The Practical Performance of Subgradient Computational Techniques for Mesh Network Utility Optimization

Peng Wang and Stephan Bohacek

Department of Electrical and Computer Engineering
University of Delaware, Newark DE 19716, USA,
email:{pwang,bohacek}@udel.edu phone: 302-831-4274, fax: 302-831-4316

Abstract. In the networking research literature, the problem of network utility optimization is often converted to the dual problem which, due to nondifferentiability, is solved with a particular subgradient technique. This technique is not an ascent scheme, hence each iteration does not necessarily improve the value of the dual function. This paper examines the performance of this computational technique in realistic mesh network settings. The traditional subgradient technique is compared to a subgradient technique that is an ascent algorithm. It is found that the traditional subgradient techniques suffers from poor performance. Specifically, for large networks, the convergence is slow. While increasing the step size improves convergence speed, due to stability problems, the step size cannot be set arbitrarily high, and suitable step sizes result in slow convergence. The traditional subgradient technique also suffers from difficulties when used online. The ascent scheme performs well in all respects, however, it is not a distributed technique.

Keywords: Network capacity optimization, subgradient techniques.

1 Introduction

There has been extensive effort focused on finding time division multiplexing schedules that maximize the capacity of wireless networks [1]- [10]. A common approach is to maximize the sum of flow utilities subject to constraints related to interference. Specifically, we consider

$$\min -\sum_{\phi\in\Phi} U_\phi(f_\phi) \tag{1}$$

$$\text{subject to:} \quad \sum_{\{\phi | l\in P(\phi)\}} f_\phi \leq \sum_{v\in V} \alpha_v R(v,l) \text{ for all } l \text{ and } \sum_{v\in V} \alpha_v = 1,\ \alpha_v \geq 0.$$

where f_ϕ is the data rate of flow ϕ, $U_\phi(f_\phi)$ is the utility of flow ϕ when the flow rate is f_ϕ, $\mathcal{P}(\phi)$ is the set of links that flow ϕ traverses (i.e., $\mathcal{P}(\phi)$ is the path of flow ϕ), $R(v,l)$ is the data rate over link l when assignment v is used, and

α_v is the duration that assignment v is used. We define an *assignment* to be a specification of which links transmit and the transmit power. Thus, a schedule is a weighted combination of assignments where the weights are α_v. We let V denote the set of considered assignments. If power control is not used, then there are 2^L distinct assignments, where L is the number of links in the network, and if power control is used, the space of assignments is $[0,1]^L$. In [1], a technique is presented that generates a small set V that results in nearly optimal utility. Hence, currently, utility optimization is tractable for networks with hundreds of links. Most efforts to solve (1) use dual or primal-dual techniques. Specifically, after some manipulation, the dual function is written as

$$q\left(\mu\right) = \sum_{\phi \in \Phi} \inf_{f_\phi \geq 0} \left(-U_\phi(f_\phi) + f_\phi \sum_{l \in \mathcal{P}(\phi)} \mu_l \right) - \max_{v \in V} \sum_{l=1}^{L} R(v,l)\mu_l, \tag{2}$$

where μ_l is the Lagrange multiplier associated with link l. The dual problem is

$$\max_{\mu \geq 0} q\left(\mu\right). \tag{3}$$

Due to the term $\max_{v \in V} \sum_{l=1}^{L} R(v,l)\mu_l$, the dual function, q, is not differentiable for all μ. Hence, computational methods based on the gradient are not available. To circumvent this difficulty, supergradient[1] techniques can be employed. In the networking literature [2]- [10], the most popular supergradient technique is to iterate

$$\mu_l\left(k+1\right) = \left(\mu_l\left(k\right) + \gamma_k \left(\sum_{\{\phi | l \in \mathcal{P}(\phi)\}} f_\phi^*\left(\mu\left(k\right)\right) - R\left(v\left(k\right), l\right) \right) \right)^+ \tag{4}$$

where

$$v\left(k\right) \in \arg\max_{v \in V} \sum R\left(v,l\right) \mu_l\left(k\right), \tag{5}$$

γ_k is a step size, and $f_\phi^*\left(\mu\left(k\right)\right)$ is the optimal flow given $\mu\left(k\right)$, i.e., f_ϕ^* is the solution to the infimum in (2). Since this scheme is widely used, it will be referred to as the *traditional supergradient scheme*.

