An Interior-Point Method for Large Scale Network Utility Maximization

Argyrios Zymnis, Nikolaos Trichakis, Stephen Boyd, and Dan O'Neill

August 6, 2007

Abstract

We describe a specialized truncated-Newton primal-dual interior-point method that solves large scale network utility maximization problems, with concave utility functions, efficiently and reliably. Our method is not decentralized, but easily scales to problems with 10^6 flows and links. We compare our method to a standard decentralized algorithm based on dual decomposition, and show by example that our method converges significantly faster for problems with congested networks or long routes. We describe an extension to problems that take into account delay or latency in the objective.

1 Introduction

We consider a network that supports a set of flows, each of which has a nonnegative flow rate, and an associated utility function. Each flow passes over a route, which is a subset of the edges of the network. Each edge has a given capacity, which is the maximum total traffic (the sum of the flow rates through it) it can support. The network utility maximization (NUM) problem is to choose the flow rates to maximize the total utility, while respecting the edge capacity constraints [Sri04, Ber98]. We consider the case where all utility functions are concave, in which case the NUM problem is a convex optimization problem.

A standard technique for solving NUM problems is based on dual decomposition [DW60, Sho85]. This approach yields fully decentralized algorithms, that can scale to very large networks. Dual decomposition was first applied to the NUM problem in [KMT97], and has led to an extensive body of research on distributed algorithms for network optimization [LL99, CLCD07, PC06] and new ways to interpret existing network protocols [Low03].

In this paper we describe a specialized primal-dual interior-point method [BV04, Wri97] for the NUM problem. The method is not decentralized, since it requires a few inner products at each step, but we show by example that it exhibits faster convergence than dual decomposition, especially when the network is congested (*i.e.*, has bottleneck links), or there are long routes. The method scales to very large networks, and has modest memory requirements (only a small multiple of the size of the problem data).

Another advantage over dual decomposition is that the method handles utility functions that are not strictly concave. In particular, it can handle linear utility functions. These are interesting because the optimal flow rates can be zero. In this case we can interpret NUM as a method for carrying out admission control; zero flow rates correspond to flows that are denied admission to the network.

In §2, we formally introduce the NUM problem, give its dual and the optimality conditions, and describe the standard dual decompositin algorithm for NUM. We describe our method in §3, and we give some numerical examples in §4. In §5 we describe an extension of the basic NUM problem to take latency into account, and show how it can be handled by our method.

2 Problem Formulation

There are n flows in a network, each of which is associated with a fixed route, *i.e.*, some subset of m links. Each flow has a nonnegative *rate*, which we denote $f_1, \ldots, f_n$. With the flow j we associate a utility function $U_j : \mathbf{R} \to \mathbf{R}$, which is concave and twice differentiable, with $\mathbf{dom}\, U_j \subseteq \mathbf{R}_+$. The utility derived by a flow rate f_j is given by $U_j(f_j)$. The total utility associated with all the flows is then $U(f) = U_1(f_1) + \cdots + U_n(f_n)$.

The total traffic on a link in the network is the sum of the rates of all flows that utilize that link. We can express the link traffic compactly using the *routing* or *link-route* matrix $R \in \mathbf{R}^{m \times n}$, defined as

$$R_{ij} = \begin{cases} 1 & \text{flow } j\text{'s route passes over link } i \\ 0 & \text{otherwise.} \end{cases}$$

Each link in the network has a (positive) *capacity* $c_1, \ldots, c_m$. The traffic on a link cannot exceed its capacity, *i.e.*, we have $Rf \le c$, where $\le$ is used for componentwise inequality.

The NUM problem is to choose the rates to maximize total utility, subject to the link capacity and the nonnegativity constraints:

$$\begin{array}{ll} \text{maximize} & U(f) \\ \text{subject to} & Rf \le c, \quad f \ge 0, \end{array} \tag{1}$$

with variable $f \in \mathbf{R}^n$. This is a convex optimization problem and can be solved by a variety of methods. We say that f is *primal feasible* if it satisfies $Rf \le c$, $f \ge 0$.

2.1 Dual problem

The dual of problem (1) is

$$\begin{array}{ll} \text{minimize} & \lambda^T c + \sum_{j=1}^n (-U_j)^*(-r_j^T \lambda) \\ \text{subject to} & \lambda \ge 0, \end{array} \tag{2}$$

where $\lambda \in \mathbf{R}_+^m$ is the dual variable associated with the capacity constraint of problem (1), r_j is the jth column of R and $(-U_j)^*$ is the conjugate of the negative jth utility function [BV04, §3.3],

$$(-U_j)^*(a) = \sup_{x \geq 0}(ax + U_j(x)).$$

We say that λ is *dual feasible* if it satisfies $\lambda \geq 0$ and $\lambda \in \cap_{j=1}^n \mathbf{dom}(-U_j)^*$.

