Approximation algorithms for utility-maximizing network design problem

Maciej Drwal

1 Introduction

Network utility maximization (NUM) problem is stated as a problem of maximizing concave objective function over a set of linear constraints [9]. The objective function aggregates the "utility" of a collection of flows, which is a nondecreasing function of the amount of flow transmitted between a source and sink nodes. Flows share common links along their paths, and total flow allocation in each link is limited by a fixed capacity.

The formulation of NUM problem is as follows. Let F be the set of flows (or commodities), $|F| = m$. For each flow $k \in F$ let $x_k \geq 0$ represent the transmission rate assigned to it. The network consists of L links, each of capacity $c_l > 0$. Each flow passes through a subset of links, which is expressed by a binary $m \times L$ matrix $\mathbf{A} = [A_{kl}]$, defined as: $a_{kl} = 1$ iff. kth flow passes through lth link. For each flow we have utility function $u_k : \mathbb{R} \to \mathbb{R}$, which assigns utility (pay-off) to rate x_k of a flow. The goal is to maximize the sum of all utilities:

$$\max_{\mathbf{x}} \{ \sum_{i \in F} u_i(x_i) : \ \mathbf{A}\mathbf{x} \leq \mathbf{c}, \mathbf{x} \geq 0 \}. \tag{1}$$

An instance of NUM consists of a set of flows F, set of functions u_i for each $i \in F$, matrix $\mathbf{A}$ and vector $\mathbf{c}$.

While NUM problem, formulated in this way, can be used to model performance of transmission control mechanisms in computer networks, there are also several related issues that can be captured within this framework. In this paper we focus on the *network design* problem from the perspective of utility maximization. The basic model (1) assumes fixed flow assignment, expressed in terms of matrix $\mathbf{A}$. This is reasonable, since in typical IP networks routing problem is separated from transmission rate control problem. One may ask however, whether it is possible to solve these problems jointly (in an efficient way)? Matrix $\mathbf{A}$ can be seen as a result of transformation of input set consisting of source-destination pairs into a flow-to-link assignment. We consider the problem of selecting a matrix that represents such paths for each flow, that the corresponding NUM problem has the greatest optimal solution over all possible sets of paths.

1.1 Problem formulation

We state a mixed-integer programming formulation of the considered problem as follows. Consider a complete graph G on n vertices, $V = \{1, \ldots, n\}$, with capacity function $c : V \times V \to \mathbb{R}_+ \cup \{0\}$. Let $I = \{(s_1, t_1), (s_2, t_2), \ldots, (s_m, t_m)\}$ be a subset of $V \times V$. Given is a set of functions $u_k : \mathbb{R} \to \mathbb{R}$, $k \in F = \{1, \ldots, m\}$.

Let $x_k \geq 0$ be a real variable denoting the rate of kth flow, let y_{ijk} be binary variable, assuming the value 1 if and only if kth flow uses edge (i, j) on its path from source s_k to destination t_k, and value 0 otherwise. We wish to maximize the following objective function:

$$U(\mathbf{x}, \mathbf{y}) = \sum_{k \in F} u_k(x_k) \tag{2}$$

subject to constraints:

$$\forall_{k \in F} \quad \forall_{i \in V} \quad \sum_{j=1}^{n} y_{ijk} - \sum_{j=1}^{n} y_{jik} = \begin{cases} 1, \text{ if } i = s_k, \\ -1, \text{ if } i = t_k, \\ 0, \text{ otherwise}, \end{cases} \tag{3}$$

$$\forall_{(i,j) \in V \times V} \quad \sum_{k \in F} x_k y_{ijk} \leq c(i,j), \tag{4}$$

$$\forall_{k \in F} \quad x_k \geq 0. \tag{5}$$

M. Drwal (✉)
Institute of Computer Science, Wrocław University of Technology, Wrocław, Poland
e-mail: maciej.drwal@pwr.wroc.pl

$$\forall_{i,j,k \in V\times V\times F} \quad y_{ijk} \in \{0,1\}, \tag{6}$$

We call this problem Utility-Maximizing Network Design problem (UMND). The problem can be seen as maximizing over all routing matrices **A** in (1), constrained to represent m simple paths between given set of source-destination node pairs I, in addition to selecting the best transmission rates **x**. The underlying connection structure can be, in general, an arbitrary graph, since capacity function c may assign values of zero to some edges.

