DYNAMIC ROUTING AND QUEUE MANAGEMENT VIA BUNDLE SUBGRADIENT METHODS

Almir Mutapcic, Majid Emami and Keyvan Mohajer

Information Systems Laboratory
Stanford University, Stanford, CA 94305

Keywords: Dynamic network routing, queue management, dual methods, bundle subgradient methods.

Abstract: In this paper we propose a purely distributed dynamic network routing algorithm that simultaneously regulates queue sizes across the network. The algorithm is distributed since each node decides on its outgoing link flows based only on its own and its immediate neighbors' information. Therefore, this routing method will be adaptive and robust to changes in the network topology, such as the node or link failures. This algorithm is based on the idea of bundle subgradient methods, which accelerate convergence when applied to regular non-differentiable optimization problems. In the optimal network flow framework, we show that queues can be treated as subgradient accumulations and thus bundle subgradient methods also drive average queue sizes to zero. We prove the convergence of our proposed algorithm and we state stability conditions for constant step size update rules. The algorithm is implemented using Matlab and its performance is analyzed on a test network with varying data traffic patterns.

1 INTRODUCTION

This paper investigates joint dynamic routing and queue management in data networks. In networking literature, queue management is often referred to as congestion control, since congestion in networks occurs when current link capacities cannot satisfy the user's needs, and we have to delay the user's data packets by storing them in queues. Network performance and its *Quality of Service* (QoS) are measured by metrics such as routing delay, maximum link utilization, convergence time after failures, *etc*.

Many current routing algorithms used in practice are based on heuristics and are not optimal in any sense. They are often *static* (fixed-time) algorithms; they make flow decisions ahead of time and fix all routing tables for future network use. For example, most of routing protocols use hop-counts or some artificial weights assigned to the links in order to derive routing tables for a given network.

We will investigate new routing strategies that are *dynamic* in time, and that simultaneously optimize queue lengths in the network. Dynamic routing algorithms continuously change packet routes as the network topology and users' demands change; for a dynamic routing algorithm example see (Segall, 1977).

Routing methods can also be classified based on the type of network coordination. We can have centralized, synchronously distributed, and asynchronously distributed algorithms. We are interested in distributed algorithms since they do not require any centralized coordinator with a global network knowledge, and they are more robust to adversarial changes in network topology.

2 OPTIMAL ROUTING AND QUEUE MANAGEMENT

2.1 Optimal routing

One way to improve routing through network is by using multiple paths between any pair of source and destination nodes in the network. Current routing methods like the routing mechanism in the *OSPF* protocol utilize this idea in a limited sense, *i.e.*, the data payload is divided among the shortest paths (if more than one) toward the given destination. This is still not an optimal solution; however, it is better than using a single routing path.

Mutapcic A., Emami M. and Mohajer K. (2004).
DYNAMIC ROUTING AND QUEUE MANAGEMENT VIA BUNDLE SUBGRADIENT METHODS.
In *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics*, pages 12-19
DOI: 10.5220/0001144800120019

2.2 Queue management

The main goal of a queue management algorithm is to maximize throughput and minimize queue delays in a network. In this case it is highly desirable to have a distributed queue management algorithm since it is impossible to coordinate all source nodes in a large network.

Our routing strategy will optimize average queue sizes, and therefore implement a dynamic routing strategy with active queue management.

2.3 Problem formulation

General joint dynamic routing and queue management problems can be formulated as a convex optimization problem constrained by a linear dynamical system and feasibility constraints. This problem is an instance of the optimal control problem, where our objective (performance) function can be any convex function. Using a discrete time queuing model, and a single user traffic on a connected directed network with n nodes and p links, we obtain the following problem

$$\begin{array}{ll} \text{minimize} & \sum_{t=t_i}^{t_f} \Big[\sum_{j=1}^{p} \phi_j \left(x_j(t)\right) \\ & \qquad + \sum_{i=1}^{n} \psi_i \left(q_i(t)\right) - U\left(s(t)\right) \Big] \\ \text{subject to} & q(t+1) = q(t) + s(t) - Ax(t), \\ & \qquad -c \preceq x(t) \preceq c, \\ & \qquad 0 \preceq q(t) \preceq Q_{\max}, \\ & \qquad S_{\min} \preceq s(t) \preceq S_{\max}. \end{array}$$

