Dividing the Traffic Matrix to Approach Optimal Traffic Engineering

Simon Balon and Guy Leduc

Research Unit in Networking
EECS Department- University of Liège
Institut Montefiore, B28 - B-4000 Liège - Belgium
{balon,leduc}@run.montefiore.ulg.ac.be

***Abstract*— In this paper we propose a new method to approach optimal Traffic Engineering routing. The method consists of dividing the traffic matrix into N sub-matrices, called strata, and route each of these independently. We propose two different implementations of our method in routers. Our method can also be used to compute a very precise approximation of the optimal value of a given objective function for comparison to heuristic Traffic Engineering algorithms. For this application, our algorithm is very efficient on large topologies compared to an LP formulation.**

***Index Terms*— Traffic Engineering, IP, MPLS, Optimal Routing**

I. Introduction

We consider the traffic engineering routing problem. Given the topology of the network to be engineered and an estimate of the traffic matrix to be routed on it, the problem is to find a routing scheme that optimises the network, with the joint goal of good user performance and efficient use of network resources ([1]). A good routing scheme is usually defined as a routing scheme that minimises a predefined objective function.

The goal of the Traffic Engineering (TE) algorithm is to approximate the optimal routing scheme, i.e. the optimal set of paths. In MPLS networks ([2]), each path (LSP) is established independently. A TE algorithm for such network has to find a path for each (aggregated) traffic flow passing in the network. In OSPF or ISIS networks, it is a set of link metrics that defines the routes used by IP packets. In this case routes are computed using Dijkstra's Shortest Path First (SPF) algorithm possibly using ECMP (Equal Cost MultiPath). A TE algorithm for such networks has to find a good set of link metrics such that corresponding routing scheme is good. We can notice that the set of feasible routing schemes in OSPF or ISIS networks is a subset of the one available for MPLS networks. To find the optimal routing scheme is a complex problem. Usually heuristic algorithms are used. Papers [3], [4], [5], [6], [7], [8] present some heuristic TE algorithms for both MPLS and OSPF/ISIS networks. Most of these algorithms are dedicated to specific TE objective functions.

In this paper we propose to divide the traffic matrix into N equal sub-matrices, called strata, for which the routing scheme can be independently chosen. The sum of the N strata is obviously equal to the original traffic matrix. We compute the routing scheme of each stratum considering the network state resulting of the routing of lower strata. In section II, we present the objective functions we consider in this paper, while our method can be used with other of objective functions. Section III presents the first derivative of the objective functions we consider. In section IV, we propose an algorithm to compute one routing scheme per strata using the derivatives computed in section III. Section VI shows that the total resulting routing scheme is close to the optimum when $N \to \infty$. We have noticed that in practice, a low value of N is sufficient to obtain a routing scheme very close to the optimal one. In section VII we evaluate the efficiency of our algorithm compared to LP formulation. Finally, in section VIII we propose two methods to implement our routing scheme in routers. The first one is to establish N MPLS full-meshes in the network. The second one is to use the IGP Multi-Topology functionality ([9]). Our algorithm provides the paths of all the LSPs for the MPLS solution and the N sets of metrics for the Multi-Topology Routing solution.

S. Balon is a Research Fellow of the Belgian National Fund for the Scientific Research (F.N.R.S).

II. Objective functions

A. Notations

A network is modeled by a directed graph, $G = (N, A)$ whose nodes and arcs represent routers and links. Each arc has a capacity c_a. Traffic on the network is represented by a traffic matrix D that with every pair (s, t) of nodes associates the value of the traffic demand, i.e. the traffic that flows from node s to node t.

Basically, the graph G and the traffic matrix D are the inputs of the problem. A traffic engineering algorithm has to find *good* paths between each pair of source and destination nodes to route corresponding traffic flows. A good set of paths will be a set of paths that optimises a pre-defined objective function.

Once the paths are chosen, we can associate with each arc (a) a load l_a which is the total load on the arc, i.e. the sum over all demands of the amount of traffic sent over a. The utilisation of link a is $u_a = l_a/c_a$.

B. Presented objective functions

In this paper, we consider the objective functions presented in table I. These functions must be minimised by TE algorithms. All these objective functions can be written under the form of $\sum_{a\in A} f_a(l_a)$ and $f_a(x)$ are convex. $MinHop$ is the objective function minimised by a minimum hop routing. A minimum hop route is a route with minimal number of links. $InvCap$ objective function is minimised by a shortest path routing considering a metric of $\frac{1}{c_a}$ for each link a[1]. $WMeanDelay$ minimises the weighted mean delay[2] and is a good TE objective function, according to [10]. $MeanDelay$ is the (unweighted) mean delay. Finally, $NonLinearFortz$ is a non linear function whose linear approximation is introduced in [7] by Fortz and Thorup[3].