2 Related works

Problems of maximizing the aggregate utility of flows in network are called *network utility maximization* (NUM) problems [2], [3], [6], [9]. NUM is related to the well-known *maximum multicommodity flow* problems [10]. In the latter we are given a set of k source-destination pairs (each representing a single commodity) with a demand D_k, and the goal is to find flow assignment function for each pair, such that link capacities are not exceeded and the flow conservation law is preserved on each node, and the common fraction of all routed commodity demands is maximized. In NUM demands are not fixed, but expressed using utility functions.

Combinatorial problems in which we are concerned with selecting subgraphs from a family of graphs are usually called *network design* problems. One commonly encountered problem in this area is called *buy-at-bulk network design* [1]. In this problem we are given a set of source-destination pairs in an undirected graph with given edge lengths, and we need to connect these pairs of nodes by purchasing the capacity c of any edge at some cost $f(c)$ per unit of length. The goal is to purchase sufficient amount of capacity for each edge so that it is possible to route total demand d_i between ith source-destination pair, while minimizing the cost paid. This problem is similar to UMND in the aspect that the cost is usually modeled using concave function f for purchasing capacity, which reflects the fact that increasing allocated resource for larger values yields reduction in cost.

In [7] a network design problem is formulated in which we look for set of shortest paths between all vertex pairs with total weights no greater than given threshold. It is established that such problem is NP-complete. In [12] a similar problem of joint routing and rate control to the one considered in this paper was formulated, and for general utility functions it was shown to be NP-complete. Here we consider only the class of iso-elastic utility functions, and show NP-completeness for such restricted class of problems. Moreover, we indicate that for such class of utility functions the randomized rounding [4], [11], technique may lead to a constant-factor approximation algorithm.

3 Main Results

It is easy to see that for an optimal solution we should assign to each flow the *widest simple path* between source and destination (this is such a path on which the minimal capacity of edges is maximized, and there are no repeated nodes). In case $m = 1$ this is easy to find just by computing maximum spanning tree in given graph, and restricting it to the path between s_1 and t_1. However, for $m > 1$ the paths resulting in highest bandwidth (and highest utility) for different flows may not be disjoint, which would violate capacity constraints.

For general utility functions u_i the problem appears to be difficult to solve. However, in practical applications, such as in computer networking, we are usually interested in utility functions which allow for *fair* allocations [8]. Henceforth, we restrict the choice of utility function to the family of *iso-elastic* functions:

$$u_i(x_i) = \begin{cases} w_i \dfrac{1}{1-\gamma} x_i^{1-\gamma} & \gamma > 0, \gamma \neq 1, \\ w_i \log x_i & \gamma = 1, \end{cases} \tag{7}$$

where $\gamma \in (0,\, 1]$ is a parameter. Such form of utility has the property that each flow $i \in F$ receives a share of capacity proportional to its weight w_i. If a link of capacity c is the bottleneck link for a collection of flows F, then maximal value of $\sum_{i \in F} u_i(x_i)$ for any $i \in F$ is:

$$x_i^* = c \frac{w_i^{1/\gamma}}{\sum_{k \in F} w_k^{1/\gamma}}.$$

If in addition to the network size n, also the number of flows m is given as a part of the input, then the utility-maximizing network design problem is very likely to be intractable.

Theorem 1. *Problem UMND is NP-complete.*

Proof. We reduce the BIN-PACKING problem [5] to UMND with iso-elastic utility functions with parameter $\gamma = 1$. In the decision version of BIN-PACKING problem we are given a set of items $U = \{a_1, a_2, \ldots, a_m\}$, each with integer size $w(a_k) = w_k > 0$, and two integers $B, K > 0$, and we ask whether it is possible to pack all the items into K bins of size B each.

Given an instance of BIN-PACKING, we construct an instance of UMND as follows. Each $a_i \in U$ corresponds to one source node s_i. Let $C = \sum_{i=1}^{m} w_i$. There is a central layer of K edges of capacity B, such that each of m source nodes is connected with each of these K edges via an edge of capacity w_i, (thus there are K outgoing edges from each source node s_i). Each of K edges from the central layer is connected to a single edge of capacity C. The other endpoint of this edge is the terminal node for all m flows. This is illustrated in Figure 1.