This paper examines the practical performance of (4) through extensive computational experiments. The conclusions are that the traditional supergradient scheme suffers from poor performance. Specifically, for large networks, the convergence is slow. While increasing the step size improves convergence speed, due to stability problems, the step size cannot be set arbitrarily high, and suitable step sizes result in slow convergence. On the other hand, this method does not find the exact solution, but merely oscillates around the optimal solution. However, the oscillations are small, hence in terms of error, (4) works well. Often the traditional supergradient techniques are used for online and distributed

[1] Subgradient is a more common term. However, subgradient and supergradient techniques are the same, the only difference is that the former refers to minimization while the later refers to maximization, which is the focus here.

computation. However, this approach suffers from several problems. Finally, an alternative ascent algorithm is also investigated. While this approach does not appear to lend itself to distribution, it does perform well in all other aspects.

The remainder of the paper proceeds as follows. In the next section, a few theoretical aspects of supergradient based optimization are presented. In Section 3, some details of the computational experiments are provided. The rest of the paper is focused on the performance of the traditional supergradient scheme, specifically, Section 4 examines the convergence rate, Section 5 examines the error, Section 6 examines stability, and Section 7 examines the performance when the traditional supergradient scheme is used as an online and distributed computational method. Finally, Section 8 provides some concluding remarks.

2 Theoretical Results on Supergradient Optimization

The performance of (4) has been extensively investigated (e.g., see [11]). In [9], the following is proved.

Theorem 1. *Let γ_k be a constant γ and let $G = \max_\mu \|\partial q(\mu)\|$, where $\|\partial q(\mu)\|$ is the norm of the largest element in the superdifferential $\partial q(\mu)$ and let $u(k)$ be given by (4). Then*

$$\lim_{K\to\infty} \sup \frac{1}{K}\sum_{k=1}^{K} |q(\mu(k)) - q(\mu^*)| < \gamma G^2/2.$$

Thus, one can expect that if a fixed step size is used, then $\mu(k)$ will enter a ball around μ^* and remain in this ball, where μ^* is the solution to (3). Hence, using the terminology of [9], we can consider that the $\mu(k)$ has *stochastically converged* when it enters this ball. The ball can be made smaller by using a smaller step size. In fact, by slowly decreasing the step size, this scheme will converge. However, in order to guarantee convergence, the step size must converge slowly. Specifically, in general, we must have $\lim_{k\to\infty} \gamma_k g_k = 0$ and $\sum_{k=1}^{\infty} 1/(\gamma_k g_k)^2 = \infty$, where $g_k = \left(\sum_l \left(\sum_{\{\phi | l\in\mathcal{P}(\phi)\}} f_\phi^*(\mu(k\Delta t)) - R(v(k), l)\right)^2\right)^{1/2}$ [11].

When $q(\mu)$ is not differentiable, the superdifferential, $\partial q(u)$, is a set of vectors. The algorithm (4) arbitrarily selects one element from the superdifferential and uses it as if it was a direction of ascent. As just mentioned, if the step size is selected correctly, then this scheme will converge. However, it is possible to more carefully select the direction so that it is a direction of ascent[2].

Theorem 2 (Thm 1.11 in [11]). *Let $\partial q(\mu)$ be the superdifferential of q at μ. Suppose $0 \notin \partial q(\mu)$ and let η be the element of $\partial q(\mu)$ that is nearest to the origin, i.e.,*

$$\eta = \arg\min \|g\|^2 \qquad (6)$$
$$\textit{subject to: } g \in \partial q(\mu).$$

Then η is a direction of steepest descent at μ.

[2] Note due to convexity, there must be a direction of ascent, unless $\mu(k) = \mu^*$.

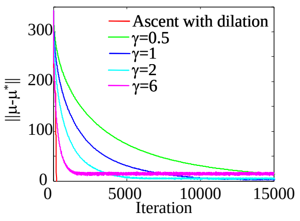

Fig. 1. Example of the convergence of (4) for a 60 link network.

Therefore, steepest ascent is an alternative computational scheme to the traditional supergradient scheme (4). However, as is well known, steepest ascent can lead to oscillations that result in slow convergence. The steepest ascent algorithm can be further improved by using space dilation (see page 69 in [11]). We refer to this approach as the *ascent algorithm*. More details on the ascent algorithm can be found in [1]. Section 4 compares the convergence rate of this ascent algorithm to the traditional supergradient algorithm (4). In the other sections, this ascent algorithm is used to find μ^*, the optimal solution to (3) as well as optimal flow and link rates.