Let us work out the dual more explicitly for a NUM problem with mixed logarithmic and linear utility functions: $U_j(f_j) = \log f_j$ for $j = 1, \ldots, k$, and $U_j(f_j) = w_j f_j$ for $j = k+1, \ldots, n$, where $w_j > 0$. The (primal) NUM problem is

$$\begin{array}{ll}
\text{maximize} & \sum_{j=1}^k \log f_j + \sum_{j=k+1}^n w_j f_j \\
\text{subject to} & Rf \leq c, \quad f \geq 0.
\end{array}$$

For $U_j(f_j) = \log f_j$, we have $(-U_j)^*(a) = -\log(-a) - 1$. For $U_j(f) = w_j f_j$, we have

$$(-U_j)^*(a) = \left\{ \begin{array}{ll} 0 & -a \geq w_j \\ \infty & \text{otherwise.} \end{array} \right.$$

The dual NUM problem is then

$$\begin{array}{ll}
\text{minimize} & \lambda^T c - \sum_{j=1}^k \left(\log(r_j^T \lambda) + 1 \right) \\
\text{subject to} & r_j^T \lambda \geq w_j, \quad j = k+1, \ldots, n \\
& \lambda \geq 0.
\end{array}$$

(Here we have included the dual objective terms associated with the linear utilities as explicit inequality constraints.)

The solution $f^\star$ of the primal NUM problem can be recovered from the solution $\lambda^\star$ of the dual NUM problem provided all the functions U_j are strictly concave. In this case we have

$$f_j^\star = \operatorname*{argmax}_{x \geq 0} \left(U_j(x) - x(r_j^T \lambda^\star) \right).$$

For more on this see [BV04, §5.5.5]. For the particular case $U_j(f_j) = \log f_j$, we have $f_j^\star = 1/(r_j^T \lambda^\star)$.

2.2   Dual decomposition

Dual decomposition [DW60, Sho85, KMT97, LL99] is a projected (sub)gradient algorithm for solving problem (2), in the case when all utility functions are strictly concave. We start with any positive λ, and repeatedly carry out the update

$$\begin{array}{rl}
f_j & := \operatorname*{argmax}_{x \geq 0} \left(U_j(x) - x(r_j^T \lambda) \right), \quad j = 1, \ldots, n, \\
\lambda & := (\lambda - \alpha (c - Rf))_+,
\end{array}$$

where $\alpha > 0$ is the step size, and x_+ denotes the entrywise nonnegative part of the vector x. It can be shown that for small enough α, f and λ will converge to $f^\star$ and $\lambda^\star$, respectively,

provided all U_j are differentiable and strictly concave. The term $s = c - Rf$ appearing in the update is the *slack* in the link capacity constraints (and can have negative entries during the algorithm execution). It can be shown that the slack is exactly the gradient of the dual objective function.

Dual decomposition is a distributed algorithm. Each flow is updated based on information obtained from the links it passes over, and each link dual variable is updated based only on the flows that pass over it.

Note that dual decomposition, at least in its simple form described above, cannot deal with utility functions that are not strictly concave. In particular, this algorithm would not be able to solve a NUM problem with mixed logarithmic and linear utility functions.

2.3   Optimality conditions

Let $\mu \in \mathbf{R}_+^n$ be the dual variable associated with the nonegativity constraint of problem (1). The optimality conditions [BV04, §5.5] of this problem can then be formulated as

$$
\begin{aligned}
-\nabla U(f^\star) + R^T \lambda^\star - \mu^\star &= 0 \\
\mathbf{diag}(\lambda^\star)s^\star &= 0 \\
\mathbf{diag}(\mu^\star)f^\star &= 0 \\
f^\star, s^\star, \lambda^\star, \mu^\star &\geq 0,
\end{aligned}
$$

where $s^\star = c - Rf^\star$ is the optimal link capacity slack. The second and third equations are the *complementary slackness* conditions.

2.4   Duality gap

Given a primal feasible f and a dual feasible λ we define the *duality gap* η as the difference between the respective primal and dual objectives, *i.e.*,

$$
\eta = \lambda^T c + \sum_{j=1}^n (-U_j)^*(-r_j^T \lambda) - U(f).
$$

The duality gap, which is always nonnegative, is a bound on suboptimality for a primal-dual pair (f, λ). It is therefore often used as a (nonheuristic) stopping criterion for optimization algorithms [BV04, NW99].

Given a primal feasible f and any $\lambda, \mu \geq 0$ we also define the *surrogate duality gap* $\hat{\eta}$ as

$$
\hat{\eta} = s^T \lambda + f^T \mu.
$$

It can be shown that if λ and μ are dual feasible then $\eta = \hat{\eta}$.