Each route from a source node to the terminal node must pass through exactly one of K edges of capacity B in the central layer and through the single terminal edge of capacity C. If the latter edge were a bottleneck, its capacity must have been completely filled, and since the utility functions are iso-elastic, each flow would be assigned a fraction w_i of its capacity. Consequently, the optimal solution of UMND problem would have the value:

$$v^* = \sum_{i=1}^{m} w_i \log w_i$$

We claim that such flow assignment is achieved if and only if the given instance of BIN-PACKING problem has a solution. To see this, suppose that all items U can be packed into K bins of size B. Then each flow of rate $x_i = w_i$ would pass through one of the central layer edges, and then through the terminal edge of capacity C, contributing to filling it completely. On the other hand, if not all items from U fit into K bins, the only way to route all flows through central layer would be to reduce the rate of at least one flow below the corresponding value w_i. But that would result in a solution of UMND strictly less than v^*.

Consequently, UMND is NP-hard. Moreover, its decision version obviously belongs to NP, since given a routing matrix and transmission rates, we can easily verify feasibility and evaluate the value of objective function in polynomial time, using formulation (2)–(6). Then it remains to compare the value of objective with a given threshold value. □

3.1 Linear-factor approximation algorithm

Consequently, we are interested in designing polynomial time approximation algorithms for UMND. Perhaps one of the simplest heuristics is based on finding the widest path for each flow separately, followed by squeezing the flows appropriately in order to fit into all links whose capacities were violated. This is summarized as Algorithm 1. As shown in Theorem 2 such algorithm never returns a solution that is less than a factor $O(\frac{1}{m})$ of optimal value.

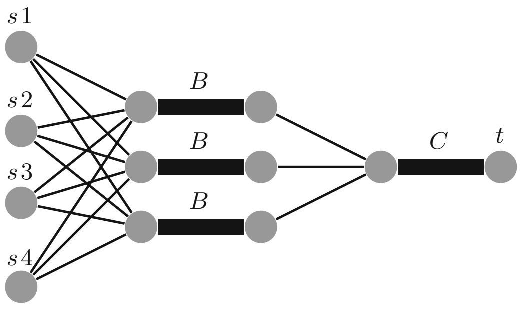

Fig. 1. Example network resulting from reducing instance of BIN-PACKING for $m = 4$ items and $K = 3$ bins of size B.

Theorem 2. *Algorithm 1 is $O(\frac{1}{m})$-approximate for UMND.*

Proof. Let $(x_i^*)_{i \in F}$ be rates in optimal solution of UMND, and let OPT be the value of optimal solution. Let $(x_i)_{i \in F}$ be rates in a solution produced by Algorithm 1, and let υ be the value of this solution. Let P_j denote the path computed by Algorithm 1 for realizing flow j (i.e., set of edges that make up this path).

For each flow $i \in F$ in solution given by Algorithm 1, let $e(i)$ denote the bottleneck edge of i. Observe, that if $e(i)$ is a bottleneck edge for some flow i in the solution produced by Algorithm 1, it must be that $x_i^* \le c(e(i))$, as the Algorithm 1 in its first phase finds the widest edge for flow i, unconstrained by the presence of other flows, so no higher allocation for i can be in optimal solution. But since $e(i)$ is a bottleneck edge, some subset of flows contribute to filling its capacity: $\sum_{j:e(i)\in P_j} x_j = c(e(i))$ (this sum always includes flow i). In the special case this sum may include all flows, if $e(i)$ happens to be the bottleneck for all flows. Thus we have:

$$\sum_{j\in F} x_j \ge \sum_{j:e(i)\in P_j} x_j = c(e(i)) \ge x_i^*,$$

which, after summing over all $i \in F$, gives:

$$\sum_{i\in F}\sum_{j\in F} x_j = m\sum_{j\in F} x_j \ge \sum_{i\in F} x_i^*. \tag{8}$$

Let $M = \max_{(i,j)\in V\times V} c(i,j)$. Since each $u_j(x_j)$ for $x_j \in [0, M]$, can never assume value greater than $w_j \frac{1}{1-\gamma} M^{1-\gamma}$, we can write:

$$\sum_{j\in F} u_j(x_j) \ge \alpha \sum_{j\in F} x_j,$$

where $\alpha = \min_{i\in F} \{w_i \frac{1}{1-\gamma} M^{-\gamma}\}$, and, consequently from (8):

$$\tilde{\upsilon} = \sum_{j\in F} u_j(x_j) \ge \frac{\alpha}{m} \sum_{j\in F} x_j^*.$$

