A Multi Path Routing Algorithm for IP Networks Based on Flow Optimisation*

Henrik Abrahamsson, Bengt Ahlgren, Juan Alonso, Anders Andersson, and Per Kreuger

SICS – Swedish Institute of Computer Science
E-mail: first.lastname@sics.se

Abstract. Intra-domain routing in the Internet normally uses a single shortest path to forward packets towards a specific destination with no knowledge of traffic demand. We present an intra-domain routing algorithm based on multi-commodity flow optimisation which enable load sensitive forwarding over multiple paths. It is neither constrained by weight-tuning of legacy routing protocols, such as OSPF, nor requires a totally new forwarding mechanism, such as MPLS. These characteristics are accomplished by aggregating the traffic flows destined for the same egress into one commodity in the optimisation and using a hash based forwarding mechanism. The aggregation also results in a reduction of computational complexity which makes the algorithm feasible for on-line load balancing. Another contribution is the optimisation objective function which allows precise tuning of the tradeoff between load balancing and total network efficiency.

1 Introduction

As IP networks are becoming larger and more complex, the operators of these networks gain more and more interest in *traffic engineering* [3]. Traffic engineering encompasses performance evaluation and performance optimisation of operational IP networks. An important goal with traffic engineering is to use the available network resources more efficiently for different types of load patterns in order to provide a better and more reliable service to customers.

Current routing protocols in the Internet calculate the shortest path to a destination in some metric without knowing anything about the traffic demand or link load. Manual configuration by the network operator is therefore necessary to balance load between available alternate paths to avoid congestion. One way of simplifying the task of the operator and improve use of the available network resources is to make the routing protocol sensitive to traffic demand. Routing then becomes a flow optimisation problem.

One approach taken by others [8, 9, 12] is to let the flow optimisation result in a set of link weights that can be used by legacy routing protocols, e.g., open shortest path first (OSPF), possibly with equal cost multi-path (ECMP)

* Supported in part by Telia Research AB.

forwarding. The advantage is that no changes are needed in the basic routing protocol or the forwarding mechanism. The disadvantage is that the optimisation is constrained by what can be achieved with tuning the weights. Another approach is to use MPLS [4], multi-protocol label switching, for forwarding traffic for large and long-lived flows. The advantage is that the optimisation is not constrained, but at the cost of more complexity in the routing and forwarding mechanisms.

Our goal is to design an optimising intra-domain routing protocol which is *not* constrained by weight-tuning, and which *can* be implemented with minor modifications of the legacy forwarding mechanism based on destination address prefix.

In this paper we present a routing algorithm for such a protocol based on multi-commodity flow optimisation which is both computationally tractable for on-line optimisation and also can be implemented with a near-legacy forwarding mechanism. The forwarding mechanism needs a modification similar to what is needed to handle the ECMP extension to OSPF.

The key to achieve this goal, and the main contribution of this paper, is in the modelling of the optimisation problem. We aggregate all traffic destined for a certain egress into one commodity in a multi-commodity flow optimisation. This reduces the number of commodities to at most N, the number of nodes, instead of being N^2 when the problem is modelled with one commodity for each pair of ingress and egress nodes. As an example, the computation time for a 200 node network was in one experiment 35 seconds. It is this definition of a commodity that *both* makes the computation tractable, *and* the forwarding simple.

Another important contribution is the definition of an optimisation objective function which allows the network operator to choose a maximum desired link utilisation level. The optimisation will then find the most efficient solution, if it exists, satisfying the link level constraint. Our objective function thus enables the operator to control the trade-off between minimising the network utilisation and balancing load over multiple paths.

The rest of the paper is organised as follows. In the next section we describe the overall architecture where our optimising routing algorithm fits in. Section 3 presents the mathematical modelling of the optimisation problem. We continue with a short description of the forwarding mechanism in Sect. 4. After related work in Sect. 5 we conclude the paper.

2 Architecture

In this work we take the radical approach to completely replace the traditional intra-domain routing protocol with a protocol that is based on flow optimisation. This approach is perhaps not realistic when it comes to deployment in real networks in the near future, but it does have two advantages. First, it allows us to take full advantage of flow optimisation without being limited by current practise. Second, it results in a simpler overall solution compared to, e.g., the metric tuning approaches [8, 9, 12]. The purpose of taking this approach is to assess its feasibility and, hopefully, give an indication on how to balance flow optimisation functionality against compatibility with legacy routing protocols.