Problem variables are: traffic flows $x(t) \in \mathbf{R}^p$, which can have negative components since we allow reverse flow on the network links (*i.e.*, the links are bi-directional); queue lengths $q(t) \in \mathbf{R}^n$ represent amount of packets waiting to be processed at each node's queue; source rates $s(t) \in \mathbf{R}^n$ is a vector of incoming and removed network traffic at each node, such that $\sum_{i=1}^{n} s_i(t) = 0$. The flows are restricted by the given link capacities $c_j > 0$, the queues size limit is $Q_{\max}$, and the source-sink rates can be varied between $S_{\min}$ and $S_{\max}$. The matrix $A \in \mathbf{R}^{n \times p}$ is the node incidence matrix for the given directed graph.

The function $\phi_j : \mathbf{R} \to \mathbf{R}$ is the flow cost function for link j, $\psi_i : \mathbf{R} \to \mathbf{R}$ is the queue size penalty function for node i, and $U : \mathbf{R}^n \to \mathbf{R}$ is the utility measure function for a given source-sink rate vector s.

Typically encountered flow cost functions are

$$\phi_j(x_j(t)) = \frac{|x_j(t)|}{c_j - |x_j(t)|}, \qquad (1)$$

$$\phi(x(t)) = \max_j \left\{ \frac{|x_j(t)|}{c_j} \right\}$$

where the domain of ϕ is $\mathbf{dom}\,\phi_j = (-c_j, c_j)$. The first function gives the expected waiting time in $M/M/1$ queue, while the second function gives the maximum link utilization, see (Bertsekas and Gallager, 1991).

Typical queue penalty functions ψ_i are linear or quadratic functions, where later one heavily penalizes buildup of very large queues.

Utility functions U represent the user utility for different source-sink flows. They show willingness of the user to pay for an additional amount of network bandwidth.

An equivalent (relaxed) formulation of the general problem will be considered in this paper. We will assume that each customer (node) has an exact network rate agreement, and therefore we can remove utilization functions from our objective and eliminate source rate constraints. We will choose the flow cost functions ϕ_j such that they act as barrier functions for $x_j(t)$ feasibility. For example, the flow cost functions from equation (1) can be defined to be finite inside of their domain $(-c, c)$ and infinite outside. Therefore, if we start in the feasible flow region, we will always stay feasible, and the link capacity constraints will be automatically enforced. Final relaxation is that we will not limit queue sizes, since we want to observe queues behavior especially when they become unbounded. In practical systems, queues will be finite and they will start dropping packets when they become over-saturated. Therefore, our final problem formulation is

$$\begin{array}{ll} \text{minimize} & \sum_{t=t_i}^{t_f} \Big[\sum_{j=1}^{p} \phi_j \left(x_j(t)\right) \\ & \qquad + \sum_{i=1}^{n} \psi_i \left(q_i(t)\right) \Big] \\ \text{subject to} & q(t+1) = q(t) + s(t) - Ax(t), \\ & \qquad q(t) \succeq 0. \end{array} \qquad (2)$$

Since we will only consider convex flow cost and queue size penalty functions, this is a convex optimization problem; convex optimization topics are beautifully treated in (Boyd and Vandenberghe, 2003). Since we have a convex optimization problem, there exists a global optimal solution which we will seek to find using a dynamical and distributed algorithm.

3 DUAL METHODS

In order to gain some insight into problem (2) in dynamical settings, we will first investigate its solution in a static case. We formulate the static-time optimal network flow problem by setting all queues to zero for all the time (basically removing queues from the system) and thus only enforcing the flow conservation

constraint in the network.

$$\begin{array}{ll} \text{minimize} & \sum_{j=1}^{p} \phi_j(x_j) \\ \text{subject to} & Ax = s. \end{array}$$

The Lagrangian for this problem is

$$\begin{aligned} L(x,\nu) &= \sum_{j=1}^{p} \phi_j(x_j) + \nu^T(s - Ax) \\ &= \sum_{j=1}^{p} (\phi_j(x_j) - \Delta\nu_j x_j) + \nu^T s, \end{aligned}$$

where we interpret the Lagrangian dual variable ν_i as a *potential* at node i, and $\Delta\nu_j$ denotes *potential difference* across link j. For more details about the optimal network flow problem, and the following material on dual decomposition and subgradient methods please refer to the manuscript (Boyd et al., 2003).

