# Metric Optimization in IP Networks

Cédric Fortuny Olivier Brun Jean-Marie Garcia

February 25, 2005

**Abstract**

In IP networks, traffic flows are routed along shortest paths according to link weights set by the network operator. In this paper, we propose a local search based algorithm for weight optimization. This algorithm can be used either in single-start mode, for incremental weight optimization, as well as in multi-start mode, for global weight optimization. Extensive computational results show that this algorithm yields IP routings which are close to the optimal load-sharing routings within low computing times with respect to known weight optimization algorithm.

## 1 Introduction

With more than 800 millions worldwide users, Internet has become pervasive in the last decade thanks to a succession of drivers, starting with the rise of the World Wide Web, followed by business adoption of Internet applications, and more recently, the use of peer-to-peer file-sharing software. Internet traffic keeps growing at a very high rate (66% in 2003), due to the increasing number of users but also to the phenomenal development of peer-to-peer traffics. More users are staying on the Internet longer and the same users are using higher-access bandwidths as well. Beside, the Internet has also evolved into a global communication infrastructure supporting real-time services (e.g. Voice Over IP) with stringent Quality-of-Service (QoS) requirements [2, 3].

Network operators have to regularly upgrade their network infrastructure in order to face the growth of Internet traffic and the rising QoS requirements. In the current competitive context, a cost-efficient alternative to installing excessive amounts of capacity is to perform route optimization in order to avoid network congestion and service degradation. Route optimisation allows to make more efficient use of existing network resources by tailoring routes to prevailing traffic.

Open Shortest Path First (OSPF) and Intermediate System to Intermediate System (IS-IS) are the most commonly used intra-domain Internet routing protocols [19, 20]. Traffic flow is routed along shortest paths, splitting flow at nodes where several outgoing links are on shortest paths to the destination. The weights of the links, and thereby the shortest path routes, can be changed by the network operator. The weights could

be set to 1 or proportional to their physical distances, but the standard practice recommended by Cisco is to make the weight of a link inversely proportional to its capacity. Although fairly efficient in most cases, such heuristics often lead to poor network resource utilization.

Given a set of traffic demands between origin-destination pairs, the routing weight optimization problem amounts to finding a set of link weights optimizing a given performance measure. This problem is known to be NP-hard [10]. Previous works can be classified into two broad approaches. Given an optimal flow allocation [7], the first approach amounts to solving an inverse shortest path problem. It yields a set of link weights such that shortest path routing gives the same link loads as the optimal flow allocation. Since it is not guaranteed that a feasible solution to the inverse shortest path problem exist, approximate methods have been developed using linear programming [4, 5, 6], quadratic programming [16], or greedy heuristics [17]. Given an initial solution, i.e. a set of link weights, the other approach aims at iteratively improving the solution. Fortz and Thorup [10] proposed a local search heuristic and tested it on a realistic AT&T backbone network and on synthetic networks. Other authors have proposed genetic algorithms for IP metric optimization [22, 23].

The weight optimization algorithm proposed in this paper is based on local search techniques and has several advantages with respect to the above mentioned works. The first one is its versatility since it can be used either in single-start mode, for incremental weight optimization, as well as in multi-start mode, for global weight optimization. The incremental weight optimization enables the network operator to improve its current routing with few weight changes. If more weight changes are allowed, global weight optimization can be used to avoid beeing trapped in a local minimum and thus get a better solution. Extensive computational results show that the proposed algorithm delivers solutions which are often very close to the lower bound given by the optimal load-sharing routing. These results show also that our algorithm is very fast with respect to the other weight optimization algorithms proposed in the litterature.

The next section is devoted to the mathematical statement of the OSPF weight setting problem. The proposed algorithm is described in section 3. Computational results are reported in section 4. Concluding remarks are drawn in the last section.

# 2 Problem Statement

Let us assume that we are given a network of $N$ nodes and $M$ links represented by a graph $G = [X, U]$, the link bandwidth capacities $C_u$ for all $u \in U$ and a set of $K$ origin-destination (OD) flows, where each flow $k = 1, \ldots, K$ is defined by its source node $s(k)$, its destination node $t(k)$ and its bandwidth requirement $\lambda_{s(k)}^{t(k)}$.