	Objective Function
MinHop	$\sum_{a\in A} l_a$
InvCap	$\sum_{a\in A} u_a$
WMeanDelay	$\sum_{a\in A} \frac{l_a}{c_a-l_a} = \sum_{a\in A} \frac{u_a}{1-u_a}$
MeanDelay	$\sum_{a\in A} \frac{1}{c_a-l_a}$
NonLinearFortz	$\sum_{a\in A} \frac{l_a}{1-u_a}$

TABLE I

Objective functions

III. The first derivative of objective functions

The goal of the TE algorithm is to find good paths between each pair of nodes. The idea of our algorithm is that a good path minimises the increase of the score function due to the routing of some traffic on this path. The increment of the cost function of using one particular link on the path of an OD pair is $\sum_{a\in A} f_a(l'_a) - \sum_{a\in A} f_a(l_a)$, if l_a and l'_a are the loads of link a before and after routing some traffic on it. Let x be the load of the link a^* we consider and dx the increment of traffic on this link. The increment is 0 for all other links. Thus $\sum_{a\in A} f_a(l'_a) - \sum_{a\in A} f_a(l_a) = f_{a^*}(x+dx) - f_{a^*}(x)$. The increment is thus $\frac{f_{a^*}(x+dx)-f_{a^*}(x)}{dx}$ per unit of traffic flowing on this link a^*. If we suppose that the traffic increment on this link is sufficiently small, we can assume that the increment of the cost function is $\lim_{dx\to 0} \frac{f_{a^*}(x+dx)-f_{a^*}(x)}{dx} = \frac{\partial f_{a^*}(x)}{\partial x}$. Hereunder we compute this first derivative for all the presented objective functions.

[1] This is proven in section IV.

[2] If we take the delay to be the average delay of an M/M/1 queue, the mean queuing + transmission delay of link a is given by $Delay_a = \frac{mean\_packet\_size}{c_a - l_a}$. For a M/M/1 queue, all the percentiles/quantiles are also proportional to this value.

[3] In [10] we explain why we consider this non-linear function.

$$
\begin{aligned}
MinHop \quad & \frac{\partial f_a(x)}{\partial x} = \frac{\partial x}{\partial x} = 1 \\
InvCap \quad & \frac{\partial f_a(x)}{\partial x} = \frac{1}{c_a} \\
WMeanDelay \quad & \frac{\partial f_a(x)}{\partial x} = \frac{c_a}{(c_a - x)^2} \\
MeanDelay \quad & \frac{\partial f_a(x)}{\partial x} = \frac{1}{(c_a - x)^2} \\
NonLinearFortz \quad & \frac{\partial f_a(x)}{\partial x} = \frac{c_a^2}{(c_a - x)^2}
\end{aligned}
$$

We notice that for $MinHop$ and $InvCap$ objective functions, the first derivative does not depend on x, while it is the case for all the other functions.

IV. Algorithm

We claim that for $MinHop$ and $InvCap$ objective functions, if we use Dijkstra's Shortest Path First (SPF) algorithm taking as link metric the first derivative of the objective function then we obtain the optimal routing scheme, i.e. we find for this routing the minimal value of the objective function.

Theorem 1: A SPF algorithm where link metrics are equal to $\frac{1}{c_a}$ (resp. 1) leads to the minimal value of the $InvCap$ objective $= \sum_{a\in A} u_a$ (resp. $MinHop$ objective $= \sum_{a\in A} l_a$) independently of the traffic matrix.

Proof: We want to minimise the $InvCap$ objective function: $\sum_{a\in A} u_a$. We have

$$
\begin{aligned}
\sum_{a\in A} u_a &= \sum_{a\in A} \frac{\sum_{(s,t)} \delta_{a\in\mathcal{P}_{st}} D_{(s,t)}}{c_a} \\
&= \sum_{(s,t)} D_{(s,t)} \sum_{a\in\mathcal{P}_{st}} \frac{1}{c_a}
\end{aligned}
$$

if $\mathcal{P}_{st}$ is the path from node s to node t and $\delta_{a\in\mathcal{P}_{st}} = 1$ if link a is on the path from s to t and 0 otherwise[4]. $\sum_{(s,t)} D_{(s,t)}$ is constant and so to minimise $\sum_{a\in A} u_a$, we have to minimise $\sum_{a\in\mathcal{P}_{st}} \frac{1}{c_a} \forall s,t$ which is minimised by SPF algorithm taking $\frac{1}{c_a}$ as link metrics. ∎

For the $InvCap$ objective function, we thus find the CISCO recommendation for IGP metric setting ($\frac{1}{c_a}$). The proof can be easily adapted for the $MinHop$ function. In this case $\frac{1}{c_a}$ is replaced by 1, which justifies its name. We see that the $MinHop$ routing thus minimises the total load of the network, as expressed by the $MinHop$ objective function.