In cases where U is strictly concave, $\lambda \geq 0$ is the only requirement for dual feasibility. In cases where U is not strictly concave (such as in the mixed linear and logarithmic utility example considered above) dual feasibility is a more complicated condition. In these cases it is more convenient to use $\hat{\eta}$ rather than η as a stopping criterion.

3   Primal-Dual Interior-Point Method

The primal-dual interior-point method is based on using a Newton step, applied to a suitably modified form of the optimality conditions. The modification is parametrized by a parameter t, which is adjusted during the algorithm based on progress, as measured by the actual duality gap (if it is available) or a surrogate duality gap (when the actual duality gap is not available).

We first describe the search direction. We modify the complementary slackness conditions, to obtain the modified optimality conditions

$$
\begin{aligned}
-\nabla U(f) + R^T \lambda - \mu &= 0 \\
\mathbf{diag}(\lambda)s &= (1/t)\mathbf{1} \\
\mathbf{diag}(\mu)f &= (1/t)\mathbf{1},
\end{aligned}
$$

where $t > 0$ is a parameter that sets the accuracy of the approximation. (As $t \to \infty$, we recover the optimality conditions for the NUM problem.) Here we implicitly assume that $f, s, \lambda, \mu > 0$. The modified optimality conditions can be compactly written as $r_t(f, \lambda, \mu) = 0$, where

$$
r_t(f, \lambda, \mu) = \begin{bmatrix} -\nabla U(f) + R^T \lambda - \mu \\ \mathbf{diag}(\lambda)s - (1/t)\mathbf{1} \\ \mathbf{diag}(\mu)f - (1/t)\mathbf{1} \end{bmatrix}.
$$

The primal-dual search direction is the Newton step for solving the nonlinear equations $r_t(f, \lambda, \mu) = 0$. If $y = (f, \lambda, \mu)$ denotes the current point, the Newton step $\Delta y = (\Delta f, \Delta \lambda, \Delta \mu)$ is characterized by the linear equations

$$
r(y + \Delta y) \approx r_t(y) + Dr_t(y)\Delta y = 0,
$$

which, written out in more detail, are

$$
\begin{bmatrix} -\nabla^2 U(f) & R^T & -I \\ -\mathbf{diag}(\lambda)R & \mathbf{diag}(s) & 0 \\ \mathbf{diag}(\mu) & 0 & \mathbf{diag}(f) \end{bmatrix} \begin{bmatrix} \Delta f \\ \Delta \lambda \\ \Delta \mu \end{bmatrix} = -r_t(f, \lambda, \mu). \tag{3}
$$

During the algorithm, the parameter t is increased, as the primal and dual variables approach optimality. When we have easy access to a dual feasible point during the algorithm, we can make use of the exact duality gap η to set the value of t; in other cases, we can use the surrogate duality gap $\hat{\eta}$.

The primal-dual interior point algorithm [BV04, §11.7], [Wri97] for solving problem (1) proceeds as follows.

> **given** tolerance $\epsilon > 0$, parameters $\alpha \in (0, 1/2)$, $\beta \in (0, 1)$, $\kappa > 1$
> initialize: y with $Rf < c$; $y > 0$
> **while** $\hat{\eta}/n > \epsilon$
> $\quad t := \kappa m / \hat{\eta}$

compute Δy from (3)
find the smallest integer k for which

$\qquad y + \beta^k \Delta y > 0,\ R(f + \beta^k \Delta f) < c,$

$\qquad \|r_t(y + \beta^k \Delta y)\|_2 \leq \|r_t(y)\|_2 + \alpha\beta^k \left.\frac{d}{d\gamma}\|r_t(y + \gamma\Delta y)\|_2\right|_{\gamma=0}$

update:

$\qquad y := y + \beta^k \Delta y$

Typical parameter values are $\alpha = 0.01$, $\beta = 0.5$, and κ between 2 and 10.

To find the primal-dual search direction, *i.e.*, solve equation (3), we eliminate $\Delta\lambda$ and $\Delta\mu$, and solve the reduced (symmetric, positive definite) system of equations

$$(-\nabla^2 U(f) + D_1 + R^T D_2 R)\Delta f = \nabla U(f) + (1/t)v_1 - (1/t)R^T v_2, \qquad (4)$$

where

$$D_1 = \mathbf{diag}(\mu_1/f_1, \ldots, \mu_n/f_n), \qquad D_2 = \mathbf{diag}(\lambda_1/s_1, \ldots, \lambda_m/s_m)$$
$$v_1 = (1/f_1, \ldots, 1/f_n), \qquad v_2 = (1/s_1, \ldots, 1/s_m).$$