A feasible solution of the weight setting problem is a vector $W = (w_1, \ldots, w_M)$, where $w_a \in \Omega$ is the weight assigned to arc $a \in U$ and $\Omega = \{1, \ldots, w_{max}\}$ is the set of allowed integer values for link weights. The OSPF protocol allows for $w_{max} \leq 2^{16} - 1$.

Given a feasible solution $W = (w_1, \ldots, w_M)$, a shortest path algorithm can be used to compute the following parameters:

- $D_i^x(W)$ is the distance from node $i$ to destination node $x$,
- $\delta_{i,j}^x(W) = 1$ if link $(i,j)$ is on a shortest path towards destination node $x$, i.e. $D_i^x(W) = w_{i,j} + D_j^x$, and 0 otherwise,
- $n_i^x(W) = \sum_j \delta_{i,j}^x(w)$ is the number of outgoing arcs from node $i$ that are on a shortest path towards destination node $x$.

These parameters summarize all the information on shortest paths from node $i$ to node $x$ for the weight vector $W$. Assuming that traffic is split evenly between all outgoing links on the shortest paths towards the destination, the total traffic received at node $i$ for destination $x$ is,

$$\gamma_i^x(W) = \lambda_i^x \; + \; \sum_{j \neq x} \frac{\delta_{j,i}^x(W)}{n_j^x(W)} \, \gamma_j^x(W) \quad i, x \in X \tag{1}$$

Therefore, the traffic on link $(i,j)$ is given by:

$$Y_{i,j}(W) = \sum_{x=1}^{N} Y_{i,j}^x(W) \quad \text{where} \quad Y_{i,j}^x(W) = \frac{\delta_{i,j}^x(W)}{n_i^x(W)} \, \gamma_i^x(W) \tag{2}$$

$Y_{i,j}^x(W)$ is the traffic on link $i$ for destination $x$. Thus, if we are given a weight vector $W = (w_1, \ldots, w_M)$, the previous formula allow the recursive computation of the traffic $\gamma_i^x(W)$ at node $i$ for destination $x$ and the traffic $Y_{i,j}(w)$ on each link $(i,j)$ once the parameters $\delta_{i,j}^x(w)$ and $n_i^x(w)$ have been computed.

Let $\Phi(w)$ be the cost of the solution $W$. The problem can be stated as follows:

$$\min_{W \in \Omega^M} \Phi(W) \tag{3}$$

In the following, we will consider two different cost functions. The first one is an additive cost function,

$$\Phi(W) = \frac{1}{\lambda} \sum_{l=1}^{M} \Phi_l(W) \quad \text{with} \quad \lambda = \sum_k \lambda_{s(k)}^{t(k)}$$

, where $\Phi_l(W)$ is the average delay inccured by a packet in interface $l = 1 \ldots M$. Let $\rho_l = Y_l(W)/C_l$ be the utilization rate of interface $l$ and $K_l$ be the capacity of its buffer (in packets). The average delay $\Phi_l(W)$ can be computed using the famous M/M/1/K queueing formula,

$$\Phi_l(W) = \begin{cases} \dfrac{K_l \, \rho_l^{K_l+2} - (K_l+1) \, \rho_l^{K_l+1} + \rho_l}{Y_l(W) \, \left[1 - \rho_l - \rho_l^{K_l+1} + \rho_l^{K_l+2}\right]} & \text{if } \rho_l \neq 1 \\ \dfrac{K_l}{2 \, Y_l(W)} & \text{if } \rho_l = 1 \end{cases}$$

In this case, the cost $\Phi(w)$ represent the average delay of a packet in the network. However, our algorithm is not restricted to additive cost functions, and the second cost function we will consider is the maximum of the interface utilizations,

$$\Phi(W) = \max_{1 \leq l \leq M} \Phi_l(W) \quad \text{where} \quad \Phi_l(W) = \frac{Y_l(W)}{C_l} \; l = 1 \ldots M$$

# 3 A local search algorithm for weight optimization

To solve the weight optimization problem, we propose a local search heuristic [1, 11, 12]. The main originality of this algorithm lies in the neighbourhood structure it uses.