3.1 Dual network flow problem

The dual function is defined as the infimum of Lagrangian over the primal problem variables, and we have

$$\begin{aligned} d(\nu) &= \inf_x L(x,\nu) \\ &= \sum_{j=1}^{p} \inf_{x_j} (\phi_j(x_j) - \Delta\nu_j x_j) + \nu^T s \\ &= -\sum_{j=1}^{p} \phi_j^*(\Delta\nu_j) + \nu^T s, \end{aligned}$$

where ϕ_j^* is the conjugate function of ϕ_j.

We express the dual network flow problem as

$$\text{maximize} \quad d(\nu) = -\textstyle\sum_{j=1}^{p} \phi_j^*(\Delta\nu_j) + \nu^T s,$$

which is an unconstrained maximization of a concave function and we can use appropriate optimization algorithms to obtain an optimal dual solution $\nu^\star$.

3.2 Subgradient methods

Since the dual function is a concave, but possibly nondifferentiable function, we will use a subgradient-based method for its optimization. We will first derive an expression for the subgradients of a negative dual function.

Consider a dual function for the primal problem $\min_x f_0(x)$ subject to the equality constraints $f_i(x) = 0$, where $i = 1, \ldots, n$. We will assume f_0 is strictly convex, and denote,

$$x^*(\nu) = \operatorname*{argmin}_z (f_0(z) + \nu_1 f_1(z) + \cdots + \nu_n f_n(z))$$

so the dual function is

$$d(\nu) = f_0(x^*(\nu)) + \nu_1 f_1(x^*(\nu)) + \cdots + \nu_n f_n(x^*(\nu)).$$

Then, a subgradient of the negative dual function $-d$ at ν is given by $g_i = -f_i(x^*(\nu))$. The dual optimization method will consist of maximizing the dual function by stepping in the direction of the subgradient g. Thus, we obtain the subgradient method optimization rules for the dual problem:

$$x^{(k)} = x^*(\nu^{(k)}), \qquad \nu_i^{(k+1)} = \nu_i^{(k)} + \alpha_k f_i(x^{(k)}).$$

In the case of optimal network flow problem (which only has equality constraints $Ax = s$), subgradient at ν is

$$g = Ax^*(\Delta\nu) - s.$$

The ith component of subgradient is $g_i = a_i^T x^*(\Delta\nu) - s_i$, which is the *excess flow* at node i, but also the amount of data that q_i would accumulate after that iteration if we did not remove the queues.

Finally, the original optimal network flow problem can be solved by applying the subgradient method optimization to its dual problem, and then recovering the primal solutions after the algorithm converges. The algorithm's main steps are outlined below:

$$\begin{aligned} x_j &:= x_j^*(\Delta\nu_j) \\ g_i &:= a_i^T x - s_i \\ \nu_i &:= \nu_i - \alpha g_i \end{aligned}$$

where $x_j^*(\Delta\nu_j) = \operatorname{argmin}_{x_j} (\phi_j(x_j) - \Delta\nu_j x_j)$.

The method proceeds as follows. Given the current value of node potentials, the flow for each link is locally calculated. We then compute the flow surplus at each node. Again, this is local; to find the flow surplus at node i, we only need to know the flows on the links that enter or leave node i. Finally, we update the potentials based on the current flow surpluses. The update is very simple: we increase the potential at a node with a positive flow surplus, which will result in reduced flow into the node. Provided the step length α can be computed locally, the algorithm is distributed; the links and nodes only need information relating to their adjacent flows and potentials. There is no need to know the global topology of the network, or any other nonlocal information, such as what the flow cost functions are. Since we use the simplest subgradient update method with constant step sizes α, this algorithm is *completely distributed*.

Also since we use constant step sizes, this algorithm (if stable) converges to the optimal solution (or precisely, it converges to the ball of some arbitrary radius R related to step size α and centered at the optimal solution). For more details about this algorithm and its performance, please see (Boyd et al., 2003).