3 Experiment Set Up

The performance of (4) and the ascent algorithm were examined in realistic mesh network scenarios that were based on downtown Chicago. Specifically, random mesh networks were generated by placing one infrastructure node randomly on each block in a region of downtown Chicago. A centrally located infrastructure node was designated as the base station. All other nodes were set to be wireless relays. These wireless relays were also set as destinations. Hence, for each relay, there was one flow from the base station to the relay. Shortest path routing was used, where the channel loss along each hop was required to be no more than 55 dB. The propagation was determined from the UDelModels ray-tracing tool [12]. If some relays were disconnected from the network, then the relay was excluded from the topology. Finally, by adjusting the size of the region of Chicago where the mesh network was constructed, the number of links could be approximately controlled. Topologies were grouped together based on the number of links. Twenty topologies were generated for each number of links, where the number of links ranged from 16 to 75 links in steps of five links.

As mentioned in the Introduction, when there are L links, there are 2^L possible assignments. Hence, for large topologies it is intractable to consider all possible assignments. Instead, the scheme described in [1] was used to construct a good set of assignments. In [1], it is shown that this technique results in network utility that is with 0.05% of optimal. Thus, the set V in the Introduction was set to be this set of good assignments.

Finally, the utility function used was $U(f) = \log(f)$ and data rates were given by Shannon's Theorem, i.e., $\log_2(1 + SNIR)$ bits/Hz.

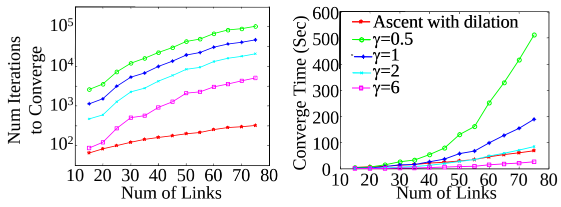

Fig. 2. Left: Number of Iterations until convergence. Right: Computation time on an 2.8GHz 64 bit PC until convergence.

4 Convergence Rate

There are few theoretical results on the convergence rate of (4). However, it is intuitive that a smaller step size results in a slower convergence. Figure 1 shows examples of $\|\mu(k) - \mu^*\|$ for (4) with several step sizes and for the ascent algorithm. Note that the ascent algorithm will eventually converge, hence the curve representing the ascent algorithm is only shown for the iterations before convergence.

We will say that the traditional supergradient scheme has converged the when

$$\|\mu(k) - \mu^*\| \leq \lim_{k\to\infty} E(\|\mu(k) - \mu^*\|).$$

Once this condition has been met, we can assume that $\mu(k)$ remains in a ball around μ^* and the flow and link rates will be approximately correct.

Figure 2 shows the number of iterations until convergence and the computation time until convergence. Here we assume the computation is performed centrally. Thus, the computation time for one iteration is the time to update μ_l for each link.

The left-hand frame of Figure 2 shows that the convergence time does not grow exponentially with the number of links. However, the right-hand frame shows a superlinear growth in the convergence time with the number of links. On the other hand, the ascent algorithm shows a slower growth than the traditional supergradient method. To see this, note that for $\gamma = 2$, the traditional supergradient method converges in less time than ascent algorithm when the number of links is small, but requires more time when the numbers of links is large. Similarly, while $\gamma = 6$ takes less time than the ascent algorithm when there are 75 or fewer links, it takes more time for large networks. For example, we found for a set of networks with 280 links, the traditional supergradient method with $\gamma = 6$ takes approximately 800 seconds, whereas the ascent algorithm takes approximately 550 seconds.

5 Error

The relationship between the number of iterations to reach convergence and the step size indicates that if γ is selected very large, then convergence will be very fast. On the other hand, Theorem 1 indicates that the error $\|\mu(k) - \mu^*\|$ grows with γ.

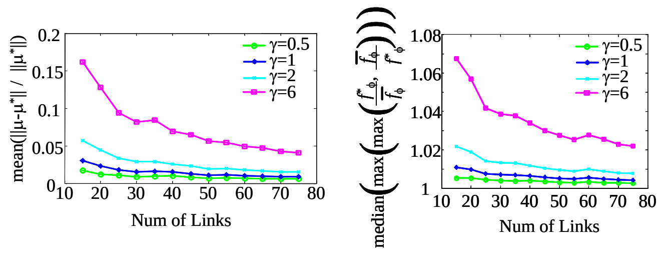

Fig. 3. Left: The relative error of μ. The mean is over all topologies with L links. Right: The ratio of average flow rates to optimal flow rate.

