

Stochastic Approximation and Transaction-Level Model for IP Network Design

Linhai He and Jean Walrand
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, CA USA
{linhai, wlr}@eecs.berkeley.edu

Abstract

We investigate the use of simulation and transaction-level models for TCP in IP network design. More specifically, we focus on the transaction level dynamics of TCP and approximate it by max-min fair sharing. Based on this model, we formulate a network dimensioning problem as a nonlinear constrained optimization problem. The constraints and their gradients, which do not have analytical forms, are estimated through fluid simulation of the transaction-level model of TCP. The problem is solved by a gradient descent type of algorithm, with additional heuristics based techniques to improve its convergence. The performance of the proposed algorithm is evaluated through experimental studies on example networks. Results show that our algorithm is effective and can be applied to practical network design.

1 Introduction

One key feature of today's IP networks is that it is openly shared by all network users. An end host does not need to reserve any network resources before sending information across the network. Such an architecture requires end hosts to be aware of the congestion in the network and adapt their transmission rates accordingly. This rate adaptation, supported via Transmission Control Protocol (TCP) [1], ensures fair sharing of the network resources and a graceful degradation of quality of service (QoS) in the presence of congestion.

Studies have shown that under certain assumptions, TCP connections going through a single bottleneck link tend to share the link bandwidth fairly [2]. In other words, if N TCP connections share a link with a capacity of C , the average transmission rate that a connection can achieve is about C/N . However, the result becomes more complicated when multiple TCP connections share a network of links. Simulation studies have shown that in such cases the bandwidth share of each TCP connection satisfies approximately max-min fair-

ness criterion (See [3] for its definition). This max-min fair equilibrium is unique, but it can be determined only through numerical iteration and no analytic solution is available in general.

In real networks, TCP connections arrive and leave randomly, driven by network users' behaviors. Consequently, the instantaneous transmission rate of a TCP connection, which is a function of the number of connections on the links it traverses, is a random process. This statistical fluctuation of rates has an important impact on the level of QoS perceived by end users. Because the max-min bandwidth share of connections cannot be explicitly derived, how to plan and engineer an IP network to meet users' QoS requirements poses a new challenge for network designers.

Traditional network design problems have been based on "open loop" models [3]. Such approaches usually model the networks as certain types of product-form queueing networks, which fail to capture the adaptive nature of TCP. In addition, the performance measures used in the models emphasize on packet-level dynamics such as packet delay or loss. However, we believe that it is really the transaction level dynamics of TCP instead of details of its protocol behaviors that directly impact the QoS perceived by end users. A more appropriate model for IP network design therefore should focus on transaction-level issues such as the effect of random arrival and departure of connections, max-min bandwidth sharing behavior of TCP, and so on [4].

Recently, [5] and [6] have studied the problem of link dimensioning for a single bottleneck link shared by a fixed number of ON-OFF type of traffic sources. During an ON period, a traffic source transmits a random amount of data using TCP. By a separation of time scale argument, they assume that the link bandwidth is shared by all active sources in a processor-sharing fashion and then derive the resulting performance metrics such as throughput and average ON period. However, for the aforementioned reason, their analysis becomes intractable when it is extended to a general network of links. How to solve this network dimensioning problem

is the focus of our work. In this paper, we consider a general IP network in which TCP connections arrive randomly according to a known random process. Each connection has a random amount of data to send, and the rate at which the data is transmitted is determined by the instantaneous max-min fair share of bandwidth available to the connection. With fixed routing paths, we are interested in determining the link capacities that can provide an expected level of QoS for network users, while minimizing a given cost function.

The main challenge in solving this problem is that the probabilistic QoS constraints cannot be expressed in analytical forms, since the max-min fair share can only be determined numerically. In this paper, we use simulation techniques to obtain approximation to the constraints and their gradients, and then explore the convergence properties of the proposed algorithm by experimental studies. In the following section, we describe the problem more precisely and formulate it as a formal constrained optimization problem. In Section 3, we present an algorithm that solves the problem. Their performance, evaluated through experimental studies, are shown in Section 4. We close in Section 5 with our conclusions.