3.3 Optimal network flow in dynamic setting

We can apply the optimal network flow algorithm derived in the previous subsection to the problems in a

dynamic setting. Note that now each iteration step k is the actual time t. The subgradient method finds the optimal solution for network flows that satisfies the flow conservation constraint; however, queues accumulate the excess flow data and are not drained after this initial build-up of data. This scenario is presented in figures 1 and 2.

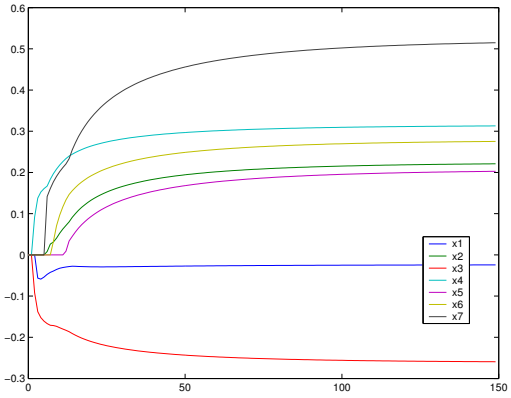

Figure 1: Flows $x_j(t)$ vs time t with $\alpha = 2.5$.

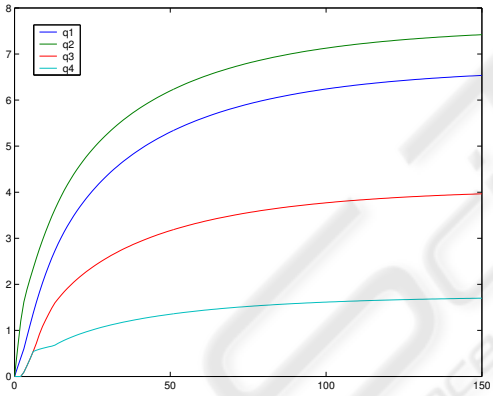

Figure 2: Queue sizes $q_i(t)$ vs time t with $\alpha = 2.5$.

We see that link flows x_j converge to their optimal values (figure 1), while the queues accumulate the excess flow until the flow conservation equality is satisfied (figure 2).

4 BUNDLE SUBGRADIENT METHOD AND ALGORITHM

We consider a simple modification for the static-time dual subgradient method algorithm derived in the previous section. In order to discharge the queues we will linearly add queue lengths $q_i(t)$ to the potential updates $\nu_i(t)$ in the algorithm. Therefore, our new potential update step is:

$$\nu_i(t+1) = \nu_i(t) - \alpha^{(t)} g_i(t) + \beta^{(t)} q_i(t).$$

This modification can be interpreted as follows; we adjust the potential at each node proportional to its queue size in order to increase the flow out of that node, while we still step in the direction of negative dual subgradient, which will reduce the excess flow (future queue accumulation) for that node. If the queue size is very large, then we greatly influence the link flow out of that node (its queue discharge), whereas if the queue size is small the link flow will be mainly determined by balancing of excess flow equations. This rule should simultaneously decrease the excess flow and queue lengths, as experimentally verified in section 6. Also note that our modification has preserved the *distributed* nature of the algorithm; we only require local queue information for each node.

The modified algorithm was applied to the static case simulation, and the results shown in figures 3 and 4 verify that queues are discharged after the initial build-up.

We have obtained an efficient queue regulation using this new algorithm, and next we will analyze its theoretical performance.

4.1 Queues are subgradients

As we have already mentioned, queues are accumulations of previous excess flows, which are subgradients at the previous time points (iterations).

$$q(t+1) - q(t) = Ax^*(\Delta\nu(t)) - s(t) = g(t).$$

Assuming zero initial conditions at the start of the network operation, then at time $t = 0, 1, \ldots, N$, we have

$$\begin{aligned} q(1) &= g(0) = Ax^*(\Delta\nu(0)) - s(0), \\ q(2) &= g(0) + Ax^*(\Delta\nu(1)) - s(1), \\ &\vdots \\ q(N+1) &= \sum_{t=0}^{N-1} g(t) + Ax^*(\Delta\nu(N)) - s(N). \end{aligned}$$

The original subgradient method iteration for dual network flow problem is

$$\nu^{(k+1)} = \nu^{(k)} - \alpha_k g^{(k)},$$

where $\nu^{(k)}$ is the voltage at kth iteration, $g^{(k)}$ is any subgradient of $-d$ at $\nu^{(k)}$, and $\alpha_k > 0$ is the kth step size.