In this section we outline how the multi-commodity flow algorithm fits into a complete routing architecture. Figure 1 schematically illustrates its components. Flow measurements at all ingress nodes and the collection of the result are new components compared to legacy routing. The measurements continuously (at regular intervals) provide an estimate of the current demand matrix to the centralised flow optimisation. The demand matrix is aggregated at the level of all traffic from an ingress node destined for a certain egress node.

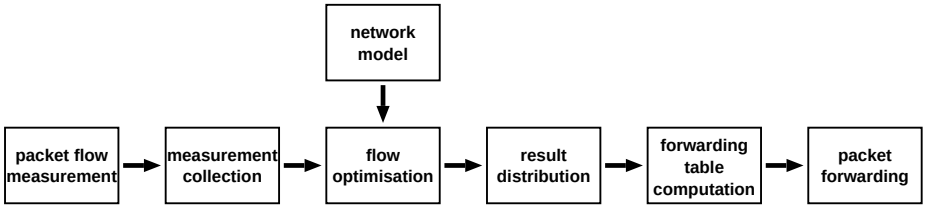

Fig. 1. Routing architecture with flow optimisation.

If a more fine-grained control over the traffic flows are desired, for instance to provide differentiated quality of service, a more fine-grained aggregation level can be chosen. This results in more commodities in the optimisation, which can be potential performance problem. One approach is to introduce two levels in the optimisation, one with a longer time-scale for quality of service flows.

The demand matrix is input to the flow optimiser together with a model of the network. The result of the optimisation is a set of values y_{ij}^t, which encode how traffic arriving at a certain node (i), destined for a certain egress node (t) should be divided between the set of next hops (j). These values are used at each node together with a mapping between destination addresses and egress nodes to construct forwarding tables. Finally, the packet forwarding mechanism is modified to be able to distinguish packets destined for a certain egress node, and to forward along multiple paths toward those egresses.

The computation of the multi-commodity flow optimisation algorithm is inherently centralised. In this paper we also think of the computation as implemented in a central server. If a so-called bandwidth broker is needed or desired for providing a guaranteed quality of service, it is natural to co-locate it with optimisation. We however see the design of a distributed mechanism implementing flow optimisation as an important future work item.

The timescale of operation is important in an optimising routing architecture. There are several performance issues that put lower bounds on the cycle flow measurement–optimisation–new forwarding tables. The flow measurement need to be averaged over a long enough time to get sufficiently stable values. Our current research as well as others [5] indicate that the needed stability exists in real networks at the timescale of a few, maybe five to ten, minutes. Other performance issues are the collection of the flow measurements, the computation of the optimisation algorithm, and the distribution of the optimisation result.

Our initial experiments indicate that a new optimisation cycle can be started in approximately each five minutes for typical intra-domain sizes.

An issue that we have identified is how to handle multiple egresses for a destination injected into the domain by BGP, the border gateway protocol. A straightforward way to solve this is to introduce additional virtual nodes in the network to represent a common destination behind both egresses. This approach may however introduce a large number of additional nodes. This will need to be more carefully considered in the future.

3 Optimisation

The routing problem in a network consists in finding a path or multiple paths that send traffic through the network without exceeding the capacity of the links. When using optimisation to find such (multiple) paths, it is natural to model the traffic problem as a (linear) multi-commodity network flow problem (see, e.g., Ahuja et al. [1]), as many authors have done.

First, the network is modelled as a directed graph (this gives the topology, i.e., the static information of the traffic problem), and then the actual traffic situation (i.e., the dynamic part of the problem, consisting of the current traffic demand and link capacity) as a linear program. In modelling the network as a graph, a node is associated to each router and a directed edge to each directional link physically connecting the routers. Thus, we assume a given graph $G = (N, E)$, where N is a set of nodes and E is the set of (directed) edges. We will abuse language and make no distinction between graph and network, node and router, or edge and link.