2 Problem Formulation

We consider a network with a set of nodes N and directional links L . The capacity of a link $l \in L$ is denoted by C_l . The collection of all routes in the network is denoted by R . The incidence matrix for the network, $A = (A_{rl}, r \in R, l \in L)$, indicates the set of links that route r traverses. In other words, $A_{rl} = 1$ if route r traverses link l and zero otherwise. We assume that TCP connections arrive in each route r according to a Poisson process with a mean rate of λ_r , and each connection has an i.i.d amount of data to send, exponentially distributed with mean s_r . We further assume that on different routes, the arrival processes of connections and the distribution of the amount of data to be transmitted, are independent from each other. For an active connection in the network, its data is transmitted at rate of $x_r(t)$, which is its max-min fair share of bandwidth at time t ¹. Once a connection has transmitted all its data, it leaves the network.

We are interested in determining the link capacities that can provide an expected level of QoS for users, while minimizing a given cost function. In this paper, the QoS metric is chosen to be the completion time of a connection, as we believe that it directly reflects users' feeling about the degree of congestion inside a network. It is required that on route r , the probability that the

completion time of a typical connection, D_r , exceeding a specified threshold T_r , is no larger than a given level q_r . Mathematically, this requirement can be expressed as the following constraint:

$$Pr[D_r(\vec{C}) > T_r] \leq q_r, \forall r \in R,$$

where $\vec{C} = [C_l, l \in L]$ is used to emphasize the fact that D_r is a function of \vec{C} . We choose the cost function J to be the sum of the cost of all links, i.e.

$$J(\vec{C}) \triangleq \sum_{l \in L} a_l C_l^{\alpha_l}, a_l > 0, \alpha_l \in (0, 1), \forall l \in L$$

With these notations, our problem can be formulated as the following nonlinear constrained optimization problem:

$$\begin{aligned} \min \quad & J(\vec{C}) \\ \text{s.t.} \quad & Pr[D_r(\vec{C}) > T_r] \leq q_r, \forall r \in R. \end{aligned} \quad (1)$$

The nonnegativity constraint on \vec{C} is implicitly required in (1). For convenience, we define constraint functions

$$g_r(\vec{C}) \triangleq Pr[D_r(\vec{C}) > T_r] - q_r, \forall r \in R$$

for later use.

3 Algorithm

This formulation is that of a typical nonlinear constrained optimization problem. Many standard techniques, such as gradient descent and penalty function methods, are available to solve this class of problems [7]. The main challenge specific to our problem is that the constraints are expressed in probabilistic terms and do not have analytical forms. So our work focuses on how to develop effective techniques to deal with this difficulty, rather than developing new optimization algorithms.

3.1 Basic Search Algorithm

For a general optimization problem with inequality constraints:

$$\begin{aligned} \min \quad & J(\vec{x}) \\ \text{s.t.} \quad & g_i(\vec{x}) \leq 0, \forall i \in I \end{aligned}$$

where $J(\vec{x})$ and $g_i(\vec{x})$ are differentiable functions, the necessary condition for \vec{x}^* to be a local minimum is that there exists a unique set of nonnegative Lagrange multiplier $\vec{\mu} = [\mu_i, i \in I]$ such that the following conditions are satisfied [7]:

$$\begin{aligned} \nabla_{\vec{x}} J(\vec{x}^*) + \sum_{i \in I} \mu_i \nabla g_i(\vec{x}^*) &= 0 \\ \mu_i g_i(\vec{x}^*) &= 0, \forall i \in I \end{aligned}$$

A simple algorithm that could find \vec{x}^* is gradient descent method. With Lagrange function defined as

$$L(\vec{x}, \vec{\mu}, \vec{z}) = J(\vec{x}) + \sum_{i \in I} \mu_i (g_i(\vec{x}) + z_i^2),$$

¹So $x_r(t)$ changes whenever a connection arrives in or leave the network.

where $z_i, i \in I$ are slack variables to convert the inequality constraints into equality ones, \vec{x} and $\vec{\mu}$ are updated by the following equations:

$$\begin{aligned} \vec{x}(j+1) &= \vec{x}(j) - \theta_1 \nabla_{\vec{x}} L(\vec{x}(j), \vec{\mu}(j), \vec{z}(j)) \\ \mu_r(j+1) &= \max\{0, \mu_r(j) + \theta_2(g_r(\vec{x}(j)) + z_i(j)^2)\} \end{aligned}$$

where θ_1 and θ_2 are appropriately scaled step sizes. The slack variables z_i can be updated in a similar way.