At each iteration, the subgradient method uses only the *current* subgradient g (the current excess flow or queue increment at time k).

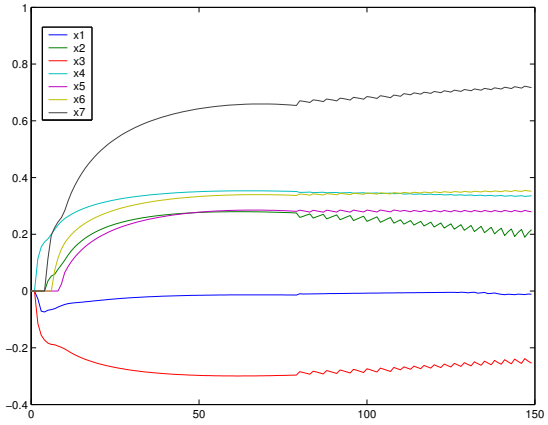

Figure 3: Flows $x_j(t)$ vs t with $\alpha = 2.5$ and $\beta = 0.2$.

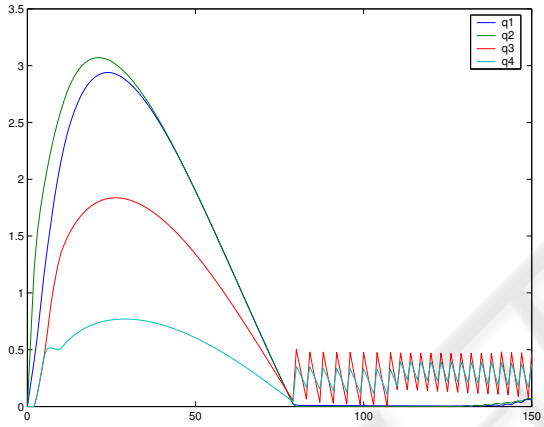

Figure 4: Queues $q_i(t)$ vs t with $\alpha = 2.5$ and $\beta = 0.2$.

4.2 Subgradient bundle method

We will define the subgradient bundle method iteration as

$$\nu^{(k+1)} = \nu^{(k)} - \alpha_k w^{(k)},$$

where

$$w^{(k)} = \begin{cases} g^{(k)} & k = 0, \\ g^{(k)} + \beta_k w^{(k-1)} & k > 1. \end{cases}$$

Here, $\nu^{(k)}$ is the voltage at kth iteration, $g^{(k)}$ is any subgradient of $-d$ at $\nu^{(k)}$, and $w^{(k)}$ is the *subgradient bundle* (or memory) of $-d$ at kth step. Now we have two step size constants, $\alpha_k > 0$ is the kth step size for excess flow update, and $\beta_k > 0$ is the kth step size for subgradient bundle. This new algorithm is a variant of the subgradient bundle method, originally developed independently by Lamarechal and Wolfe, see (Lamarechal and Wolfe, 1975).

Switching from iterations k to the actual time t, we have

$$\begin{aligned} w(t) &= g(t) + \beta^{(t)} w(t-1) \\ &= g(t) + \beta^{(t)} \tilde{q}(t) \\ &= \tilde{q}(t+1) \end{aligned}$$

where $\tilde{q}(t) = cq(t)$, $c \in \mathbf{R}_+$, only if $\beta^{(t)}$ is a constant step size. If we fix $\beta = 1$ for all t, then $w(t)$ is precisely $q(t+1)$, the queue size at the end of the current time slot.

Thus, if we take the queue penalty functions to be linear, $\psi(q(t)) = q(t)$, we are simultaneously solving the optimal flow problem *and* minimizing queue lengths. We can claim that we are searching for the optimal point of our problem, where the subgradient is equal to zero, and therefore the queue sizes will be driven to zero.

5 CONVERGENCE PROOF

In this section, we derive the convergence results for the bundle subgradient method applied to the optimal network flow problem.