Every edge $(i, j) \in E$ has an associated capacity k_{ij} reflecting the bandwidth available to the corresponding link. In addition, we assume a given *demand matrix* $D = D(s, t)$ expressing the traffic demand from node s to node t in the network. This information defines the routing problem. In order to formulate it as a multi-commodity flow (MCF) problem we must decide how to model commodities. In the usual approach [1, 8, 11] commodities are modelled as source-destination pairs that are interpreted as "all traffic from source to destination". Thus, the set of commodities is a subset of the Cartesian product $N \times N$; consequently, the number of commodities is bounded by the square of the number of nodes. To reduce the size of the problem and speed-up computations, we model instead commodities as (only destination) nodes, i.e., a commodity t is to be interpreted as "all traffic to t". Thus, our set of commodities is a subset of N and, hence, there are at most as many commodities as nodes. The corresponding MCF problem can be formulated as follows:

$$\min \{f(y) \mid y \in P_{12}\} \qquad (MCF_{12})$$

where $y = (y_{ij}^t)$, for $t \in N, (i, j) \in E$, and P_{12} is the polyhedron defined by the equations:

$$\sum_{\{j|(i,j)\in E\}} y_{ij}^t - \sum_{\{j|(j,i)\in E\}} y_{ji}^t = d(i,t) \quad \forall i, t \in N \qquad (1)$$

$$\sum_{t \in N} y_{ij}^t \;\leq\; k_{ij} \qquad \forall (i,j) \in E \tag{2}$$

where

$$d(i,t) = \begin{cases} -\sum_{s \in N} D(s,t) & \text{if } i = t \\ D(i,t) & \text{if } i \neq t \end{cases}.$$

The variables y_{ij}^t denote the amount of traffic to t routed through the link (i,j). The equation set (1) state the condition that, at intermediate nodes i (i.e., at nodes different from t), the outgoing traffic equals the incoming traffic plus traffic created at i and destined to t, while at t the incoming traffic equals all traffic destined to t. The equation set (2) state the condition that the total traffic routed over a link cannot exceed the link's capacity.

It will also be of interest to consider the corresponding problem *without* requiring the presence of the equation set (2). We denote this problem (MCF_1). Notice that every point $y = (y_{ij}^t)$ in P_{12} or P_1 represents a possible solution to the routing problem: it gives a way to route traffic over the network so that the demand is met and capacity limits are respected (when it belongs to P_{12}), or the demand is met but capacity limits are not necessarily respected (when it belongs to P_1). Observe that $y = (0)$ is in P_{12} or in P_1 only in the trivial case when the demand matrix is zero.

A general linear objective function for either problem has the form $f(y) = \sum_{t,(i,j)} b_{ij}^t \; y_{ij}^t$. We will, however, consider only the case when all $b_{ij}^t = 1$ which corresponds to the case where all commodities have the same cost on all links. We will later use different objective functions (including non-linear ones) in order to find solutions with desired properties.

3.1 Desirable Solutions

In short, the solutions we consider to be desirable are those which are *efficient* and *balanced*. We make these notions precise as follows.

We use the objective function considered above, $f(y) = \sum_{t,(i,j)} y_{ij}^t$, as a measure of efficiency. Thus, given y_1, y_2 in P_{12} or P_1, we say that y_1 is *more efficient* than y_2 if $f(y_1) \leq f(y_2)$. To motivate this definition, note that whenever traffic between two nodes can be routed over two different paths of unequal length, f will choose the shortest one. In case the capacity of the shortest path is not sufficient to send the requested traffic, f will utilise the shortest path to 100% of its capacity and send the remaining traffic over the longer path.

Given a point $y = (y_{ij}^t)$ as above, we let $Y_{i,j} = \sum_{t \in N} y_{ij}^t$ denote the total traffic sent through (i,j) by y. Every such y defines a *utilisation* of edges by the formula $u(y,i,j) = Y_{ij}/k_{ij}$, and $u(y,i,j) = 0$ when $k_{ij} = 0$. Let $u(y)$ denote the maximum value of $u(y,i,j)$ where (i,j) runs over all edges. Given an $\ell > 0$, we say that $y \in P_{12}$ (or $y \in P_1$) is *ℓ-balanced* if $u(y) \leq \ell$. For instance, a solution is (0.7)-balanced if it never uses any link to more than 70 % of its capacity.