To use this algorithm and other nonlinear constrained optimization algorithms as well, one has to know the value of $g_i(\vec{x})$ and its gradient $\nabla g_i(\vec{x})$. But they are not readily available in our case. This motivates us to use their approximations from simulation.

3.2 Approximations

At each iteration step, we simulate the dynamics of TCP connections arriving in and departing from the network. For all active connections on route r , they are served at rate of x_r . Whenever a connection arrives in or departs from any route, x_r is updated for all routes. As the simulation progresses, we count the number of connections whose completion time exceeds T_r and estimate g_r by the following estimator:

$$\tilde{g}_r = \frac{\sum_{m=1}^M I\{D_r(m) > T_r\}}{M} - q_r$$

where M is the total number of connections that have departed from route r . By the Law of Large Numbers, this estimate approaches g_r almost surely as M approaches infinity. Therefore, with sufficiently large number of samples, we can obtain estimate of g_r with any desired accuracy.

It is relatively more elaborate to obtain approximation to ∇g_r . Since the distribution function of the random variable $I\{D_r > T_r\}$ does not have an analytical form, and itself is discontinuous, we cannot use the perturbation analysis based gradient estimation method [8]. An alternative is to use finite-difference method to approximate ∇g_r . In the simulation, we run L additional copies of the original network. These additional networks, called perturbed networks, have the same configuration as the original one, except that the l th one has the capacity of its l th link perturbed by a small amount ϵ . The original and perturbed networks are fed with the same arrival processes of TCP connections. But due to the difference in the link capacities, the departure time of connections in these networks are different, and so is D_r . If we denote the estimation of g_r in the l th perturbed network by $\tilde{g}_r^{(l)}$, and that of the original network by $\tilde{g}_r^{(0)}$, then an approximation² to ∇g_r is

$$\widetilde{\nabla g_r} \approx \frac{\tilde{g}_r^{(l)} - \tilde{g}_r^{(0)}}{\epsilon}$$

²There are other methods which can better approximate the gradient but with higher complexity.

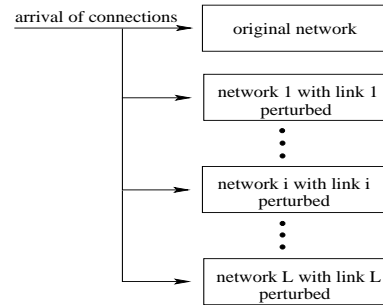


Figure 1: Perturbed networks are used to estimate the gradient of g_r .

This scheme is illustrated in Figure 1.

The update equations for \vec{C} and $\vec{\mu}$ in our algorithm then are:

$$\begin{aligned} \vec{C}(j+1) &= \max\{\vec{\rho}, \vec{C}(j) - \theta_1[\nabla J(j) + \sum_r \mu_r(j) \widetilde{\nabla g_r}(j)]\} \\ \mu_r(j+1) &= \max\{0, \mu_r(j) + \theta_r(\tilde{g}_r(j) + z_r(j)^2)\}, \forall r \in R \end{aligned}$$

where $\vec{\rho} \triangleq [\rho_l, l \in L]$ and ρ_l is the average work load on link l , defined as $\sum_{r \in R} A_{rl} \lambda_r / S_r$.

3.3 Algorithmic Techniques

It is commonly known that the basic gradient descent algorithm usually makes significant progress in the early stage of the search, but exhibits less-productive oscillatory behavior when the search is near the optimum or the boundary of the constraint set. This problem is particularly serious in our problem, as the approximations to g_r and ∇g_r introduce random noise into the updates. Although in theory the approximation error can be made arbitrarily small by sufficiently long simulation, this would increase the total run time of the algorithm. We therefore need some efficient techniques to reduce the impact of random noise and improve the convergence properties of the algorithm.

