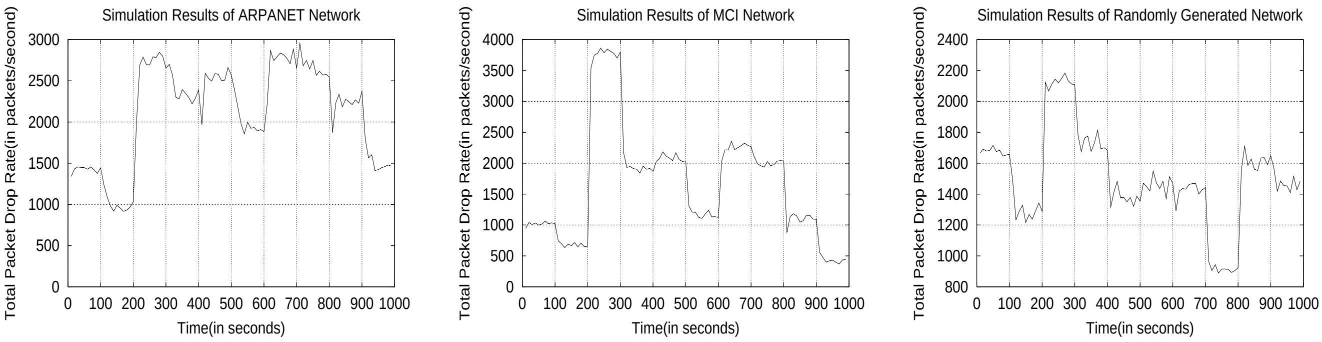


Fig. 9. Overall OSPF optimization setup using on-line simulation architecture



(a) (b) (c)

Fig. 10. Figure showing total packet drop rate as a function of 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,...

The OLS monitors the traffic to provide the estimates of mean and variance of the traffic demand for performance evaluation of link weights. Recursive random search is then be used to search for better link weight setting for the network. When a certain stopping criteria is met, for example, the time limit is reached, the best-so-far link weight setting found by RRS may be deployed in the real network if it results in substantial improvement in the performance.

We used ns[19] to simulate the real network running OSPF. The traffic in the network was generated in the same way as outlined in the beginning of this section. However, every 200 seconds the traffic pattern (the mean and variance of demand matrix) was changed in order to create a dynamic scenario. The traffic generator is implemented over UDP to generate bursty traffic with the GE inter-arrival distribution described in (4). In our simulation, we assume OLS has a complete knowledge of necessary network information, such as, traffic demands, network topology, etc.. Whenever a change of traffic pattern happens, the OLS runs the recursive random search for a certain iterations to obtain a better parameter setting. If the optimized setting is much better than the original, it will be deployed at 100 seconds after the traffic change. The 100-seconds time difference is used because we want to observe the performance difference between before optimization and after optimization. Note that here we assume the running time of the search algorithm is faster than the traffic change period, i.e., the search algorithm has finished running at 100 seconds after the traffic change.

The actual packet drop rates are collected during the simulation for all the traffic sinks in the network and then summed together to get the total packet drop rate. Figure 10 shows total packet drop rate in the network as a function of time. Table III summarizes the maximum improvement in packet drop rates for different topologies. Note that more 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 DROP RATES OBTAINED FOR DIFFERENT TOPOLOGIES FOR THE RESULTS SHOWN IN FIGURE 10

V. CONCLUSIONS AND FUTURE WORK

In this paper we investigated the problems associated with the optimization of OSPF weights using on-line simulation framework. The optimization problem was formulated where total packet drop rate was chosen as the optimization criteria. According to the offered load on each link, total packet drop rate was analytically computed using GI/M/1/K queuing model which is more general than Poisson model for allowing burstiness in traffic, and is also mathematically tractable. A fast recursive random search algorithm was used to address the issues in network optimization. Performance results demonstrate that our search algorithm took 50-90% fewer function evaluations to find a good OSPF weights "setting" as compared to the local search algorithm of [4]. The simulation results of our OSPF optimization scheme also demonstrate improvements of the order of 30-60% in the total drop rate in the network. Future work includes demonstration of dynamic optimization of OSPF weights in a real network. Investigating the issues associated with traffic monitoring and modeling and its impact on the performance of dynamic optimization will be another goal for future work.

REFERENCES

[1] 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.

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

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

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

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

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

[8] 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 Transaction on Networking*, pp. 265-278, June 2001.

[9] A. Feldmann, A. Greenberg, C. Lund, N. Reingold and J. Rexford, "Netscope: traffic engineering for IP networks," *IEEE Network Magazine*, special issue on Internet traffic engineering, pp. 11-19, March/April 2000.

[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 and S. Kalyanaraman, "A Recursive Random Search Algorithm for Optimization Network Protocol Parameters," Technical report, ECSE Department, Rensselaer Polytechnic Institute, 2001.

[19] (1997) NS(*network simulator*). http://www-mash.cs.berkeley.edu /ns.