Let $\nu^\star$ be an optimal dual solution. Define convergence ranges for step sizes α_k and β_k as

$$0 < \alpha_k \leq \frac{d(\nu^\star) - d(\nu_k)}{\|w^{(k)}\|^2},$$

and

$$\beta_k = \begin{cases} -\gamma \frac{(g^{(k)})^T w^{(k-1)}}{\|w^{(k-1)}\|^2} & \text{if } (g^{(k)})^T w^{(k-1)} < 0, \\ 0 & \text{otherwise,} \end{cases}$$

where $\gamma \in [0, 2]$.

Using induction we will prove that for step sizes α_k and β_k satisfying the above convergence conditions, we have

$$\|\nu^\star - \nu^{(k+1)}\| < \|\nu^\star - \nu^{(k)}\| \qquad (3)$$

and furthermore,

$$\frac{(\nu^\star - \nu^{(k)})^T w^{(k)}}{\|w^{(k)}\|} \geq \frac{(\nu^\star - \nu^{(k)})^T g^{(k)}}{\|g^{(k)}\|}, \qquad (4)$$

so the angle between $w^{(k)}$ and $\nu^\star - \nu^{(k)}$ is no larger than the angle between $g^{(k)}$ and $\nu^\star - \nu^{(k)}$.

Proof: Let $d^\star = d(\nu^\star)$ be the optimal value of the dual function. By induction we will first show that

$$(\nu^\star - \nu^{(k)})^T w^{(k)} \geq (\nu^\star - \nu^{(k)})^T g^{(k)}, \qquad (5)$$

for all k. We have $w^{(0)} = g^{(0)}$, thus it holds for $k = 0$. Assuming it holds for k, we will prove it for $k + 1$. Since $w^{(k)} = g^{(k)} + \beta_k w^{(k-1)}$, we have

$$\begin{aligned} (\nu^\star - \nu^{(k+1)})^T w^{(k+1)} &= (\nu^\star - \nu^{(k+1)})^T g^{(k+1)} + \\ &\quad \beta_k (\nu^\star - \nu^{(k+1)})^T w^{(k)} \end{aligned}$$

and using concavity of the dual function d and step size convergence conditions, we obtain,

$$(\nu^\star - \nu^{(k+1)})^T w^{(k+1)} \geq (\nu^\star - \nu^{(k+1)})^T g^{(k+1)},$$

and hence equation (5) holds for all k.

This inequality after some manipulations leads to the conclusion that equation (3) holds true.

From the definitions of $w^{(k)}$ and β_k we have

$$\begin{aligned} \|w^{(k)}\|^2 - \|g^{(k)}\|^2 &= \|g^{(k)} + \beta_k w^{(k-1)}\|^2 - \|g^{(k)}\|^2 \\ &= (\beta_k)^2 \|w^{(k-1)}\|^2 + \\ &\quad 2\beta_k (g^{(k)})^T w^{(k-1)} \\ &= (2-\gamma)\beta_k (g^{(k)})^T w^{(k-1)} \\ &\leq 0, \end{aligned}$$

and therefore $\|w^{(k)}\| \leq \|g^{(k)}\|$, which combined with equation (5) implies that

$$\frac{(\nu^\star - \nu^{(k)})^T w^{(k)}}{\|w^{(k)}\|} \geq \frac{(\nu^\star - \nu^{(k)})^T g^{(k)}}{\|g^{(k)}\|},$$

for all k, which proves that the convergence of the bundle subgradient method is never worse than convergence of the regular subgradient method. This proof was inspired by a homework problem in (Bertsekas, 1999).

6 SIMULATIONS

6.1 Simulations setup

All simulations were performed using Matlab. Our test network topology is presented in figure 5, it has $n = 5$ nodes and $p = 7$ links. The node-link incidence matrix for this network is

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix}.$$

Link capacity vector was chosen to be `c = [.5; .5; .5; .5; .5; .5; 1]`, and dynamical source rates $s(t)$ behaviour versus time is presented in figure 6. We have chosen node 5 to be the destination (sink) node for all of the traffic originating at other nodes.

Source rates $s_2(t)$ and $s_3(t)$ oversaturate the network since the maximum capacity of links on their paths is equal to 1; therefore, for time period $t = [50, 150]$ we are guaranteed to have increasing queues and a congested network.