Algorithm 1

Require: Graph $G = (V, c)$ with $|V| = n$ vertices and edge capacity function c. Set of s-d pairs $I \subset V \times V$ of size $|I| = m$.
Ensure: Set of paths $P_k \subset V \times V$ and rates of flows $x_k \geq 0$, for $k = 1, \ldots, m$.

```
1: Find maximum spanning tree in $G$. Denote the set of its edges by $T$.
2: for $k = 1, \ldots, m$ do
3:     Let $P_k \subset T$ be a path between $(s_k, t_k) \in I$ in the maximum spanning tree.
4:     Let $x'_k = \min_{(i,j) \in P_k} c(i, j)$.
5: end for
6: for $(i, j) \in V \times V$ do
7:     Let $S$ be the set of flows passing through edge $(i, j)$, i.e., $S = \{k : (i, j) \in P_k\}$.
8:     if $\sum_{k \in S} x'_k > c(i, j)$ then
9:         For all $k \in S$ let $x_k = c(i, j) \frac{x'_k}{\sum_{l \in S} x'_l}$.
10:    end if
11: end for
```

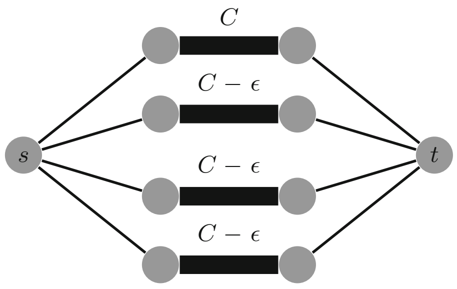

Fig. 2. Example network illustrating the worst case instance for Algorithm 1.

Since for all $j \in F$, $\alpha x_j^* + \alpha \geq u_j(x_j^*)$, we obtain:

$$\tilde{v} \geq \frac{\alpha}{m} \sum_{j \in F} u_j(x_j^*) - \alpha = \alpha \frac{1}{m} OPT - \alpha = O(\frac{1}{m}) OPT. \qquad \square$$

To see that the bound of $\frac{1}{m}$ is strict, consider the network of m parallel links, of which the first one has capacity C, and the remaining $m - 1$ of them have capacities $C - \varepsilon$ each, for some small $\varepsilon > 0$ (see Figure 2). There are m identically weighted flows originating from common node s and terminating at common node t (other edges have sufficiently high capacities to never be bottlenecks). Algorithm 1 would assign all flows to the first link, as it greedily looks for the widest paths for each of them separately. The value of solution would be $\sum_{i \in F} u_i(C/m) = m^{\gamma - 1} \sum_{i \in F} u_i(C)$. Optimal solution would consist of each flow assigned to a separate link, thus giving solution $\sum_{i \in F} u_i(C) - \varepsilon'$, where $\varepsilon' > 0$ can be arbitrarily small.

3.2 Constant-factor randomized approximation algorithm

It is possible to obtain a better algorithm by allowing randomization. The idea is to use the the mixed-integer program capturing UMND problem, and to relax integrality constraints. This allows to obtain an upper bound on the optimal solution in polynomial time, by solving the relaxation. Although we allow only for a single path for each flow, after the relaxation, the fractional solution might split a flow among several paths. However, such solution can be treated as a probability distribution on paths for a given flow. Subsequently, we can apply randomized rounding of such solution.

Since constraints (4) are nonlinear, we first need to reformulate it to an equivalent linear program. Let us introduce variables $\mathbf{z} = [z_{ijk}]$, and substitute (4) by the following set of constraints, to make sure that $z_{ijk} = x_r y_{ijk}$:

$$\forall_{i,j,k \in V \times V \times F} \quad z_{ijk} \leq C y_{ijk}, \tag{9}$$

$$\forall_{i,j,k \in V \times V \times F} \quad z_{ijk} \leq x_k, \tag{10}$$

$$\forall_{i,j,k \in V \times V \times F} \quad z_{ijk} \geq x_k - C(1 - y_{ijk}), \tag{11}$$

$$\forall_{i,j,k \in V \times V \times F} \quad z_{ijk} \geq 0. \tag{12}$$

Here $C = \max_{i,j} c(i, j)$. A path chosen for a flow is represented by a set of selected edges, that is a set of indices of variables, such that $y_{s_k u_1} = y_{u_1 u_2} = \ldots = y_{u_q t_k} = 1$. Observe, that constraints (3) guarantee that for each flow k there would be exactly one path selected.