Theorem 1 cannot be applied directly with the last three objective functions of table I because they are non linear with respect to the link loads.

[4] If we consider ECMP, $\delta_{a\in\mathcal{P}_{st}}$ is the fraction of traffic sent on a.

A. Dividing the traffic matrix

To approximate infinitesimal increments of traffic for the last three objective functions, we propose to divide the traffic matrix into N equal traffic sub-matrices called strata. The algorithm first computes the paths of the first stratum. Then the algorithm takes into account the link loads induced by the routing of this first stratum to compute the paths of the second stratum, and so on, until the N^{th} stratum. Let OBJ_n be the value of the objective function at step n and OBJ_N its value at the end of the process.

Lemma 1: For large N,

$$OBJ_n - OBJ_{n-1} \approx \sum_{(s,t)} \frac{D_{(s,t)}}{N} \sum_{a \in \mathcal{P}^n_{(s,t)}} f'_a(l_a^{n-1})$$

Proof:

$$\begin{aligned} OBJ_n &= OBJ_{n-1} + \sum_{a \in A} \left[f_a(l_a^n) - f_a(l_a^{n-1}) \right] \\ &= OBJ_{n-1} \\ &+ \textstyle\sum_{a \in A} \left[f_a(l_a^{n-1} + \frac{\sum_{(s,t)} D_{(s,t)} \delta_{a \in \mathcal{P}^n_{(s,t)}}}{N}) - f_a(l_a^{n-1}) \right] \end{aligned}$$

As

$$lim_{\epsilon \to 0} f_a(l_a^{n-1} + \epsilon) = f_a(l_a^{n-1}) + \epsilon \times f'_a(l_a^{n-1})$$

we have for large N:

$$OBJ_n \approx OBJ_{n-1} + \sum_{a \in A} \sum_{(s,t)} \frac{D_{(s,t)}}{N} \delta_{a \in \mathcal{P}^n_{(s,t)}} f'_a(l_a^{n-1})$$

■

Theorem 2: For large N, the routing paths of the n^{th} stratum that minimize $OBJ_n - OBJ_{n-1}$ are the shortest paths according to the following link metrics: $w_a^n = f'_a(l_a^{n-1})$.

Proof: This result is derived directly from Lemma 1 with $\mathcal{P}^n_{(s,t)}$ being the shortest paths with w_a^n as link metrics. ■

From this theorem we can conclude that if the traffic matrix is split into a large number of strata, each new stratum can be routed so as to minimize the increase of the objective function, simply by routing the n^{th} stratum along the shortest paths with link metrics w_a^n. The succession of n such steps, all minimizing the increase of OBJ, is thus expected not to depart too much from the optimum.

The algorithm we propose to implement this method is thus the following (see Algorithm 1). First compute the paths using SPF algorithm with metrics equal to $w_a^1 = \left.\frac{\partial f_a(x)}{\partial x}\right|_{x=0}$. Route the first stratum using these paths and recompute the new metrics considering the load introduced by this routing. Compute the paths for the second stratum using these updated metrics, and so on, until the N^{th} partial traffic matrix. $\delta_{a \in SPF^n_{(s,t)}}$ is equal to 1 if a belongs to the shortest path from s to t considering the set of metrics w_a^n and 0 otherwise[5].

Algorithm 1: Divide TM into N sub-matrices

```
1 $l_a^0 \leftarrow 0 \quad \forall a \in A$;
2 $n \leftarrow 1$;
3 while $n \leq N$ do
4     $w_a^n \leftarrow \left.\frac{\partial f_a(x)}{\partial x}\right|_{x=l_a^{n-1}} \quad \forall a \in A$;
5     $l_a^n \leftarrow l_a^{n-1} + \sum_{(s,t)} \delta_{a \in SPF^n_{(s,t)}} \frac{D_{(s,t)}}{N} \quad \forall a \in A$;
6     $n \leftarrow n + 1$;
7 end
8 $l_a \leftarrow l_a^N \quad \forall a \in A$;
```

V. Simple example

In this section, we present an example of our algorithm running on a simple topology and its limit when $N \to \infty$. We highlight why it is not optimal and when this non-optimality occurs. In this example we use the $MeanDelay$ objective function but the other functions would lead to the same kind of results.