At the beginning of the algorithm, we use relatively large step size and short simulation runs to allow the search to reach the neighborhood of the optimum quickly. As the search trajectory is near the boundary of the constraint set, the algorithm becomes very sensitive and oscillation can easily happen. So in this stage of the search, we increase the accuracy of the approximations and use more careful search to home in on the optimum. We find the following heuristics based techniques have worked well in our experiments:

- We set the length of the simulation based on the tightness of the active constraints³. More precisely, after j th step, we find $\delta_j = \min_r |g_r|$, and $I = \text{argmin}_r |g_r|$. Then in the next step, the simulation is terminated only after the standard deviation of

³In our experiments, we find that the inactive constraints usually can be detected in the early stage of search.

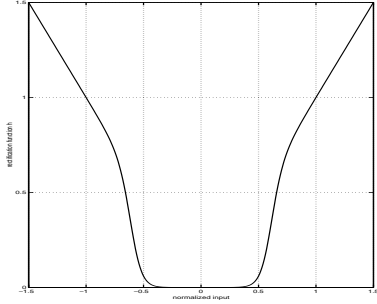


Figure 2: rectification function $h(x)$

$\tilde{g}_I(j+1)$, $\sigma_I(j+1)$, is less than some fraction of δ_j . This helps ensure the accuracy of the estimation without unnecessarily long simulations.

- The constraints control the update of the Lagrange multipliers $\vec{\mu}$. To reduce random noise in the update introduced by the estimation error in \tilde{g}_r , we use a rectified version of \tilde{g}_r , $h(\frac{\tilde{g}_r}{s\sigma_r})$, in the update. Here s is a constant (e.g. 3) and σ_r is the standard deviation of \tilde{g}_r . The function $h()$ is differentiable and has the property that it approaches x after $x > 1$, and approximately 0 when x is near the origin. An example of a feasible $h()$ is illustrated in Figure 2. The rationale behind the use of $h()$ is that when \tilde{g}_r is close to zero, it is likely the true value of g_r is close to zero and this small difference could be due to the estimation error. On the other hand, if $\tilde{g}_r \gg s\sigma_r$, the estimation error should be small compared to the true value of g_r and hence we may use \tilde{g}_r as if it is the true value of g_r . In our experiments we find that this technique helps reduce the oscillation of the search trajectory around the constraint boundary significantly.
- When the search trajectory is in the neighborhood of the optimum, ideally the variable \vec{C} should be updated only along the direction which is the projection of $\nabla J(\vec{C})$ in the null space of the gradients of all active constraints. This allows the update to reduce the cost without affecting the active constraints. Based on this observation, the variable \vec{C} is updated along the direction of \vec{P} , which is the component of $\nabla J(\vec{C})$ orthogonal to the linear space formed by ∇g_r of all constraints satisfying $|\tilde{g}_r| < \gamma\sigma_r$, where γ is some constant (e.g. 3). This also leads to the stop condition used in our experiments: if all active constraints satisfy the condition $|\tilde{g}_r| < \gamma\sigma_r$ and $\|\Delta P\| < \eta\|\nabla J(\vec{C})\|$, then the iteration stops. However, this approach does not work too well when the number of active constraints is large. In those cases, we only project $\nabla J(\vec{C})$ onto the null space of a few active constraints whose gradients have large norms.

As commonly know, step size sequence is important to

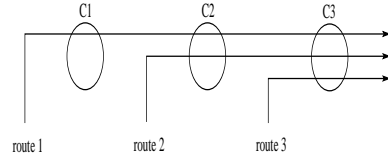


Figure 3: Example network 1.

the convergence of gradient descent algorithms. Ideally in each step a line search should be performed to find the optimal step size. But obviously this is impractical in our case. We have found that the common damping method, e.g. $\frac{1}{n}$ sequence, does not work well and often results in failure to reach the optimum. In our experiments, we use a constant step size in the early stage of the search. Once the search trajectory is near the optimum⁴, we use a small but fixed step size for \vec{C} , and have the step size individually tuned for each Lagrange multiplier (active constraints only), i.e. it is damped when \tilde{g}_r gets closer to 0 and increased when it is away from 0. This approach has worked pretty well in our experiments. On the other hand, our experience shows that careful tuning of the step sizes is still important to achieving good performance of the algorithm. In particular, the step sizes for \vec{C} and $\vec{\mu}$ need to match well; otherwise, different update speeds between these two variables can easily lead to oscillatory search trajectories.