## 3.1 Neighbourhood definition

The neighbourhood of a solution $W = (w_1, \ldots, w_M)$ is defined as follows:

$$V(W) = \left\{W^1, W^2, \ldots, W^M\right\} \tag{4}$$

where $W^i = (w_1, w_2, \ldots, w_i + \Delta_i, \ldots, w_M)$, for $i = 1, \ldots, M$, with:

$$\Delta_i = argmin_{\Delta \geq 1}[Y_i\,(w_1, \ldots, w_i + \Delta, \ldots, w_M) < Y_i(w)] \tag{5}$$

Thus, the neighbourhood of a given solution $W$ contains at most $M$ solutions. Each one is associated to a link $i$ and is obtained by increasing the weight of this link by the minimum amount such that the load of this link is decreased, i.e. such that traffic is deviated from this link.

## 3.2 Neighbourhood generation

Let $F_i$ be the set of OD flows transmitted over link $i$. If $F_i = \emptyset$, then $\Delta_i$ can not be defined since it is impossible to deviate traffic from link $i$. Otherwise, we proceed as follows to compute $\Delta_i$:

**(a)** Set $\overline{W^i} = (w_1, \ldots, w_{i-1}, \infty, w_{i+1}, \ldots, w_M)$.

**(b)** For all $u, v \in X$, update distances $D_u^v(\overline{W^i})$ using a shortest path algorithm

**(c)** Let,

$$d^i_{\min} = \min_{f \in F} \left[D^{t(f)}_{s(f)}(\overline{W^i}) - D^{t(f)}_{s(f)}(W)\right] \tag{6}$$

be the minimum increase of shortest path length over all flow $f \in F_i$.

**(d)** $\Delta_i$ is given by,

$$\Delta_i = \begin{cases} 1 & \text{if } d^i_{\min} = 0 \\ d^i_{\min} & \text{if } 0 < d^i_{\min} < \infty \\ \infty & \text{if } d^i_{\min} = \infty \end{cases}$$

The solution $W^i = (w_1, w_2, \ldots, w_i + \Delta_i, \ldots, w_M)$ deviates traffic from link $i$:

- If $d^i_{\min} = 0$, there was multiple shortest paths for at least one OD flow of $F_i$ (load sharing). By increasing the weight $w_i$ by $\Delta_i = 1$, this flow will be completely deviated from link $i$,
- If $0 < d^i_{\min} < \infty$, the solution $W^i$ introduces load-sharing for at least one OD flow belonging to $F_i$,
- If $d^i_{\min} = \infty$, link $i$ is mandatory for all flows $f \in F_i$ and it is easy to prove that it can be suppressed from the set of links whose weight has to be optimized (provided the network is connected).

## 3.3 Weight optimization algorithm

The pseudo-code in figure 1 describes the proposed local search algorithm in detail.

Two key operations are often performed during neighbourhood exploration. The first one is the update of shortest paths when a single link weight is increased. This occurs when the weight of link $i$ is set to $\infty$ in order to compute $\Delta_i$, but also when this weight is set to $w_i + \Delta_i$ in order to evaluate the neighbour $W^i$.

Considering a single weight change (increase in our case), usually only a small part of the graph is affected. It is therefore sensible to avoid the computation of shortest paths from scratch, as it is the case with Dijkstra's algorithm, but only update the part of the graph affected by the arc weight change. This problem is known as the dynamic shortest path problem. Many algorithms were proposed to solve this problem, one of the most famous being the algorithm of Ramalingam and Reps [15]. As suggested in [10], we have used an improved version of this algorithm [8].

The other key operations that is frequently performed is the propagation of flows on shortest paths in order to compute link loads and hence the cost of a neighbouring solution. As was also suggested in [10], this operation can also be optimized when considering a single weight change. The general idea is to re-propagate flows only from nodes for which an ingoing or outgoing link has appeared in or disappeared from the shortest path graph.