We then compute $\Delta\lambda$ and $\Delta\mu$ by back substitution, using the formulas

$$\Delta\mu = -\mu - D_1\Delta f + (1/t)v_1,$$
$$\Delta\lambda = D_2 R\Delta f - \lambda + (1/t)v_2.$$

The most expensive part of computing the primal-dual search direction is solving equation (4). For problems of modest size, *i.e.*, with m and n no more than 10^4, it can be solved using direct methods such as a sparse Cholesky decomposition. The cost depends on the problem size, the sparsity pattern of $R^T D_2 R$, and the details of the method (*e.g.*, the elimination ordering scheme used). (The sparsity pattern of $RD_2 R^T$ is directly related to the routes: $(R^T D_2 R)_{ij} \neq 0$ if and only if route i and j share a common link.) When a direct method is used to compute the primal-dual search direction, the algorithm typically converges to very high accuracy in a few tens of iterations. (In extensive numerical tests, we found no case in which more than 25 iterations was required to achieve high accuracy.)

For larger problem instances we can solve (4) *approximately*, using a preconditioned conjugate gradient (PCG) algorithm [Dem97, §6.6], [Kel95, chap. 2], [NW99, chap. 5]. When an iterative method is used to approximately solve a Newton system, the algorithm is referred to as an *inexact*, *iterative*, or *approximate* Newton method (see [Kel95, chap. 6] and its references). When an iterative method is used inside a primal-dual interior-point method, the overall algorithm is called a *truncated-Newton primal-dual interior-point method*. For details of the PCG algorithm, we refer the reader to the references cited above. Each iteration requires mutliplication of the matrix by a vector, and a few vector inner products.

For a truncated-Newton interior-point method to work well, we need a good preconditioner and a termination rule that yields search directions accurate enough to make good progress, using as few PCG steps as needed. In our method, we use a simple diagonal preconditioner. We terminate the PCG iterations after a limit N_{PCG} is reached, or if a

specified relative tolerance ϵ_{PCG} is reached. We adapt the tolerance according to the rule $\epsilon_{\text{PCG}} = \min\{0.1, \hat{\eta}/n\}$, so the PCG iterations terminate with relatively low tolerance early in the algorithm, and with higher accuracy as the algorithm proceeds. This termination rule is similar to the one used in [KKB07]. As initial point for the PCG iterations, we use the previous search direction.

In the truncated-Newton interior-point method, we never need to form the coefficient matrix $-\nabla^2 U(f) + D_1 + R^T D_2 R$; we only need to evaluate $(-\nabla^2 U(f) + D_1 + R^T D_2 R)z$ for some vector z at each PCG step. This can be done by multiplying z by the diagonal matrix $-\nabla^2 U(f) + D_1$, and by sequentially multiplying z by R, then D_2, and then R^T, and finally, adding the results. These operations are, in fact, decentralized. At each step of the PCG algorithm, however, two inner products have to be computed; these computations make the whole algorithm not decentralized.

The effort required in each PCG iteration of the truncated-Newton interior-point method is comparable (and quite similar to) the effort required in an iteration of the dual decomposition algorithm, which also requires matrix-vector multiplications by R and R^T in each step. The cost of the truncated-Newton interior-point method is best judged by the total number of PCG steps taken (and, in particular, not by the number of interior-point iterations).

4 Numerical Examples

We first demonstrate the performance of the primal-dual interior-point method on a small network example. In this case we use a direct method to compute the Newton step at each iteration.

To demonstrate the performance of the truncated-Newton primal-dual interior-point method, we study three additional examples. In the first two, all utility functions are logarithmic, so we can compare our method to dual decomposition. Our last example includes a mixture of flows with logarithmic and linear utilities. In this case dual decomposition cannot be used, since the utility function is not strictly concave.

We chose the step size in dual decomposition manually, to approximately maximize the convergence speed for each example. With suboptimal values of the step size more likely to be used in practice, the convergence of dual decomposition could be much slower.

4.1 Small network example

In our first example we look at the performance of our method on a small network. In this case we use a direct method (sparse Cholesky) to compute the Newton step at each iteration. The utility functions are all logarithmic, i.e., $U_j(f_j) = \log f_j$. There are $n = 10^3$ flows, and $m = 2 \cdot 10^3$ links. The elements of R are chosen randomly and independently, so that the average route length is 10 links. The link capacities c_i are chosen independently from a uniform distribution on $[0.1, 1]$.