To investigate this, we consider the average relative error after convergence (i.e., the average value of $\|\mu(k) - \mu^*\| / \|\mu^*\|$ for very large k). The left-hand frame of Figure 3 shows that the relative error is quite small and decreases with the number of links.

The right-hand frame of Figure 3 provides another view of the error. To understand this plot, recall that given $\mu(k)$, the flow rates, $f_\phi(k)$, can be determined. If $\mu(k)$ differs from μ^*, then $f_\phi(k)$ will differ from f_ϕ^*. To examine the size of this difference, we compare f_ϕ^* and $\bar{f}_\phi$, the average value of $f_\phi(k)$ for k very large, i.e., after convergence. Specifically, we examine

$$\underset{\text{over all topologies}}{\text{mean}} \max_\phi \max\left(\frac{f_\phi^*}{\bar{f}_\phi}, \frac{\bar{f}_\phi}{f_\phi^*}\right).$$

Note that the inner maximization forces the ratio to always be greater than one. The outer maximization is the maximization over all flows, i.e., the worst case flow. The mean averages over all topologies.

In both views of the error, we see that the error decreases with the number of links. Further investigation is required to understand why this is the case. Nonetheless, in both cases the error is quite small.

6 Stability

Section 4 showed that the time to convergence decreases when the step size, γ, is increased. Furthermore, the previous section showed that the error is quite small even for $\gamma = 6$. Moreover, the error decreases with the number of links. Hence, increasing the step size may improve convergence while maintaining acceptable error. However, we find that large step sizes can lead to instability and divergence.

For a particular topology, we define $\bar{\gamma}$ to be the maximum value of γ such that (4) is stable. Figure 4 shows minimum value of $\bar{\gamma}$ where the minimum is over all topologies with L links. Figure 4 also shows the median value of $\bar{\gamma}$ over all topologies with L links. While Figure 4 indicates that in some cases large values of γ might not cause instability, there are other topologies such that γ must be rather small. Indeed, from Figure 4, we conclude that it is not possible to reliably set γ larger than 6.

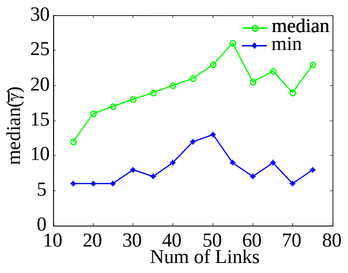

Fig. 4. The median and minimum value of $\overline{\gamma}$, the maximum allowable value of γ such that the traditional subgradient method is stable, i.e., does not diverge. The median and minimum are over all topologies with L links.

7 Online and Distributed Supergradient Optimization

In this section the possibility of distributing the supergradient optimization in such a way that it supports online computation of assignments. By online we mean that at each iteration, the assignment $v(k)$ is used, i.e., the link bit-rates are $R(v(k), l)$. This assignment is used for Δt seconds before a new assignment, $v(k+1)$, is generated. Note that Δt is not necessarily the same as γ. However, here we assume that $\gamma = \Delta t$.

We assume that the computation of a new assignment requires communication with neighboring nodes. Recall that we assume that the $v(k) \in \arg\max_{v \in V} \sum R(v, l)\, \mu_l(k)$. Thus, in order to compute $v(k)$, each link must be aware of $\mu_l(k)$ for all other links. Consequently, each iteration can be expensive in terms of bandwidth, the resource that is being optimized. In order to preserve bandwidth, one can set Δt large. In this section, the values of Δt studied range from 500 msec. to 6 sec. Refer to Figure 2 for the number of iterations required for convergence. For example, with 75 links and $\Delta t = 500$ msec, it will take 50000 seconds until convergence.

Besides slow convergence, there are two performance problems with the online approach, namely, the actual link utilization of congested links may be small and queues occupancies can be very large. These problems are discussed next.

7.1 Link Utilization

When using TDM, a link is not able to transmit at all times. However, for some time-slots, the link is able to transmit and it is expected that the link will transmit continuously during that time-slot. If the link is unable to transmit data throughout the entire time-slot, then it might be possible to either increase the flow rates or use different assignment so that other can links to transmit. If either of these options is possible, then the network utility can be increased. On the other hand, it is possible that at optimality a link will have more bandwidth allocated to it than is required to transmit the data passing over it. However, from complementary sensitivity, for links l where this occurs, we must have $\mu_l^* = 0$. Thus, if $\mu_l^* > 0$, then we expect that link l will always send data when it is allocated bandwidth, that is, the link will be fully utilized.