The convergence test is based on 3 criteria: (1) a maximum number of iterations $Q_1$, (2) a maximum number of iterations $Q_2$ before improving the best found solution and (3) an empty neighbourhood for the current solution ($\Delta_i = \infty,\ \forall i = 1 \ldots M$). If $Q_2 = 0$, the algorithm converges to a local minimum, but otherwise it can escape from

**Algorithm 1** OSPF weight optimization algorithm

**procedure** METRICOPTIMISATION
  Let $W = (w_1, \ldots, w_M)$ be the initial solution and $\Phi(W)$ its cost.
  $W^* = W$ ▷ Initialisation of minimum cost solution
  **while** Convergence() = false **do**
    $\Phi_{min} = \infty$
    **for** $i = 1 \ldots M$ **do**
      Compute $\Delta_i$ and $W^i = (w_1, w_2, \ldots, w_i + \Delta_i, \ldots, w_M)$
      Compute shortest paths : $\delta^x_{u,v}(W^i),\ n^v_u(W^i) \quad u, v, x = 1 \ldots N$
      Propagate flows : $\gamma^v_u(W^i),\ Y_{u,v}(W^i) \quad u, v = 1 \ldots N$
      Compute cost $\Phi(W^i)$
      **if** $\Phi(W^i) \leq \Phi_{min}$ **then**
        $W_{next} = W^i$ and $\Phi_{min} = \Phi(W^i)$
      **end if**
    **end for**
    $W = W_{next}$
    **if** $\Phi(W) < \Phi(W^*)$ **then**
      $W^* = W$ ▷ Update of minimum cost solution
    **end if**
  **end while**
**end procedure**

a local minimum by selecting the neighbour solution corresponding to the minimum cost increase. The algorithm then stops if no solution better than $W^*$ is found in $Q_2$ iterations.

In the following, we will consider several variations of algorithm 1. First, the algorithm can be used in single-start mode, for incremental weight optimization. In the sequel, we refer to this incremental weight optimization algorithm as algorithm $A_{\mathrm{INCR}}$. The parameters of this algorithm are $Q_1 = 100$ and $Q_2 = 5$.

Algorithm 1 can also be used in multi-start mode. In this case, algorithm 1 is repeated several times, starting from (a) the current weight settings, (b) weights of each link set to 1 and (c) weights of links set inversely proportional to the link capacities. In the sequel, we will consider two such multi-start algorithms:

- Algorithm $A^{\mathrm{stop}}_{\mathrm{MS}}$ is a multi-start algorithm with $Q_1 = 100$ and $Q_2 = 0$. Thus, this algorithms repeatedly starts from the previous initial solutions and stops once in a local minimum.
- Algorithm $A^{\mathrm{min}}_{\mathrm{MS}}$ is a multi-start algorithm with $Q_1 = 100$ and $Q_2 = 5$.

# 4 Results

To assess the performances of the proposed algorithms, we performed 104 tests using 13 synthetic networks and 8 randomly generated traffic matrices for each topology. Let us first assume that cost function 2 is used, i.e. that the goal is to minimize the average delay of a packet in the network. For each test, we have computed the cost of the optimal load-sharing solution using the flow deviation algorithm [7]. This cost is obviously a lower bound for any weight optimization algorithm. We have also computed the costs of the solutions obtained using the following algorithms:

- The two standard heuristics, i.e. link weights set to 1 (UNIT) and link weights set inversely proportional to link capacities (INVCAPA),
- The Fortz and Thorup algorithm (FTA) as described in [10], but limited to 200 iterations and to 20 consecutive iterations without cost evolution (increase or decrease). These limitations were required to avoid excessive computing times,
- Our incremental weight optimization algorithm $A_{\mathrm{INCR}}$, and our multi-start algorithms $A_{\mathrm{MS}}^{\mathrm{stop}}$ and $A_{\mathrm{MS}}^{\mathrm{min}}$.

The initial solution of FTA and $A_{\mathrm{INCR}}$ algorithms is the solution where all weights are set to 1.

For each one of the 104 tests, we have computed the relative gap in percent between the lower bound given by the optimal load-sharing routing and the costs given by the weight optimization algorithms. For each topology and each weight optimization algorithm, table 1 presents the average (AVG) over the 8 traffic matrices defined for each topology and the standard deviation (STD) of these relative gaps. Table 1 also gives the number of routers and the number of network interfaces of each topology.