For this particular example, there are about 10^4 nonzero elements in R (0.5% density) and about $7 \cdot 10^4$ nonzero elements in $D_1 + R^T D_2 R$ (3.5% density). The Cholesky factor
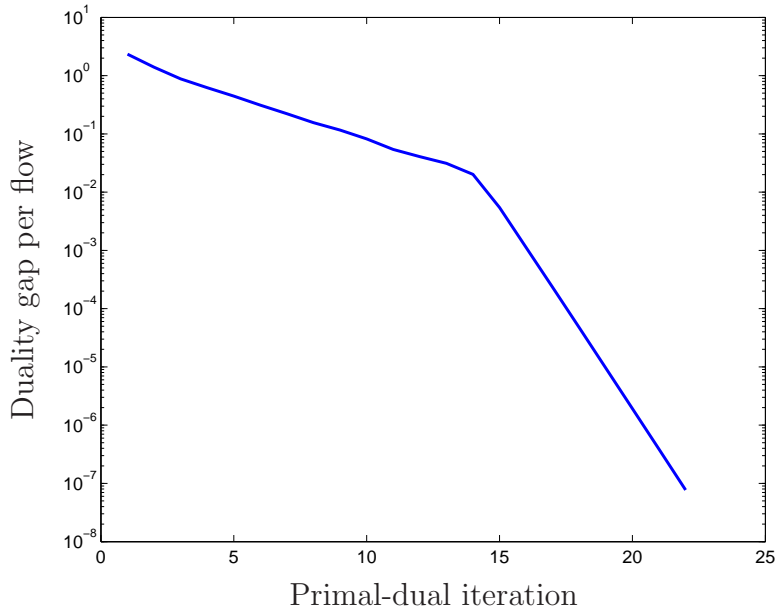
Figure 1: Convergence of primal-dual interior-point method for a small network example.

of $D_1 + R^T D_2 R$ has about $4 \cdot 10^5$ nonzero elements, using an aproximate minimum degree ordering. This represents 80% density, *i.e.*, the Cholesky factor is essentially dense. In this case, the only advantage to using sparse matrices, as opposed to dense matrices, is in forming the matrix $D_1 + R^T D_2 R$.

Figure 1 shows the convergence of the primal-dual interior-point method. We observed the same rapid convergence in many other examples, with different values of m and n, different sparsity patterns in the route matrix R, and different link capacities and utility functions.

4.2   Simple example

Our second example is too large to be solved using the primal-dual interior-point method with direct search direction computation, but is readily handled by our truncated-Newton primal-dual algorithm, and dual decomposition. The utility functions are all logarithmic: $U_j(f_j) = \log f_j$. There are $n = 10^5$ flows, and $m = 2 \cdot 10^5$ links. The elements of R and c are chosen as for the example in §4.1. For dual decomposition, we initialized all λ_i as 1. For the interior-point method, we initialized all λ_i and μ_i as 1. We initialize all f_j as γ, where we choose γ so that $Rf \leq 0.9c$.

Figure 2 shows the duality gap per flow as a function of iteration, for the two algorithms. (For the truncated-Newton method, iterations are PCG iterations.) Both perform well, with the truncated-Newton interior-point method achieving higher accuracy more quickly than
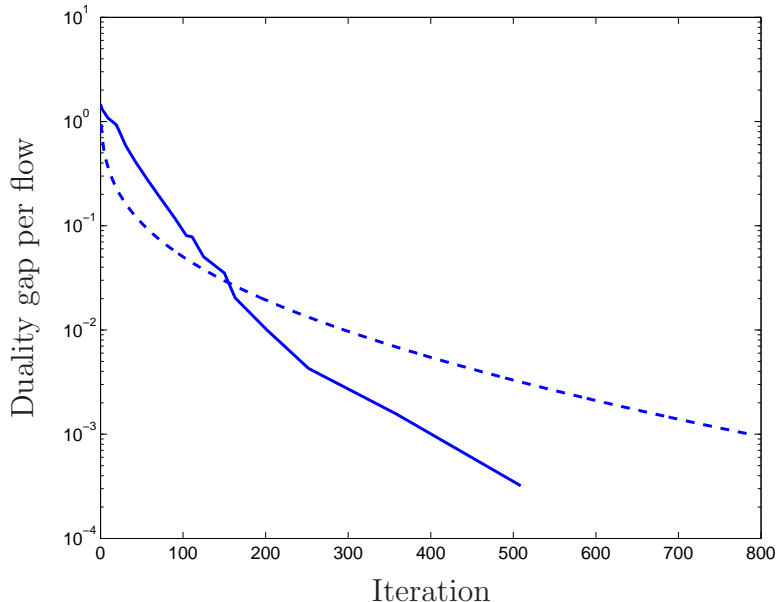
Figure 2: Convergence of truncated-Newton (solid) and dual decomposition (dashed) methods for a simple network.

dual decomposition. This behavior is typical for networks without large variations in the utility functions, with a modest number of flows passing through each link, and with no particular long flows. (These correspond to sparse rows and columns in R, respectively.)