Since a radio cannot simultaneously transmit and receive on the same bandwidth, when a node is transmitting, it must transmit data that is stored in its queue. Thus, letting $Q_l(k)$ denote the queue occupancy of link l at the beginning of the kth time-slot, a link is underutilized if $Q_l(k) < \Delta t R(v(k), l)$, i.e., more data can be sent than is available in the queue. Thus, we define the utilization of a link to be

$$\rho_l := \frac{\sum\limits_{\{k:R(v(k),l)>0\}} \min\left(Q_l(k), \Delta t R(v(k), l)\right)}{\sum\limits_{\{k:R(v(k),l)>0\}} \Delta t R(v(k), l)},$$

where $\{k : R(v(k), l) > 0\}$ is the set of time-slots for which link l is transmitting. In the analysis that follows, the utilization is computed once the algorithm has stochastically converged.

We approximate the queue occupancy with the following

$$Q_l(k+1) = \min\left(Q_{\max}, \left(Q_l(k) + \mu_l(k+1) - \mu_l(k)\right)^+\right) \tag{7}$$

where $Q_{\max}$ is the size of the queue. Note that if $Q_{\max} = \infty$ and $\mu(0) = Q(0)$, then $\mu_l(k) = Q_l(k)$ for all k. Also, note that (7) is only an approximation of the queue occupancy since it assumes that the arrival flow rate for link l is $\sum_{\{\phi:l\in\mathcal{P}(\phi)\}} f_\phi^*(\mu(k))$. However, upstream queue overflows could result in arrival rates less than $\sum_{\{\phi:l\in\mathcal{P}(\phi)\}} f_\phi^*(\mu(k))$. Nonetheless, the analysis that follows uses (7).

There are two ways in which the traditional supergradient method results in congested links having utilization that is less than one. First, as shown in Figure 3, the larger that γ is, the larger the variations experienced by $\mu(k)$, and hence the larger the variations experienced by $Q(k)$. Consequently, when μ_l^* is small for some link l, variations in $\mu_l(k)$ around μ_l^* will occasionally result in $\mu_l(k) = 0$. Similarly, occasionally $Q(k) = 0$, and hence $\rho_l < 1$.

While $Q(k) = 0$ will result in $\rho_l < 1$, this problem is most significant for links with small μ_l^*. Considering sensitivity analysis[3], these links with small μ_l^* are not as critical as links with larger μ_l^*. Hence, if these less critical links do not reach full utilization, it will not have a significant impact on the network utility unless Δt is quite large (in which case, occasionally we will have $Q_l(k) = 0$ for links with large μ_l^*). However, as discussed in Section 6, due to instability, it is difficult to have Δt large.

Finite queue sizes is a second cause of reduced link utilizations. For example, since a node cannot send and transmit at the same time, if the maximum queue size is zero (i.e., there is no queue), then $\rho = 0$. In general, finite queue size is not a problem if $Q_{\max} \geq \max_l \max_{v\in V} \Delta t R(v, l)$. This condition can be conservative

[3] By sensitivity, the Lagrange multiplier μ_l is related to the change in the network utility due to a change in link resources. Hence, if μ_l is small, then decreasing the data rate across link l will only have a small impact on the network utility. Thus, if μ_l^* is small, then link l is less critical to the network utility.

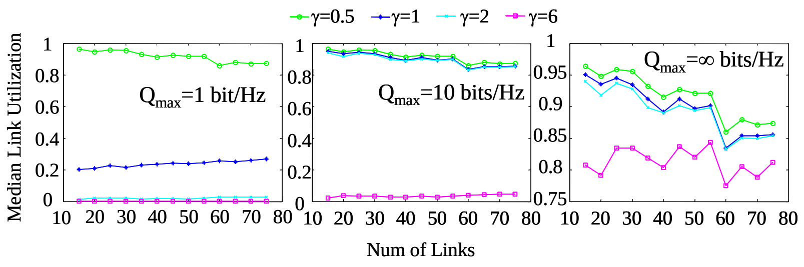

Fig. 5. Traditional subgradient methods can result in link utilization of less than one for congested links (i.e., links with $\mu_l > 0$). The above shows the median link utilization where the median is over all links and all sampled topologies with L links. As the $Q_{\max} \to \infty$, the median link utilization converges.

since links with very high data rates might not be critical links (i.e., $\mu_l^* = 0$), and not all assignments v are used.