Table 2 presents the relative increase of computing times of algorithms FTA and $A_{\mathrm{MS}}^{\mathrm{min}}$ with respect to the computing times of algorithm $A_{\mathrm{MS}}^{\mathrm{stop}}$. As previously, the results are averaged over the 8 traffic matrices defined for each topology. The incremental weight optimization algorithm $A_{\mathrm{INCR}}$ is obviously far more faster.

It can be seen the proposed algorithms are always far more faster than FTA and provide better solutions. On the 104 tests, the $A_{\mathrm{MS}}^{\mathrm{min}}$ (resp. $A_{\mathrm{MS}}^{\mathrm{stop}}$) algorithm has reached 17 (resp. 17) times the lower bound, and was 67 times (resp. 66) within 1% of this lower bound. For both multi-start algorithms, the worst solution was within 19% of the lower bound (7989% for FTA). It can also be noticed that the incremental algorithm $A_{\mathrm{INCR}}$ always improves significantly its initial solution (link weights set to 1) and that it is fairly close to the lower bound for 11 of the 13 topologies.

Table 3 provides similar results but for cost function 2. Since this cost function is not continuous, the flow deviation algorithm cannot be used. Therefore, the algorithm $A_{\mathrm{MS}}^{\mathrm{min}}$, which has always given the lowest cost in our experiments, has been used as reference.

| Topologies (#Routers, #Interfaces) | | UNIT (%) | INVCAPA (%) | $A_{\mathrm{INCR}}$ (%) | $A_{\mathrm{MS}}^{\mathrm{stop}}$ (%) | $A_{\mathrm{MS}}^{\mathrm{min}}$ (%) | FTA (%) |
|---|---|---|---|---|---|---|---|
| Topology 1 | AVG | 33.57 | 31.16 | 14.40 | 7.75 | 6.39 | 17.26 |
| (13, 48) | STD | 13.68 | 21.01 | 9.33 | 5.20 | 3.81 | 8.41 |
| Topology 2 | AVG | 186.96 | 3.74 | 4.45 | 0.83 | 0.82 | 4.30 |
| (30, 122) | STD | 47.82 | 1.46 | 2.29 | 0.53 | 0.54 | 1.25 |
| Topology 3 | AVG | 27.93 | 2.58 | 1.71 | 1.05 | 1.04 | 2.26 |
| (50, 148) | STD | 3.29 | 0.66 | 0.47 | 0.37 | 0.37 | 1.16 |
| Topology 4 | AVG | 20383.58 | 0 | 131.38 | 0 | 0 | 1800.96 |
| (8, 22) | STD | 4827.45 | 0 | 38.76 | 0 | 0 | 2759.31 |
| Topology 5 | AVG | 11.23 | 1.82 | 1.97 | 1.57 | 1.57 | 4.44 |
| (7, 16) | STD | 3.26 | 1.11 | 1.24 | 1.30 | 1.30 | 4.49 |
| Topology 6 | AVG | 6346.75 | 0.94 | 940.04 | 0.38 | 0.38 | 1782.46 |
| (15, 42) | STD | 992.50 | 1.35 | 2016.26 | 0.57 | 0.57 | 2596.26 |
| Topology 7 | AVG | 31.57 | 1.03 | 3.60 | 0.24 | 0.24 | 4.28 |
| (12, 46) | STD | 9.41 | 0.80 | 2.04 | 0.15 | 0.15 | 2.97 |
| Topology 8 | AVG | 74.42 | 1.07 | 4.35 | 0.37 | 0.37 | 4.05 |
| (10, 28) | STD | 41.00 | 0.41 | 4.08 | 0.30 | 0.30 | 3.11 |
| Topology 9 | AVG | 9.56 | 2.07 | 3.06 | 1.30 | 1.25 | 3.67 |
| (12, 38) | STD | 1.46 | 1.22 | 1.10 | 1.31 | 1.23 | 1.32 |
| Topology 10 | AVG | 14.74 | 0.33 | 0.11 | 0.09 | 0.09 | 0.29 |
| (21, 46) | STD | 1.75 | 0.49 | 0.06 | 0.07 | 0.07 | 0.30 |
| Topology 11 | AVG | 32.08 | 2.78 | 1.30 | 0.35 | 0.35 | 2.30 |
| (13, 38) | STD | 10.27 | 1.84 | 0.47 | 0.19 | 0.19 | 1.24 |
| Topology 12 | AVG | 4.15 | 0.86 | 1.07 | 0.11 | 0.10 | 1.33 |
| (32, 94) | STD | 1.51 | 0.83 | 0.45 | 0.12 | 0.12 | 0.44 |
| Topology 13 | AVG | 23.49 | 21.73 | 3.26 | 3.01 | 3.01 | 3.96 |
| (8, 24) | STD | 49.33 | 52.36 | 6.42 | 6.52 | 6.52 | 6.23 |