4.3 Congested network

In this example the network is similar to the previous example, except that we increase the congestion of the last 200 links (*i.e.*, 0.01% of them), so that each such link is utilized on average by $3 \cdot 10^4$ flows. We also increase the typical number of links used by the last 100 flows from 10 to 1000. To compensate for the increased link congestion, we multiply the capacity of the bottleneck links by a constant factor, which depends on the number of flows utilizing such a link. This gives rise to some relatively dense rows and columns in R, when compared to the previous example. The utilities are all logarithmic.

Figure 3 shows the duality gap per flow as a function of iteration for such a congested network. As can be seen from this figure, the primal-dual algorithm is relatively unaffected by the increased congestion. In contrast, dual decomposition is progressing very slowly.

The difference in convergence is not unexpected, since dual decomposition is a first-order method, using a gradient search direction, while the interior-point method at least attempts (through the approximate solution of a Newton system) to use second-order information.
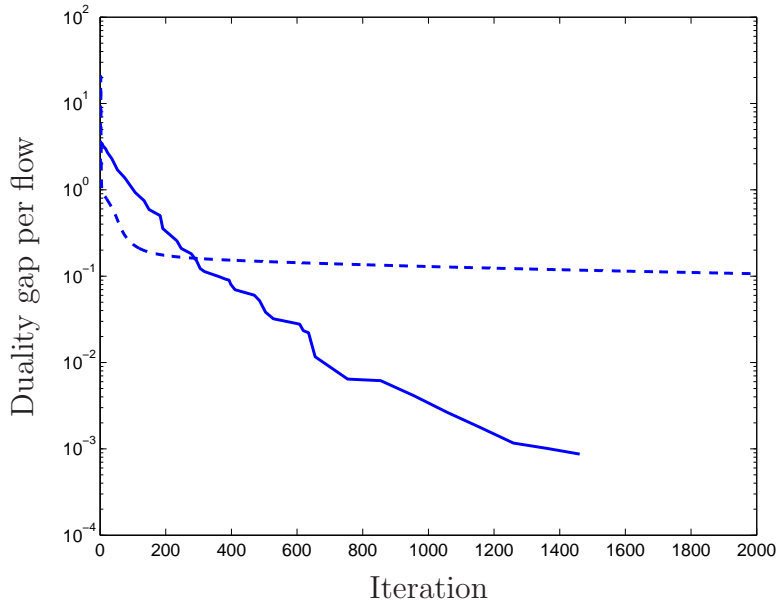
9

Figure 3: Convergence of truncated-Newton (solid) and dual decomposition (dashed) methods for a congested network.

4.4 Mixed linear and logarithmic utilities

In our last example, the flows have a mixture of linear and logarithmic utility functions. 40% of the flows have utility $U_j(f_j) = w_j f_j$, with w_j chosen i.i.d. uniformly distributed on $[10, 30]$, and the remaining 60% of the flows have utility $U_j(f_j) = \log f_j$. The routes and link capacities are generated exactly as in the example described in §4.2. Dual decomposition cannot be applied to this example, since the utility function is not strictly concave.

Figure 4 shows the convergence of the algorithm for this example. As we can see, this problem is a bit harder to solve than the previous two problems.