3.2 How to Obtain Desirable Solutions

Poppe et al. [11] have proposed using different linear objective functions in order to obtain traffic solutions that are desirable with respect to several criteria (including balance, in the form of minimising the maximum utilisation of edges). Fortz and Thorup [8, 9], on the other hand, considers a fixed piece-wise linear objective function (consisting of six linear portions for each edge) which makes the cost of sending traffic along an edge depend on the utilisation of the edge. By making the cost increase drastically as the utilisation approaches 100 %, the function favours balanced solutions over congested ones. As the authors express it, their objective function "provides a general best effort measure".

Our contribution is related to the above mentioned work in that we use different objective functions to obtain desirable solutions, and the functions are piece-wise linear and depend on the utilisation. In contrast, our work defines different levels of balance (namely, ℓ-balance). For each such level, a simple piece-wise linear objective function consisting of two linear portions for each edge is *guaranteed* to find ℓ-balanced solutions provided, of course, that such solutions exist. Moreover, the solution found is guaranteed to be more efficient than any other ℓ-balanced solution.

Another distinctive feature of our functions is that they are defined through a uniform, theoretical "recipe" which is valid for every network. We thus eliminate the need to use experiments to adapt our definitions and results to each particular network. Finally, the fact that our functions consist of only two linear portions, shorten the execution time of the optimisation.

3.3 The Result

To formulate our result we need to introduce some notation. Let $y = (y_{ij}^t)$ be a point of P_{12} or P_1, and suppose given real numbers $\lambda > 1$ and $\ell > 0$. We define the link cost function (illustrated in Fig. 2)

$$C^{\ell,\lambda}(U) = \begin{cases} U & \text{if } U \leq \ell \\ \lambda\, U + (1-\lambda)\, \ell & \text{if } U \geq \ell \end{cases} .$$

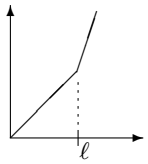

Fig. 2. The link cost function $C^{\ell,\lambda}$.

We use this function in the definition of the following objective function:

$$f^{\ell,\lambda}(y) = \sum_{(i,j)\in E} k_{ij}\; C^{\ell,\lambda}(u(y,i,j))$$

We also need to define the following constants:

$$v = \min \{f(y) \,|\, y \in P_{12}\} \qquad \text{and} \qquad V = \max \{f(y) \,|\, y \in P_{12}\}$$

Notice that $v > 0$ since $D(s,t) > 0$, and $V < \infty$ since the network is finite and we are enforcing the (finite) capacity conditions. At a more practical level, v can be computed by simply feeding the linear problem $\min \{f(y) \,|\, y \in P_{12}\}$ into CPLEX and solving it. Then, to compute V, one changes the same linear problem to a max problem (by replacing "min" by "max") and solves it.

Finally, let $\delta > 0$ denote the minimum capacity of the edges of positive capacity. We can now state the following theorem whose proof is given in a technical report [2]:

Theorem 1. *Let ℓ, ϵ be real numbers satisfying $0 < \ell < 1$ and $0 < \epsilon < 1 - \ell$. Suppose that $y \in P_1$ is ℓ-balanced, and let $\lambda > 1 + \frac{V^2}{v\delta\epsilon}$. Then any solution x of MCF_1 with objective function $f^{\ell,\lambda}$ is $(\ell + \epsilon)$-balanced. Moreover, x is more efficient than any other $(\ell + \epsilon)$-balanced point of P_1.*

Observe that, since $\ell < 1$ and $y \in P_1$ is ℓ-balanced, we can use MCF_1 instead of MCF_{12}. Informally, the theorem says that if there are ℓ-balanced solutions, then $f^{\ell,\lambda}$ will find one. The number $\epsilon > 0$ is a technicality needed in the proof. Notice that it can be chosen arbitrarily small.

Theorem 1 can be used as follows. Given a target utilisation ℓ, say $\ell = 0.7$, compute $\frac{V^2}{v\delta\epsilon}$, choose a λ as in Theorem 1, and choose $\epsilon > 0$, say $\epsilon = 0.01$. Finally, compute a solution, say x, of MCF_1 with objective function $f^{\ell,\lambda}$. Then there are two exclusive possibilities: either x is 0.71-balanced or there is no such solution. In the last case, x can be thought of as a "best effort" solution since we have penalised all utilisation above 0.7 (which forces traffic using edges to more than 70 % of capacity to try to balance) but no 0.71-balanced solution exists. At this point we can either accept this best effort solution or iterate, this time setting the balance target to, say, 0.85, etc. After a few iterations we arrive at a solution which is "sufficiently" balanced or we know that there is no solution that is ℓ-balanced for the current value of ℓ which, we may decide, is so close to 1 that it is not worthwhile to continue iterating.