Algorithm 2

```
Require: Instance of UMND problem in form of mixed-integer program (2)–(3), (5)–(12).
Ensure: Set of paths $\mathbf{y}$ and rates of flows $\mathbf{x}$.
 1: Solve the LP relaxation of (2)–(3), (5)–(12), denote fractional solution by $(\tilde{x}_k, \tilde{y}_{ijk})$.
 2: for $k = 1, \ldots, m$ do
 3:     $u \leftarrow s_k$
 4:     repeat
 5:         Select randomly $v \in V$, with probability of selecting $v$ being:
                $Pr[v \text{ is selected}] = \frac{\tilde{y}_{uvk}}{\sum_{w:\in V} \tilde{y}_{wuk}}.$
 6:         $y_{uvk} \leftarrow 1$ (and $y_{uwk} = 0$ for all other $w \in V$)
 7:         $u \leftarrow v$
 8:     until $v = t_k$
 9:     $x_k \leftarrow \frac{1}{e}\tilde{x}_k$
10: end for
11: Return solution $(\mathbf{x}, \mathbf{y}) = ([x_k], [y_{ijk}])$.
```

By relaxing constraints (6) to $y_{ijk} \geq 0$ we obtain a linear program. Let $(\tilde{x}_k, \tilde{y}_{ijk})$, for $i, j, k \in V \times V \times F$, denote an optimal fractional solution. Due to the constraint (3), for each flow k the sum of values of edge selection decision variables $\tilde{y}_{ijk}$ must be equal to 1. Since $\tilde{y}_{ijk}$ may now assume fractional values in the range [0, 1] that sum up to 1, we may use these values as a probability distribution of selecting a path among a subset of paths between s_k and t_k, that fractional solution would contain.

However, there is a concern that with nonzero probability the rounded solution may be infeasible, due to the violation of constraint (4). To overcome this, we may shrink the values of rates obtained from solving linear relaxation by multiplying them by a common factor. This would allow us to bound the probability of constraint violation. This idea leads to the Algorithm 2.

Proposition 1. *For any $\gamma \neq 1$, Algorithm 2 returns a feasible solution for UMND with nonzero probability. Such a solution is no worse than $e^{\gamma-1}$ times the optimal solution.*

Proof. The Algorithm 2 selects for each flow a path from the source s_k to destination t_k, by randomly forking on each edge, with probability given by fractional solution $\tilde{y}_{uvk}$. For each flow k, a particular vertex v is added to the currently constructed path with conditional probability:

$$Pr[\mathit{edge}(u, v) \text{ } \mathit{is\ selected} | \mathit{current\ path\ include\ vertex\ u}],$$

computed in step 5. In order to show that a solution constructed this way can be feasible, we calculate the probability of violating constraint (4).

Let $Y^k_{(i,j)} \in \{0, \tilde{x}_k\}$ be a random variable with $Pr[Y^k_{(i,j)} = \tilde{x}_k] = \tilde{y}_{ijk}$. Consider a random variable $Y = \sum_{k \in F} Y^k_{(i,j)}$. Its expected value is clearly $\mu = \sum_{k \in F} \tilde{x}_k \tilde{y}_{ijk} \leq c_{ij}$. We apply Chernoff bound to (4) including rounded solution $y_{ijk} \in \{0, 1\}$. For any $(i, j) \in V \times V$ and any $\delta > 0$ it holds that:

$$Pr[\sum_{k \in F} \tilde{x}_k y_{ijk} \geq (1+\delta)c_{ij}] = Pr[\sum_{k \in F} x_k y_{ijk} \geq c_{ij}] \leq \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^{c_{ij}},$$

where $x_r = \frac{1}{1+\delta}\tilde{x}_k$ is rate allocated by Algorithm 2.