Figure 5 shows the distribution of the flow rates of the (approximate) solution. It is interesting to note that about half the flows with linear utility have been extinguished (up to numerical accuracy). This suggests that network utility maximization with linear utility functions could be used as an admission control algorithm.
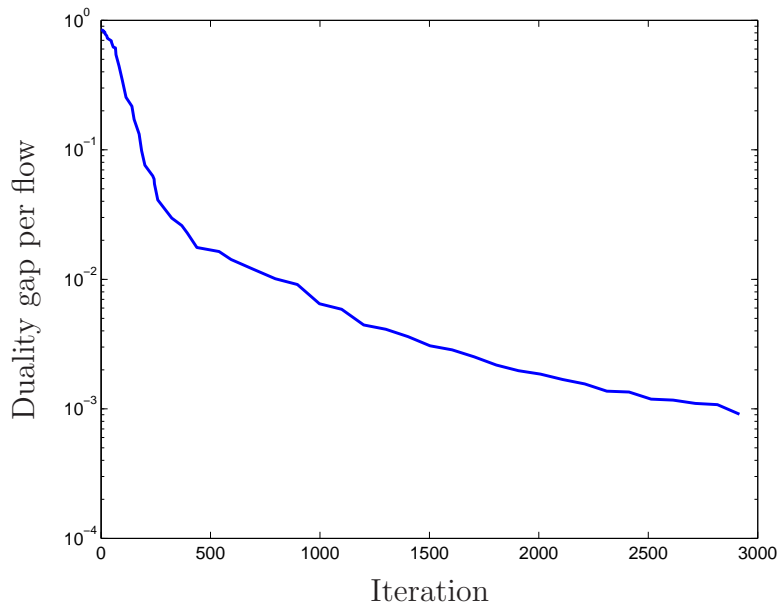
Figure 4: Convergence of truncated-Newton method for a network with linear and logarithmic utilities.
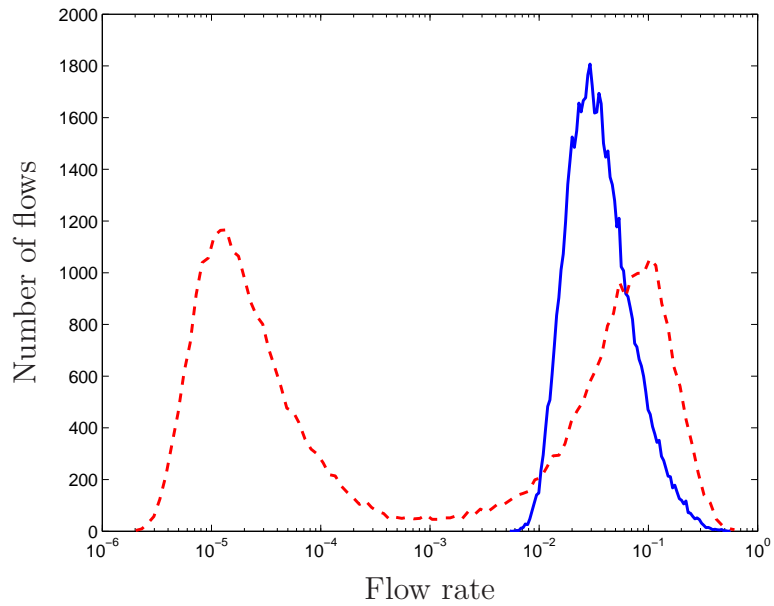

Figure 5: Histogram of flow rates with logarithmic (solid) and linear (dashed) utilities.

5 Delay-Utility Trade-off

5.1 Problem formulation

In this section we describe how to extend our method to handle network latency. We associate with link i the function $T_i : \mathbf{R} \to \mathbf{R}$ that gives the expected link delay as a function of the link capacity slack s_i. We assume that T_i is convex, nonincreasing and twice differentiable. We let $T(s) = (T_1(s_1), \ldots, T_m(s_m))$ denote the vector of link delays. The latency of flow j is the sum of the link delays on its route, and is given by $l_j = r_j^T T(s)$. The vector of latencies is given by $R^T T(s)$.

We modify the basic NUM problem to include a cost for flow latency as follows:

$$
\begin{array}{ll}
\text{maximize} & V(f) = \sum_j U_j(f_j) - \rho_j r_j^T T(s) \\
\text{subject to} & Rf \leq c, \quad f \geq 0,
\end{array}
\tag{5}
$$

where $\rho_j \geq 0$ is a parameter used to trade off utility versus latency for flow j. This is a convex optimization problem.

Our method can be applied directly to this problem, by substituting V for U. The gradient and Hessian of V are

$$
\begin{aligned}
\nabla V(f) &= \nabla U(f) + R^T \, \mathbf{diag}(b) T'(s), \\
\nabla^2 V(f) &= \nabla^2 U(f) - R^T \, \mathbf{diag}(b_i T_i''(s_i)) R,
\end{aligned}
$$

where $b = R\rho$. Therefore the system of equations that we need to solve at each primal-dual iteration for problem (5) has the same form as (4); we only have an additional term of the form $R^T D R$ (where D is a diagonal positive matrix).

5.2 Numerical results

We illustrate our method by generating the optimal trade-off curve between utility and delay, for the problem instance in our first example, described in §4.2. We use the link delay model $T_i(s_i) = c_i/s_i$, which is a common choice for the link delay function in the communications literature [Kle64, BG92, XJB04]. We set all ρ_i equal to the (scalar) ρ_0, which we vary to obtain the trade-off curve.