4 Experimental Results

In this section, we evaluate the effectiveness of the gradient descent algorithm and the above algorithmic techniques in solving our problem. We first use two simple networks to show the details of the convergence properties of the algorithm, then show the results from our experiment on a more realistic backbone-alike network. We also discuss the implication of our results to practical network design.

In the first example, which is illustrated in Figure 3, three routes with different number of hops are merged into a single link. We use the same set of the parameters on all routes ($\lambda_r = 1.0$, $S_r = 1.0$, $T_r = 1.0$, and $q_r = 0.05$), but set different costs to different links ($a_1 = 1$, $a_2 = 2$, $a_3 = 3$, $a_l = 0.5$). The initial link capacities are set to be three times the average load on the links. This example is intended to study the scenario in real networks that multiple TCP connections are merged at different access stages of a network. The cost assignment reflects the fact that the unit cost of access links usually are less expensive than higher-capacity links used in the core of networks. Since link 3 has the highest load, we expect it require the highest

⁴A simple criterion is when the search trajectory cross the boundary of the constraint set for the first time

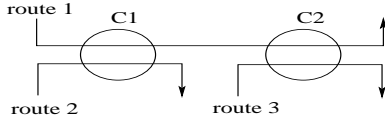


Figure 4: Example network 2.

capacity. But with the unequal cost assignment, we expect the difference between the link capacities be small, because more capacity could be allocated to less expensive link 1 and 2 to help increase the chances that link 3 being the bottleneck. Our result (Figure 6) confirms this intuitive reasoning. This implies that in practice network designers could push potential congestion spots inside the network (such as link 3 in our example) to the edge by allocating more capacities to less expensive access links.

The second example, illustrated in Figure 4, studies a long route (route 1) interacts with cross traffic (route 2 and 3). To understand the effect of the constraints on capacity assignment, we assign different QoS requirements to different routes ($q_1 = 0.05, q_2 = 0.01, q_3 = 0.1$). Other parameters are the same for all routes: $\lambda_r = 1.0, S_r = 1.0, T_r = 1.0, a_l = 1.0, \alpha_l = 0.5$. Since on link 1 route 2 has tighter constraint than route 1, we expect it be the dominant factor in determining C_1 , and link 1 is not likely to be the bottleneck for connections on route 1. On the other hand, since route 1 has tighter constraint than route 3 on link 2, we expect that the solution to C_2 would be mainly affected by the constraint of route 1, and leave route 3 as an inactive constraint. This is indeed the case, as confirmed by results in Figure 10 and 11, in which $g_3 \neq 0$ and $\mu_3 = 0$. The results also show that diverse QoS requirements also increase the difficulty of convergence. The trajectories of link capacities in Figure 9 clearly have more oscillation and takes longer to settle into the optimal solution than those in the first example.

Overall, the results from these two examples show that the algorithmic improvements we add to the basic gradient descent algorithm work well. First, the convergence of the variables $\vec{C}, \vec{\mu}$, and g_r to the optimum are smooth and has little oscillation, and the algorithm converges in a reasonable number of steps. The small magnitude of oscillation in g_r , which is largely due to the random noise in the estimation of g_r , does not introduce much undesirable effect into the update of \vec{C} and $\vec{\mu}$. This proves the effectiveness of the rectification function $h(\cdot)$ on suppressing the estimation error. Figure 10 also demonstrates the effectiveness of the projection method. In the second example, active constraints hit their boundaries much earlier than the link capacities settle to their optimum, but they remained unaffected by the subsequent adjustment of \vec{C} .