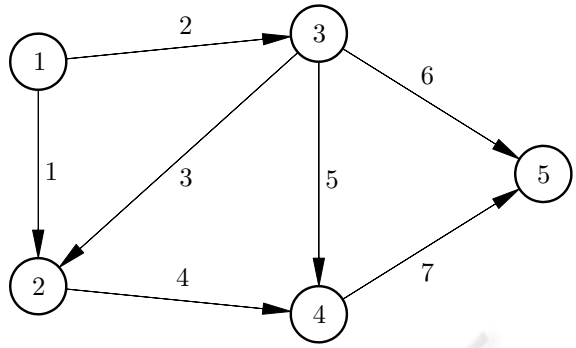

Figure 5: Test network with 5 nodes and 7 links.

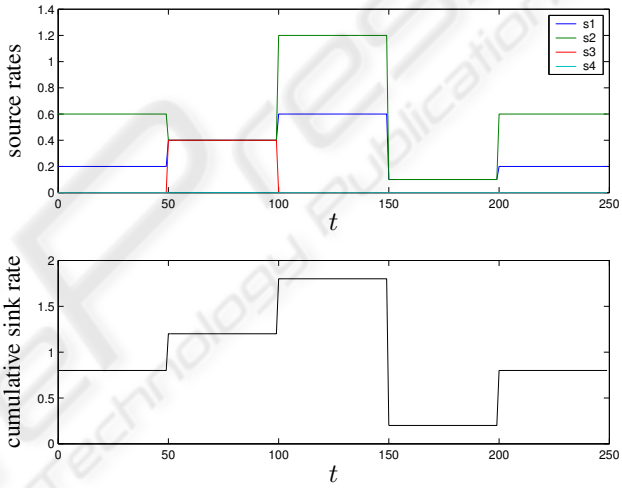

Figure 6: Plot of source rates versus time.

6.2 Algorithm performance

Performance with dynamical source rates is shown in figures 7 and 8. The bundle algorithm with constant step sizes $\alpha = 2.5$ and $\beta = 0.2$ performs very well. We note that it dynamically adjusts link flows $x_j(t)$ as the network load changes. We notice that during the time when network is over-saturated, the queue lengths grow; however, as soon as user traffic becomes feasible again, the queues are again driven to zero.

Figures 9 and 10 show algorithm performance with a more aggressive β parameter. We set constant step sizes $\alpha = 2.5$ and $\beta = 1$; therefore, we place more weight on having small queue sizes. It is clear that average queue size in the system is much smaller. However, as we use larger step sizes our system becomes less stable. There is a clear trade-off between queue size regulation and algorithm convergence and smoothness.

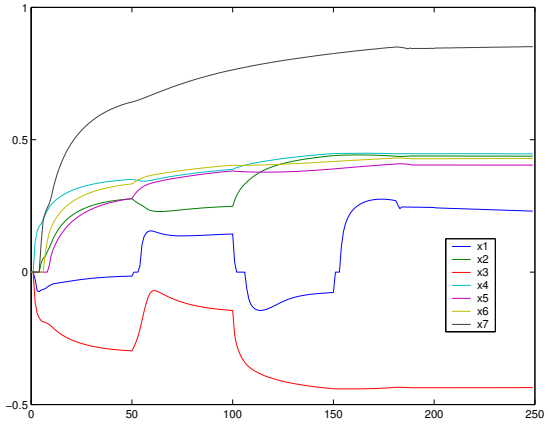

Figure 7: Flows $x_j(t)$ vs t with $\alpha = 2.5$ and $\beta = 0.2$.

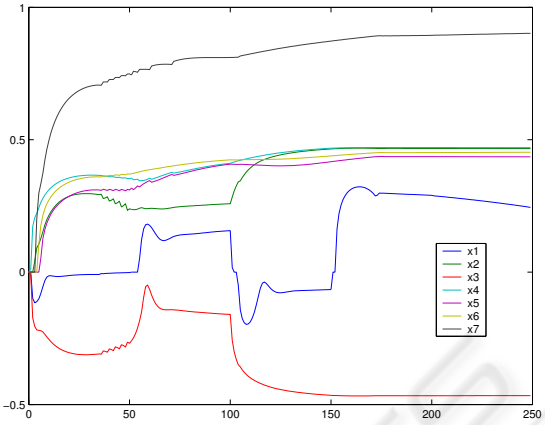

Figure 9: Flows $x_j(t)$ vs t with $\alpha = 2.5$ and $\beta = 1$.