To generate the curve, we solve the problem (5) for 40 values of ρ_0, logarithmically spaced between 10^{-4} and 10^2. We compute the first point, corresponding to $\rho_0 = 10^{-4}$, using our method, which requires about 400 (PCG) iterations. Thereafter, we use a *warm-start* technique, by initializing the algorithm with the solution for the previous value of ρ_0. As a result, the number of iterations required to converge drops considerably, and is typically on the order of 10 for each new value of ρ_0. The entire trade-off curve is computed with a total of around 1100 iterations, less than times the effort required to solve the problem (5) from a 'cold start' (*i.e.*, with the simple generic initialization described above) for a single value of ρ_0. Figure 6 shows the optimal trade-off curve of average flow delay versus average flow utility.
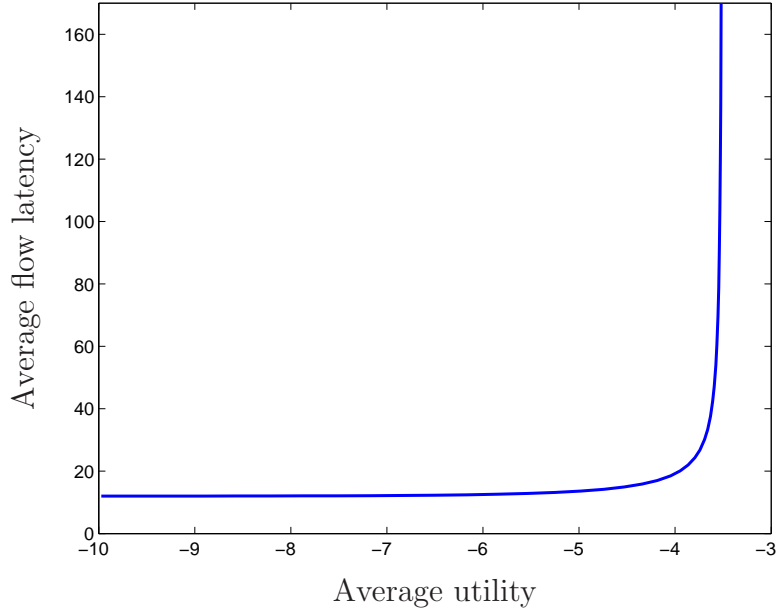
Figure 6: Optimal trade-off curve of average delay versus average utility.

6 Conclusions

We have described a primal-dual interior-point algorithm for solving network utility maximization problems. The algorithm can scale to large networks, with a million or more flows and links, and is very fast for smaller networks (say, with a few thousand flows). The algorithm has performance comparable to dual decomposition for networks without bottlenecks, long flows, or large variation in utility functions, and outperforms it in other cases. Another advantage is that it can handle utility functions that are not strictly concave, such as linear utility.

Our method works well in warm-start mode, which can be used to rapidly compute an entire trade-off curve between competing objectives, or to rapidly recompute the optimal flows when there is a modest change in the utilities, capacities, or other problem data.

The main disadvantage of our method is that it is not fully distributed, like dual decomposition, since each iteration requires the computation of a few inner products between flow specific data. In principle, inner products can be computed in a distributed way using a distributed averaging algorithm [BT89, MR06, BGPS06]. But we suspect that the resulting algorithm would be very slow.

Our method could be used to determine actual flow rates in networks (or subnetworks) in which all problem data is known centrally, or can be collected centrally, at modest cost.

13

Acknowledgments

References

[Ber98]   D. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, 1998.

[BG92]   D. Bertsekas and R. G. Gallager. *Data Networks*. Prentice-Hall, 1992.

[BGPS06] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, June 2006.

[BT89]   D. Bertsekas and J. Tsitsiklis. *Parallel and Distributed Computation*. Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[BV04]   S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[CLCD07] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, January 2007.

[Dem97]   J. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[DW60]   G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[Kel95]   C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, 1995.

[KKB07]   K. Koh, S. J. Kim, and S. Boyd. An interior point method for large-scale ℓ_1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, July 2007.

[Kle64]   L. Kleinrock. *Communication Nets: Stochastic Message Flow and Delay*. McGraw-Hill, 1964.

[KMT97]   F. Kelly, A. Maulloo, and D. Tan. Rate control for communication networks: Shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252, 1997.

[LL99]   S. H. Low and D. E. Lapsley. Optimization flow control I: Basic algorithms and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874, December 1999.

[Low03]   S. H. Low.   A duality model of TCP and queue management algorithms. *IEEE/ACM Transactions on Networking*, 11(4):525–536, August 2003.

[MR06]   C. Moallemi and B. Van Roy. Consensus propagation. *IEEE Transactions on Information Theory*, 52(11):4753–4766, December 2006.

[NW99]   J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

[PC06]   D. Palomar and M. Chiang. A tutorial on decomposition methods and distributed network resource allocation. *IEEE Journal of Selected Areas in Communication*, 24(8):1439–1451, August 2006.

[Sho85]   N. Z. Shor. *Minimization Methods for Non-Differentiable Functions*. Springer-Verlag, 1985.

[Sri04]   R. Srikant. *The Mathematics of Internet Congestion Control*. Birkhäuser, 2004.

[Wri97]   S. J. Wright. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics, 1997.

[XJB04]   L. Xiao, M. Johansson, and S. Boyd. Simultaneous routing and resource allocation via dual decomposition. *IEEE Transactions on Communications*, 52:1136–1144, 2004.