Table 1: Relative gap with respect to the optimal load-sharing routing.

# 5 Conclusion

In this paper, we have presented a local search based algorithm for weight optimization. The originality of the proposed algorithm lies in the neighbourhood structure it uses. One of the main advantage of this algorithm is its versatility since it can be used either in single-start mode, for incremental weight optimization, as well as in multi-start mode, for global weight optimization. Extensive computational results have shown that both the single-start algorithm and the multi-start algorithms perform very well The single-start algorithm enables the network operator to significantly improve the network routing with only a few weight changes. Moreover, this algorithm is generaly very fast. If more weight changes are allowed, the multi-start algorithms allow to get IP routing whose costs are often very close to the lower bound given by the optimal-load sharing routing. The computing times of these algorithms are significantly lower than those of the other algorithms proposed in the litterature.

| Topology | FTA (%) | $A_{\mathrm{MS}}^{\mathrm{min}}$ (%) |
|---|---|---|
| Topology 1 | $1511 \pm 501$ | $123 \pm 47$ |
| Topology 2 | $961 \pm 446$ | $34 \pm 22$ |
| Topology 3 | $2277 \pm 1446$ | $42 \pm 22$ |
| Topology 4 | $1906 \pm 1889$ | $86 \pm 53$ |
| Topology 5 | $2303 \pm 1112$ | $104 \pm 35$ |
| Topology 6 | $1404 \pm 1174$ | $70 \pm 90$ |
| Topology 7 | $843 \pm 168$ | $66 \pm 29$ |
| Topology 8 | $891 \pm 428$ | $78 \pm 41$ |
| Topology 9 | $1408 \pm 478$ | $103 \pm 58$ |
| Topology 10 | $4027 \pm 1377$ | $45 \pm 13$ |
| Topology 11 | $766 \pm 140$ | $35 \pm 9$ |
| Topology 12 | $923 \pm 304$ | $47 \pm 22$ |
| Topology 13 | $1924 \pm 976$ | $87 \pm 27$ |

Table 2: Average & standard deviation of computing time increases compared to $A_{\mathrm{MS}}^{\mathrm{stop}}$

| Topology | UNIT (%) | INVCAPA (%) | $A_{\mathrm{MS}}^{\mathrm{stop}}$ (%) | FTA (%) | $A_{\mathrm{INCR}}$ (%) |
|---|---|---|---|---|---|
| 1 | $18.45 \pm 22.19$ | $19.67 \pm 4.19$ | $0 \pm 0$ | 2.41±4.73 | 0±0 |
| 2 | 637.25±310.87 | 41.59±16.36 | 0±0 | 140.47±272.71 | 158.12±355.28 |
| 3 | 129.69±32.93 | 36.44±7.43 | 1.21±3.21 | 32.90±28.56 | 25.75±18.15 |
| 4 | 28181±5755 | 17.03±14.36 | 0±0 | 6036±11643 | 164.27±50.33 |
| 5 | 33.46±9.14 | 59.45±20.25 | 0±0 | 8.58±15.76 | 0±0 |
| 6 | 14464±19535 | 40.64±123.61 | 0±0 | 8772±7392 | 7753±9264 |
| 7 | 144.68±50.25 | 39.55±24.92 | 0.10±0.25 | 9.89±13.82 | 0.92±1.97 |
| 8 | 169.92±98.22 | 28.65±17.3 | 11.93±21.23 | 36.63±35.28 | 18.95±19.22 |
| 9 | 53.17±27.33 | 40.18±15.58 | 2.28±2.84 | 11.21±6 | 2.14±2.4 |
| 10 | 18.24±22.7 | 36.31±33.35 | 0±0 | 0±0 | 0±0 |
| 11 | 75.23±20.36 | 32.91±18.57 | 0±0 | 20.54±10.59 | 13.79±8.41 |
| 12 | 29.30±18.33 | 43.89±8.68 | 1.27±1.48 | 5.84±4.64 | 0±0 |
| 13 | 13.44±19.6 | 14.7±19.3 | 0±0 | 0.82±1.7 | 31.50±77.16 |

Table 3: Relative gap with respect to $A_{\mathrm{MS}}^{\mathrm{min}}$.