Let us denote $C = \min_{(i,j) \in V \times V} c_{ij}$. Then:

$$Pr[\sum_{k \in F} x_k y_{ijk} \geq c_{ij}] < \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^C = \eta. \quad (13)$$

The rounded solution generated by Algorithm 2 is feasible if all constraints in set (4) are satisfied. From (13), any constraint concerning edge (i, j) is satisfied with probability at

Table 1 Performance evaluation of Algorithms 1 and 2.

n	d	m	A1 sol.	A1 time	A2 sol.	A2 time	OPT	time
20	50	10	128.63	1	137.71	1	144.39	2
20	100	10	122.33	1	167.8	3	175.68	13
50	100	10	151.13	1	153.26	3	153.26	3
50	100	25	334.69	1	379.49	4	384.88	5
70	400	10	121.35	1	162.71	3	174.00	40
70	250	12	142.30	1	178.61	5	179.37	1115
70	400	20	212.49	2	314.92	5	> 380.0	> 2h

least $(1 - \eta)$. Since there can be up to n^2 constraints, the probability that all of them are satisfied simultaneously cannot be less than $\varepsilon = (1-\eta)^{n^2}$. We show that $\varepsilon > 0$. By taking logarithm of (13) we obtain:

$$\log \frac{e^{\delta}}{(1+\delta)^{(1+\delta)}} = \log \eta^{1/C},$$

and for $\delta \geq e - 1$:

$$\log \eta^{1/C} = \delta - (1+\delta)\log(1+\delta) \leq \log(1+\delta)(\delta - 1 - \delta) = \log \frac{1}{1+\delta}.$$

Thus fixing $\delta = e - 1$ (see step 9 of Algorithm 2) we get $\log \eta^{1/C} = \log \frac{1}{e}$, and consequently $\varepsilon \geq (1 - e^{-C})^{n^2}$, which is greater than zero.

Finally, we evaluate the objective function, given solution $(x_k)_{k \in F}$ returned by Algorithm 2, and under the assumption that u_k are of the form (7). Since $\sum_{k \in F} u_k(\tilde{x}_k) \geq \sum_{k \in F} u_k(x_k^*) = OPT$, we have:

$$\sum_{k \in F} u_k(x_k) = \sum_{k \in F} u_k(\frac{1}{e}\tilde{x}_k) = e^{\gamma - 1} \sum_{k \in F} u_k(\tilde{x}_k) \geq e^{\gamma-1} OPT. \square$$

Table 1 contains results of computational experiments for several randomly generated input data sets, comparing performance of two presented approximation algorithms: values of solutions and corresponding running times (in secs.). Last two columns contain value of optimal solution (OPT) and running time of branch and cut algorithm from CPLEX solver; n is the number of nodes, d is the number of edges, and m is the number of flows.

4 Conclusions

The utility-maximizing network design problem with iso-elastic utility functions has been shown to be NP-complete. Approximate solutions can be obtained in polynomial time deterministically, but presented study suggests that randomization may yield better results. As Algorithm 2 may fail to give a feasible solution, further investigations are needed to determine the required number of runs.

References

1. B. Awerbuch and Y. Azar. Buy-at-bulk network design. In: *38th Annual Symposium on Foundations of Computer Science, 1997*, pages 542–547. IEEE, 1997.
2. M. Chiang, et al. Layering as optimization decomposition: A mathematical theory of network architectures. *Proc. of the IEEE*, 95 (1):255–312, 2007.
3. M. Drwal and D. Gasior. Utility-based rate control and capacity allocation in virtual networks. In *Proc. of the 1st European Teletraffic Seminar*, 2011.
4. M. Drwal and J. Jozefczyk. Decomposition algorithms for data placement problem based on Lagrangian relaxation and randomized rounding. *Annals of Operations Research, DOI: 10.1007/s10479-013-1330-7*, 2013.
5. M. Garey and D. Johnson. *Computers and intractability*, Freeman New York, 1979.
6. D. Gasior and M. Drwal. Pareto-optimal Nash equilibrium in capacity allocation game for self-managed networks. *Computer Networks*, 57(14):2675–2868, 2013.
7. D. Johnson, J.K. Lenstra, and A. Kan. The complexity of the network design problem. *Networks*, 8(4):279–285, 1978.
8. F. Kelly. Fairness and stability of end-to-end congestion control. *European Journal of Control*, 9(2-3):159–176, 2003.
9. 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(3):237–252, 1998.
10. T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46(6), 1999.
11. P. Raghavan and C. Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.
12. J. Wang, L. Li, S. Low, and J. Doyle. Cross-layer optimization in TCP/IP networks. *IEEE/ACM Transactions on Networking*, 13(3): 582–595, 2005.