3.4 A Generalisation

Theorem 1 has a useful generalisation that can be described as follows. Partition the set of edges E into a family (E_i) of subsets, and choose a target utilisation ℓ_i for each E_i. The generalised theorem says that for small $\epsilon > 0$ we can define a function corresponding to $f^{\ell,\lambda}$ in Theorem 1, such that solving MCF_1 with this objective function will result in efficient solutions that are $(\ell_i + \epsilon)$-balanced on E_i provided, of course, that such solutions exist. The generalised theorem is more flexible in that it allows us to seek solutions with different utilisation in different parts of the network.

3.5 Quantitative Results

We have used CPLEX 7.1[1] on a Pentium laptop to conduct numerical experiments with a graph representing a simplified version of a real projected network. The graph has approximately 200 nodes and 720 directed edges. If we had modelled MCF with source-destination pairs as commodities, the linear problem corresponding to MCF_{12} would consist of some 8 million equations and 30 million variables. Modelling commodities as traffic to a node, MCF_{12} contains, in contrast, "only" about 40 000 constraints and 140 000 variables. Solving MCF_1 with objective function $f^{\ell,\lambda}$ takes approximately 35 seconds.

Solving the same problem with the objective function considered by Fortz and Thorup [8, 9] takes approximately 65 seconds. Our experiments suggest that this function picks solutions that minimise balance. In contrast, with $f^{\ell,\lambda}$ we can choose any desired level of balance (above the minimum, of course).

4 Multi-Path Forwarding

By modelling the routing problem as "all traffic to t", as described in the previous section, we get an output from the optimisation that is well suited for packet forwarding in the routers. The result from the optimisation, the y_{ij}^t values, tells how packets at a certain node (i) to a certain egress node (t) in the network should be divided between the set of next hops (j). We thus need a forwarding mechanism that can distinguish packets destined for a certain egress, and that can forward along multiple paths.

To enable forwarding along multiple paths, we introduce one more step in the usual forwarding process. An egress data structure is inserted in the address lookup tree just above the next hop data structure as illustrated in Fig. 3. A longest prefix match is done in the same manner as in a standard forwarding table, except that it results in the destination egress node. The egress data structure stores references to the set of next hops to which traffic for that egress should be forwarded, as well as the desired ratios (the y_{ij}^t for all js) between the next hops.

In order to populate the forwarding tables a mapping has to be created between destination addresses and egress nodes. The needed information is the same as a regular intra-domain routing protocol needs, and is obtained in much the same way. For destinations in networks run by other operators (i.e., in other routing domains), the mapping is obtained from the BGP routing protocol. For intra-domain destinations, the destination prefix is directly connected to the egress node.

Mechanisms for distributing traffic between multiple links have been thoroughly evaluated by Cao et al. [6]. We propose to use a table based hashing mechanism with adaptation, because it can distribute the load according to unequal ratios, is simple to compute, and adapts to the properties of the actual traffic.

Similar mechanisms already exist in commercial routers in order to handle the equal cost multi-path extension to OSPF and similar protocols.

[1] ILOG CPLEX 7.1 http://www.ilog.com

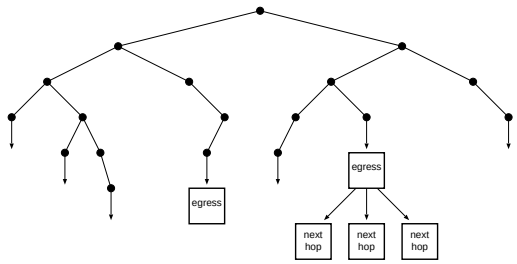

Fig. 3. Address lookup data structure for multiple path forwarding.

5 Related Work

With the prospect of better utilising available network resources and optimising traffic performance, a lot of research activity is currently going on in the area of traffic engineering. The general principles and requirements for traffic engineering are described in the RFC 3272 [3] produced by the IETF Internet Traffic Engineering working group. The requirements for traffic engineering over MPLS are described in RFC 2702 [4].