# References


[1] E. Aarts, J. K.Lenstra. *Local Search in Combinatorial Optimization* John Wiley & Sons Ltd., 1997.

[2] Y. Bernet et al. *A framework for differentiated services* Internet Draft, draft-ietf-diffserv-framework-0.1.txt, November 1998.

[3] R. Braden, D. Clark, and S. Shenker. *Integrated services in the internet architecture: an overview* Internet RFC 1633, June 1994.

[4] W. Ben Ameur, E. Gourdin and B. Liau. *Dimensioning of internet networks* Proceedings of the DRCS'2000, Munich, 2000.

[5] W. Ben Ameur and B. Liau. *Calcul des mtriques de routage pour Internet* Annales des Télécommunications, vol. 56, 2001.

[6] W. B. Ameur and N. Michel and B. Liau. *Routing Strategies for IP Networks* XXX FIX ME.

[7] D. P. Bertsekas and R. Gallager. *Data Networks 2nd ed.* Prentice Hall, 1992.

[8] L.S. Buriol, M. G. C. Resende, and M. Thorup. *Speeding up dynamic shortest path algorithms* AT&T Labs Research Technical Report TD-5RJ8B, 2003

[9] A. Farago and b. Szviatovszki. *Allocation of administrative weight in PNNI* Proceedings of NETWORKS'98, Sorrento, 1998.

[10] B. Fortz and M. Thorup. *Internet Traffic Engineering by optimizing OSPF weights* XXX FIX ME.

[11] F. Glover. *Tabu search Part I*. Operations Research Society of America (ORSA) Journal on Computing, Vol. 1, 1989.

[12] F. Glover. *Tabu search Part II*. Operations Research Society of America (ORSA) Journal on Computing, Vol. 2, 1990.

[13] M. Pioro, A. Szentesi, J. Harmatos, A. Jttner, P. Gajowniczek and S.Kozdrowski. *On open shortest path first related network optimization problems* Performance evaluation 48, 2002.

[14] M. Pioro and P. Gajoznicwek. *Solving multicommodity integral flow problems by simulated allocation* Telecommunication Systems 7(1-3), 1997.

[15] G. Ramalingam and T. Reps. *An incremental algorithm for a generalization of the shortest path problem* Journal of Algorithms, 21:267-305, 1996.

[16] D.Burton. *On the inverse shortest path problem* XXX FIX ME.

[17] A. Sridharan, R. Guérin, C. Diot. *Achieving Near Optimal Traffic Engineering Solutions in Current OSPF/ISIS Networks* Proceedings of INFOCOM 2003, San Fransisco, USA, 2003.

[18] Z. Wang, Y. Wang and L. Zhang. *Internet traffic engineering without full mesh overlaying* Proceedings of INFOCOM 2001, Anchorage, Alaska, April 2001.

[19] Internet Engineering Task Force. *Ospf version 2.* Technical Report RFC 1583, Networking Working Group, 1994.

[20] J. T. Moy. *OSPF, Anatomy of an Internet Routing Protocol* Addison-Wesley, 1998.

[21] M. Ericsson, M. G. C. Resende, and P.M. Pardalos. *A genetic algorithm for the weight setting problem in OSPF routing.* Journal of Combinatorial Optimization, 6:299-333, 2002.

[22] M. Ericsson, M. G. C. Resende, and P.M. Pardalos. *A genetic algorithm for the weight setting problem in OSPF routing.* Journal of Combinatorial Optimization, 6:299-333, 2002.

[23] L. S. Buriol, M. G. C. Resende, C. C. Ribeiro and M. Thorup. *A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing.* Accepted for publication in Networks.