We also test our algorithm on a more realistic network

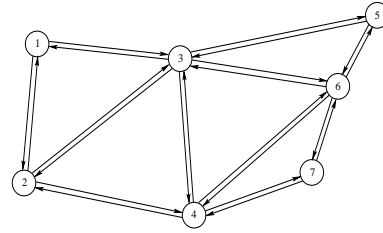


Figure 5: A backbone type of network

modeled after an IP backbone network. Its topology is illustrated in Figure 5. It has a total of 7 nodes, representing major metropolitan areas in US. There is a unique route between any source-destination pair in the network (hence a total of 42 routes). We assume that each route follows the shortest path through the network (the cost of a path is defined by its total hop count). In cases where there is not a unique shortest path, the one with the smallest sum of node IDs is chosen. In the experiment, we set the arrival rates on all routes originating from Node 2, 3, 5, and 6 to 2, and to 1 on all other routes. The rest of parameters are the same for all routes: $S_r = 1.0, T_r = 1.0$, and $q_r = 0.05$. The cost parameters for all links are the same, with $a_l = 1.0$ and $\alpha_l = 0.5$. The initial value of C_l is set to three times the average load on that link. Due to the size of the problem, in this paper we only show the search trajectory of the link from Node 3 to 4 and the route from Node 3 to 7 in Figure 12 through 14. The final optimal solution of the link capacities are shown in Figure 15. In the plot, the width of the links are proportional to their capacities, and the numbers next to them are their final values. Despite the much larger size of the problem (22 variables with 42 constraints) as compared to the previous examples, the algorithm still converges reasonably well. Although the convergence time is longer, its search trajectories show the similar characteristics of those in previous examples. We therefore expect that our algorithm will work well in practical network design cases too.

5 Conclusion

In this paper we investigate the use of simulation and transaction-level models for TCP in IP network design. We model the rate adaptation of TCP connections at transaction level by max-min fair sharing, and then use that model in a formulation of a network dimensioning problem. More specifically, we consider an IP network in which TCP connections arrive and depart randomly and share the network bandwidth according to max-min fair criterion. We then determine the link capacities that can provide an expected level of QoS for users, while minimizing a given cost function.

The main contribution of this paper is that it investi-

gates the feasibility of using stochastic approximation of the constraints and their gradients, obtained from simulation, in standard optimization algorithms. A set of heuristics based techniques are also proposed to deal with errors introduced by the approximation and improve the convergence properties of the basic algorithm. Our experimental studies on three different networks show that the proposed algorithm works well and can be applied to practical IP network design.

For the future work, our model can be improved to better approximate the behavior of TCP and include uncertainties in modeling. In addition, an analysis on the convergence speed of the algorithm would be interesting and should help better understand the performance of the algorithm.

References

- [1] IETF Request For Comments, "Transmission Protocol: DARPA Internet Program Protocol Specification," September 1981. Available at <http://www.ietf.org/rfc/rfc0793.txt>.
- [2] T. V. Lakshman and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss," *IEEE/ACM Trans. Networking*, June 1997.
- [3] D. P. Bertsekas and Robert Gallager. *Data Networks*, 2nd ed., Prentice Hall, 1992.
- [4] J. Walrand. "A Transaction-Level Tool for predicting TCP Performance and for Network Engineering," *Proceedings of MASCOTS 2000*, page 106-112, September 2000.
- [5] D. Heyman, T. V. Lakshman and A. Niedhart. "A new method for analyzing feedback-based protocols with applications to engineering web traffic over the Internet." In *Sigmetrics 97*, page 24-38, 1997.
- [6] A. Das and R. Srikant. "Diffusion approximations for models of congestion control in high-speed networks," in the Proceedings of the 37th IEEE Conference on Decision and Control, December 1998.
- [7] D. P. Bertsekas. *Nonlinear Programming*, Athena Scientific, 1995.
- [8] P. Glasserman. *Gradient Estimation Via Perturbation Analysis*, Kluwer Academic Publishers, 1991.
- [9] D. G. Luenberger. *Linear and Nonlinear Programming*, Addison-Wesley, 1989.

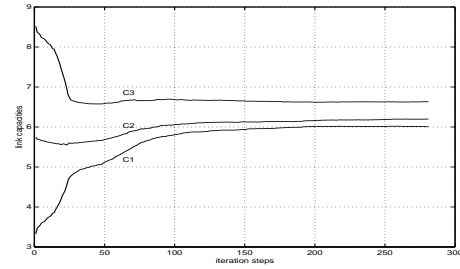


Figure 6: Link capacities in Network 1.