Figure 5 shows the median link utilization for different maximum queue sizes, $Q_{\max}$ (only links with $\mu_l^* > 0$ are considered). As expected, for very small sizes of $Q_{\max}$, the utilization is quite low. This is due to $Q_{\max} < \Delta t R(v, l)$ for some l and some v that is used by the schedule. When $Q_{\max} = \infty$, then the utilization is less than one due to $Q_l(k) = 0$ for some k and l. As can be seen, the median link utilization is far from one in all cases.

7.2 Queue Size and Delay

An important drawback of the online implementation of the supergradient method is that the link cost, μ_l, is tightly associated with the queue occupancy, Q_l. In the typical approach, $\mu_l = Q_l$. This is problematic since if the link cost is high, then the queue occupancy will be large, resulting in long delays and consuming large amounts of memory resources. For example, in our experiments, it was not uncommon to have $\mu_l^* > 100$ bits/Hz. If the bandwidth is 20 MHz, as is the case in 802.11b/g, this would result in nominal queue occupancies of 2Gb. As discussed above, limiting the queue to smaller values decreases link utilization.

Another possible option is to somehow force $Q_l(k) = \mu_l(k) - \mu_l^*$. In this case, the queue is nominally empty and only grows when $\mu_l > \mu_l^*$. In this case, delay is only caused by positive variations in μ_l. However, as shown in Figure 6, even in this ideal situation, we find that the queue must be large. Indeed, when $\Delta t = 500$ msec and the bandwidth is 20 MHz, we have some queue occupancies that exceed 10 Mb.

8 Conclusions

It is common to use a particular supergradient technique to maximize network utility. This paper examines the performance of the traditional supergradient technique and finds that in practice, it performs poorly. Specifically, convergence is slow, and instability results if the step size is increased in an attempt to improve convergence speed. An alternative ascent algorithm is found to converge much faster. Another problem with the traditional supergradient approach is that if it is distributed, then queue occupancies can become very large and

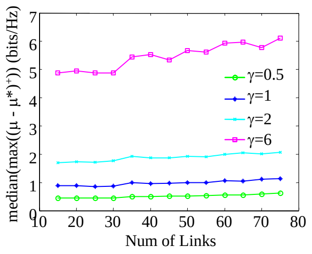

Fig. 6. Median of the Maximum Positive Deviation of $\mu - \mu^*$. The maximum is over all links in the topology and over all time, and the median is over all topologies with L links.

link utilization of critical links is below one. On the other hand, while the supergradient methods do not provide the exact solution (they oscillate around it), the error is small.

References

1. S. Bohacek and P. Wang, “Toward tractable computation of the capacity multihop wireless networks,” in *Infocom*, 2007, available at: http://udelmodels.eecis.udel.edu/publications1.php.
2. M. Chiang, “Balancing transport and physical layers in wireless multihop networks: Jointly optimal congestion control and power control,” *IEEE Journal on Selected Areas in Communications*, vol. 23, pp. 104–116, 2005.
3. R. Cruz and A. Santhanam, “Optimal routing, link scheduling and power control in multi-hop wireless networks,” in *IEEE INFOCOM*, March 2003.
4. J. Yuan, Z. Li, W. Yu, and B. Li, “A cross-layer optimization framework for multihop multicast in wireless mesh networks,” *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 2092–2103, Nov 2006.
5. N. Jin, G. Venkitachalam, and S. Jordan, “Dynamic congestion-based pricing of bandwidth and buffer,” *IEEE/ACM Transactions on Networking*, vol. 13, no. 6, pp. 1233 – 1246, Dec 2005.
6. R. Madan and S. Lall, “Distributed algorithms for maximum lifetime routing in wireless sensor networks,” in *IEEE GLOBECOM*, Nov 2004, pp. 748 – 753.
7. D. S. Lun, N. Ratnakar, M. Médard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao, “Minimum-cost multicast over coded packet networks,” *IEEE/ACM Transactions on Networking (TON)*, vol. 14, pp. 2608 – 2623, June 2006.
8. X. Wang and K. Kar, “Cross-layer rate control for end-to-end proportional fairness in wireless networks with random access,” in *MobiHoc*, May 2005.
9. L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, “Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks,” in *Infocom*, 2006.
10. X. Lin and N. B. Shroff, “The impact of imperfect scheduling on cross-layer congestion control in wireless networks,” *IEEE/ACM Transactions on Networking*, vol. 14, no. 2, pp. 302–315, April 2006.
11. N. Z. Shor, *Minimization Methods for Non-Differentiable Functions.* Berlin: Springer-Verlag, 1985, p69.
12. S. Bohacek, V. Sridhara, and J. Kim, “UDel Models,” available at: http://udelmodels.eecis.udel.edu/.