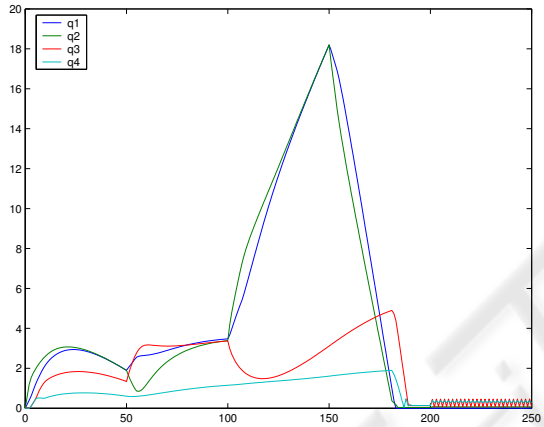

Figure 8: Queues $q_i(t)$ vs t with $\alpha = 2.5$ and $\beta = 0.2$.

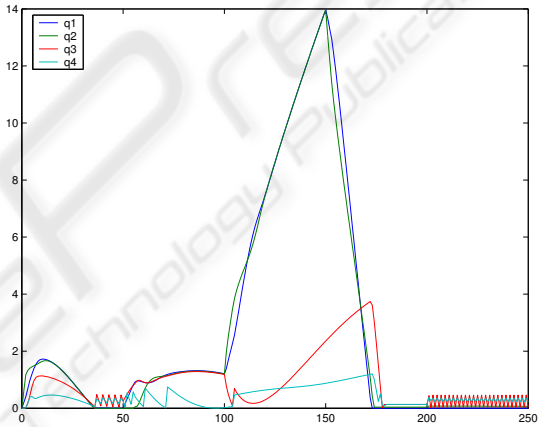

Figure 10: Queues $q_i(t)$ vs t with $\alpha = 2.5$ and $\beta = 1$.

7 CONCLUSION

Subgradient bundle methods are not a new idea; theoretical work in this area was started around 1975 by Lamarechal and Wolfe. Years of research culminated in the development of an optimization algorithm suite that is commonly referred to as the bundle methods, and which is in wide use today for many non-differentiable optimization problems.

In this paper we have derived a bundle-like subgradient method, which is completely distributed since it uses only locally available network information, and which simultaneously routes the network traffic in a dynamic manner and regulates queue sizes across the network. Thus, we have achieved our goal of finding a distributed algorithm for joint dynamic routing and queue management. We are speculating that this is the first time that bundle subgradient ideas have been applied to the network routing and the solution of the optimal network flow dual problem.

We have proved the convergence of our proposed algorithm and stated convergence conditions for constant step size update rules. Algorithm performance and theoretical results were successfully verified using Matlab simulations.

In the future, we would like to extend our algorithm to the changing step size rules such as the diminishing step size, and try to obtain absolute performance limits for a bundle-type subgradient network routing algorithm. We would also like to unify our sugradient convergence proof method with works by (Athuraliya and Low, 2000) and (Imer and Basar, 2003).

REFERENCES

Athuraliya, S. and Low, S. (2000). Optimization flow control – II: Implementation. `http://netlab.caltech.edu`.

Bertsekas, D. P. (1999). *Nonlinear Programming*. Athena Scientific, second edition.

Bertsekas, D. P. and Gallager, R. (1991). *Data Networks*. Prentice-Hall, second edition.

Boyd, S. and Vandenberghe, L. (2003). *Convex Optimization*. Cambridge University Press.

Boyd, S., Xiao, L., and Mutapcic, A. (2003). Subgradient methods - EE392o class notes, Stanford University. `http://www.stanford.edu/class/ee392o/subgrad_method.pdf`.

Imer, O. and Basar, T. (2003). Dynamic optimization flow control. In *IEEE Conference on Decision and Control*, pages 2082–2087.

Lamarechal and Wolfe (1975). Nondifferentiable optimization. In *Mathematical Programming Study*, volume 3.

Segall, A. (1977). The modeling of adaptive routing in data-communication networks. *IEEE Transactions on Communications*, 25(1):85–95.