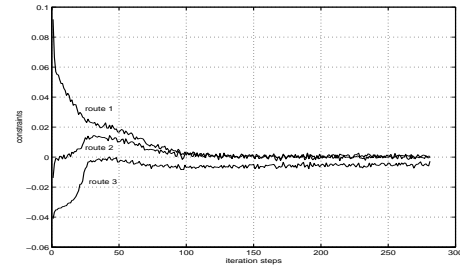


Figure 7: Constraints in Network 1.

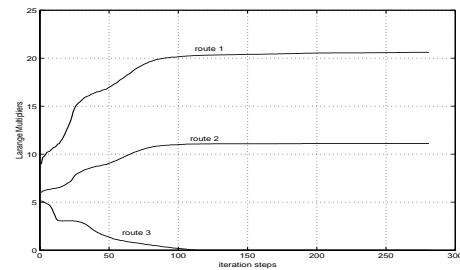


Figure 8: Lagrange multipliers in Network 1.

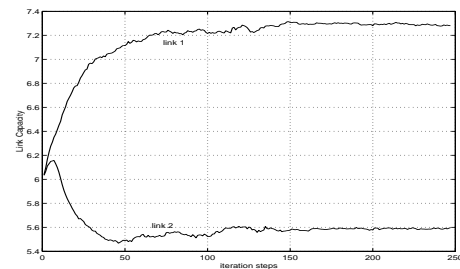


Figure 9: Link capacities in Network 2.

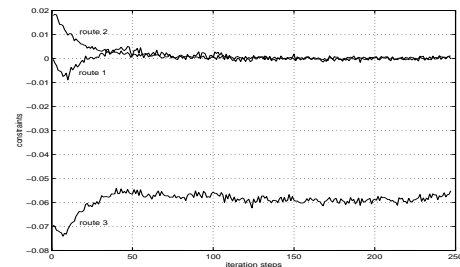


Figure 10: Constraints in Network 2.

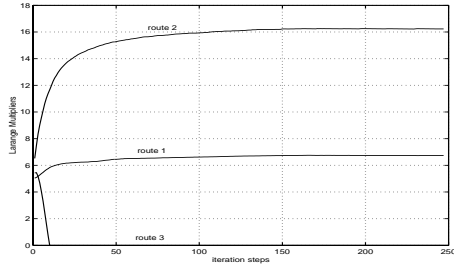


Figure 11: Lagrange multipliers in Network 2.

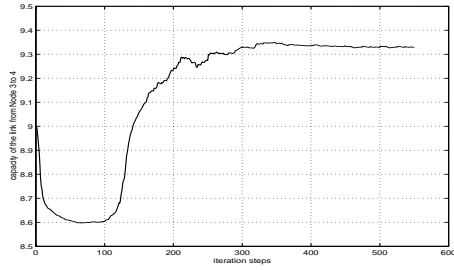


Figure 12: Capacity of the link from Node 3 to 4.

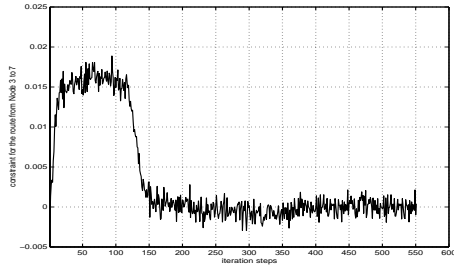


Figure 13: Constraint for the route from Node 3 to 7.

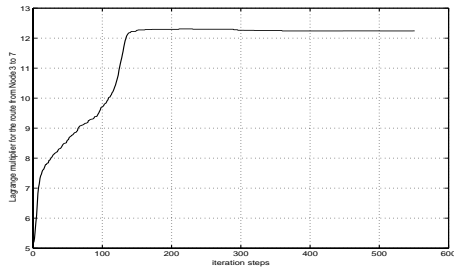


Figure 14: Lagrange multiplier for the route from Node 3 to 7.

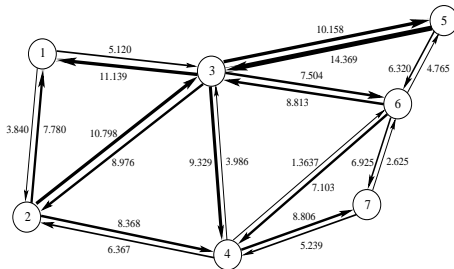


Figure 15: Optimal link capacities for Example network 3.