Several researchers use multi-commodity flow models in the context of traffic engineering. Fortz and Thorup [8, 9] use a local search heuristics for optimising the weight setting in OSPF. They use the result of multi-commodity flow optimisation as a benchmark to see how close to optimal the OSPF routing can get using different sets of weights. Mitra and Ramakrishnan [10] describes techniques for optimisation subject to QoS constraints in MPLS-supported IP networks. Poppe et al. [11] investigate models with different objectives for calculating explicit routes for MPLS traffic trunks. Multi-commodity flow and network flow models in general have numerous application areas. A comprehensive introduction to network flows can be found in Ahuja et al. [1].

A somewhat controversial assumption when using multi-commodity flow optimisation is that an estimate of the demand matrix is available. The problem of deriving the demand matrix for operational IP networks is considered by Feldmann et al. [7]. The demand matrix only describes the current traffic situation but, for an optimisation to work well, it must also be a good prediction of the near future. Current research in traffic analysis by Bhattacharyya et al. [5] and Feldmann et al. [7] indicate that sufficient long term flow stability exists on backbone links in timescales of minutes and hours and in manageable aggregation levels to make optimisation feasible.

6 Conclusions

We have taken the first steps to introduce flow optimisation as a routing mechanism for an intra-domain routing protocol. We have presented a routing algorithm based on multi-commodity flow optimisation which we claim is computationally tractable for on-line routing decisions and also only require a small

modification to the legacy packet forwarding mechanism. More work is however needed on other components in order to design and implement a complete routing protocol using our algorithm.

The key issue, and our main contribution, is the mathematical modelling of commodities. Traffic destined for a certain egress node is aggregated into a single commodity. This results in computational requirements an order of magnitude smaller than in the traditional models where the problem is modelled with one commodity for each flow from one ingress to one egress node.

Multi-path forwarding of the aggregates produced by the optimiser is then handled by a hash based forwarding mechanism very similar to what is needed for OSPF with ECMP.

Another contribution is the design of a generic objective function for the optimisation which allows the network operator to choose a desired limit on link utilisation. The optimisation mechanism then computes a most efficient solution given this requirement, when possible, and produces a best effort solution in other cases. The process can be iterated with, e.g., binary search to find a feasible level of load balance for a given network load.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows*. Prentice-Hall, 1993.

[2] J. Alonso, H. Abrahamsson, B. Ahlgren, A. Andersson, and P. Kreuger. Objective functions for balance in traffic engineering. Technical Report T2002:05, SICS – Swedish Institute of Computer Science, May 2002.

[3] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of Internet traffic engineering. Internet RFC 3272, May 2002.

[4] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for traffic engineering over MPLS. Internet RFC 2702, September 1999.

[5] S. Bhattacharyya, C. Diot, J. Jetcheva, and N. Taft. Pop-level and access-link-level traffic dynamics in a tier-1 pop. In *ACM SIGCOMM Internet Measurement Workshop*, San Francisco, USA, November 2001.

[6] Z. Cao, Z. Wang, and E. Zegura. Performance of hashing-based schemes for internet load balancing. In *Proc. of IEEE INFOCOM 2000*, Israel, March 2000.

[7] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. In *Proceedings of ACM SIGCOMM'00*, Stockholm, Sweden, August 2000.

[8] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings IEEE INFOCOM 2000*, pages 519–528, Israel, March 2000.

[9] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, May 2002.

[10] D. Mitra and K. G. Ramakrishnan. A case study of multiservice, multipriority traffic engineering design for data networks. In *Proc. of Globecom'99*, Brazil, 1999.

[11] F. Poppe, S. van den Bosch, P. de la Vallée-Poussin, H. van Hove, H. de Neve, and G. Petit. Choosing the objectives for traffic engineering in IP backbone networks based on Quality-of-Service requirements. In *Proceedings of First COST 263 International Workshop, QofIS*, pages 129–140, Berlin, Germany, Sept. 2000.

[12] D. Yuan. *Optimization Models and Methods for Communication Network Design and Routing*. PhD thesis, Linköpings Universitet, 2001.