

Optimization of Routing Metrics in IP Networks

Fortuny Cédric ¹, Brun Olivier ², and Garcia Jean-Marie ³

¹ LAAS * & QoS Design †

* 7, avenue du Colonel Roche, 31077 Toulouse, France.

† 6, av Marcel Doret, 31500 Toulouse, France.

cfortuny@laas.fr

² LAAS

7, avenue du Colonel Roche, 31077 Toulouse, France.

brun@laas.fr

³ LAAS * & QoSDesign †

* 7, avenue du Colonel Roche, 31077 Toulouse, France.

† 6, av Marcel Doret, 31500 Toulouse, France.

jmg@laas.fr

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.

Keywords: Internet, routing, weight optimization, OSPF, IS-IS, local search.

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 softwares. 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.

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. 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 [8]. Previous works can be classified into two broad approaches. Given an optimal flow allocation [5], 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 [2–4], quadratic programming [14], or greedy heuristics [15]. Given an initial solution, i.e. a set of link weights, the other approach aims at iteratively improving the solution. Fortz and Thorup [8] 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 [18, 19].

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 being 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 literature.

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 $k = 1, \dots, K$ 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 $W = (w_1, \dots, w_M)$, where $w_a \in \Omega$ is the weight assigned to arc $a \in U$ and $\Omega = \{1, \dots, w_{max}\}$ 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 $W = (w_1, \dots, w_M)$, 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 splitted evenly between all outgoing links on the shortest paths towards the destination, the total traffic received at node i for destination x is,

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

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

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

$Y_{i,j}^x(W)$ is the traffic on link i for destination x . Thus, if we are given a weight vector $W = (w_1, \dots, w_M)$, the previous formula allows 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:

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

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

$$\Phi(W) = \frac{1}{\lambda} \sum_{l=1}^M \Phi_l(W) \quad \text{with} \quad \lambda = \sum_k \lambda_{s(k)}^{t(k)} \quad (4)$$

where $\Phi_l(W)$ is the average delay incurred by a packet in interface $l = 1 \dots M$. 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 well known M/M/1/K queueing formula,

$$\Phi_l(W) = \begin{cases} \frac{K_l \rho_l^{K_l+2} - (K_l + 1) \rho_l^{K_l+1} + \rho_l}{Y_l(W) [1 - \rho_l - \rho_l^{K_l+1} + \rho_l^{K_l+2}]} & \text{if } \rho_l \neq 1 \\ \frac{K_l}{2 Y_l(W)} & \text{if } \rho_l = 1 \end{cases} \quad (5)$$

In this case, the cost $\Phi(w)$ represents 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,

$$\Phi(W) = \max_{1 \leq l \leq M} \Phi_l(W) \quad \text{where} \quad \Phi_l(W) = \frac{Y_l(W)}{C_l}, l = 1 \dots M \quad (6)$$

3 A local search algorithm for weight optimization

To solve the weight optimization problem, we propose a local search heuristic [1, 9, 10]. The main originality of this algorithm lies in the neighbourhood structure it uses.

3.1 Neighbourhood definition

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

$$V(W) = \{W^1, W^2, \dots, W^M\} \quad (7)$$

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

$$\Delta_i = \operatorname{argmin}_{\Delta \geq 1} [Y_i(w_1, \dots, w_i + \Delta, \dots, w_M) < Y_i(w)] \quad (8)$$

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 Δ_i can not be defined since it is impossible to deviate traffic from link i . Otherwise, we proceed as follows to compute Δ_i :

- (a) Set $\overline{W}^i = (w_1, \dots, w_{i-1}, \infty, w_{i+1}, \dots, w_M)$.
- (b) For all $u, v \in X$, update distances $D_u^v(\overline{W}^i)$ using a shortest path algorithm

(c) Let,

$$d_{\min}^i = \min_{f \in F_i} \left[D_{s(f)}^{t(f)}(\overline{W}^i) - D_{s(f)}^{t(f)}(W) \right] \quad (9)$$

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

(d) Δ_i is given by,

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

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

- If $d_{\min}^i = 0$, 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 $0 < d_{\min}^i < \infty$, the solution W^i introduces load-sharing for at least one OD flow belonging to F_i ,
- If $d_{\min}^i = \infty$, 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 ∞ in order to compute Δ_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 [13]. As suggested in [8], we have used an improved version of this algorithm [6].

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 [8], 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.

Algorithm 1 OSPF weight optimization algorithm

```

1: procedure METRICOPTIMISATION
2:   Let  $W = (w_1, \dots, w_M)$  be the initial solution and  $\Phi(W)$  its cost.
3:    $W^* = W$  ▷ Initialisation of minimum cost solution
4:   while Convergence() = false do
5:      $\Phi_{min} = \infty$ 
6:     for  $i = 1 \dots M$  do
7:       Compute  $\Delta_i$  and  $W^i = (w_1, w_2, \dots, w_i + \Delta_i, \dots, w_M)$ 
8:       Compute shortest paths :  $\delta_{u,v}^x(W^i), n_u^v(W^i)$   $u, v, x = 1 \dots N$ 
9:       Propagate flows :  $\gamma_u^v(W^i), Y_{u,v}(W^i)$   $u, v = 1 \dots N$ 
10:      Compute cost  $\Phi(W^i)$ 
11:      if  $\Phi(W^i) \leq \Phi_{min}$  then
12:         $W_{next} = W^i$  and  $\Phi_{min} = \Phi(W^i)$ 
13:      end if
14:    end for
15:     $W = W_{next}$ 
16:    if  $\Phi(W) < \Phi(W^*)$  then
17:       $W^* = W$  ▷ Update of minimum cost solution
18:    end if
19:  end while
20: end procedure

```

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 W^* and (3) an empty neighbourhood for the current solution ($\Delta_i = \infty, \forall i = 1 \dots M$). If $Q_2 = 0$, the algorithm converges to a local minimum, but otherwise it can escape from 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_{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) link weights set by the network operator, (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_{MS}^{stop} 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_{MS}^{min} 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

Table 1

Relative gap with respect to optimal load-sharing routing, cost function (4).

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

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

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

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

17. 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.
18. 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.
19. 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.