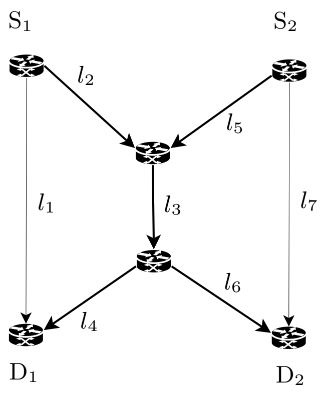

Fig. 1. Simple Example Topology

Consider the topology of figure 1. $c_{l_1} = 11$ Mbps, $c_{l_3} = 10$ Mbps and $c_{l_7} = 9$ Mbps. $c_{l_2} = \infty$, $c_{l_4} = \infty$, $c_{l_5} = \infty$, $c_{l_6} = \infty$[6]. $D_{(S_1,D_1)} = 10$ Mbps and $D_{(S_2,D_2)} = 1$ Mbps. The traffic matrix is empty for all other (source, destination) pairs of nodes. There are two possible paths for traffic from node S_1 to node D_1, the left-hand path l_1 and the right-hand path $l_2 l_3 l_4$. For the traffic from node S_2 to node D_2, the two possible paths are the left-hand path $l_5 l_3 l_6$ and the right-hand path l_7. We can easily compute that the optimal routing scheme occurs if S_1 sends 5.5 Mbps of traffic on its left-hand path and 4.5 Mbps of traffic on its right-hand path while S_2 sends all its

[5]If we consider ECMP, $\delta_{a \in SPF^n_{(s,t)}}$ is the fraction of traffic sent on a considering w_a^n as link metrics.

[6]Instead of infinite capacities, we can also use capacities $\gg \{c_{l_1}, c_{l_3}, c_{l_7}\}$

traffic on its right-hand path. In this case the value of the objective function is equal to $\frac{1}{c_{l_1}-l_{l_1}} + \frac{1}{c_{l_3}-l_{l_3}} + \frac{1}{c_{l_7}-l_{l_7}} = \frac{1}{5.5} + \frac{1}{5.5} + \frac{1}{8} \approx 0.489$.

If we apply our algorithm with $N = 1$, both S_1 and S_2 will send the whole traffic on their left-hand path, because $\frac{1}{11^2} < \frac{1}{10^2}$ and $\frac{1}{10^2} < \frac{1}{9^2}$ (as we consider the $MeanDelay$ objective function, link metrics are $f'_a(l_a) = \frac{1}{(c_a-l_a)^2}$). Thus $OBJ_{N=1} = \frac{1}{1} + \frac{1}{9} + \frac{1}{9} \approx 1.222$. If $N = 2$, the first stratum will be routed on the left-hand path for both commodities and the second one on the left-hand path for (S_1, D_1) and on the right-hand path for (S_2, D_2), because $\frac{1}{6^2} > \frac{1}{9.5^2}$ and $\frac{1}{9.5^2} < \frac{1}{9^2}$. Thus $OBJ_{N=2} = \frac{1}{6} + \frac{1}{4} + \frac{1}{9} \approx 0.528$, which is already far better than $OBJ_{N=1}$.

When N becomes sufficiently large the strata become infinitesimal. Thus the traffic is routed on the left-hand path for both commodities until the portion of traffic routed for both commodities (α) is such that the left-hand path and the right-hand path for (S_1, D_1) have the same cost, i.e. $\frac{1}{(11-10\alpha)^2} = \frac{1}{(10-\alpha)^2}$. This occurs when $\alpha = \frac{1}{9}$. It is already clear that this routing is not optimal because the optimal routing requires S_2 to send all its traffic on the right-hand path. Now the loads on all the links are such that $c_{l_1} - l_{l_1} = 9.889$, $c_{l_3} - l_{l_3} = 9.889$ and $c_{l_7} - l_{l_7} = 9$. If we continue this reasoning until the whole traffic matrix is routed, we can compute that $lim_{N\to\infty} OBJ_N = OBJ_\infty \approx \frac{1}{5.364} + \frac{1}{5.364} + \frac{1}{8.273} \approx 0.494$ which is at 1% from the optimum.

From this example, we can see why our method is not optimal. Our method would be asymptotically optimal if there were only one commodity in the network, i.e. $lim_{N\to\infty} OBJ_N = OBJ_{opt}$ in this case. This is not the case if there are multiple commodities in the network, as we have seen in our example. The non-optimality is due to the non-reevaluation of the paths that were computed in previous steps of the algorithm and whose costs have been increased due to other commodities. Indeed, in our example, if we could change the paths of the first α portion of traffic from S_2 to D_2 when we realize that it is not optimal anymore, we could reach the optimum. Anyway, as we see in our simulations, our algorithm behaves very well in practice.

VI. Simulations

We have tried our algorithm on two real networks. The first topology is the US research network (Abilene). It is composed of 11 nodes and 14 bidirectional links of 10 Gbps each. The second topology is the topology of another operational network which is composed of about 20 nodes and 40 full-duplex links. To build a realistic traffic matrix, we have collected netflow data on each ingress interface of the network and aggregated this information to build a traffic matrix. We have run our simulation on two traffic matrices per topology: one "low" traffic matrix measured during the night and one "high" traffic matrix measured during peak hours. The method used to generate traffic matrices from netflow traces is described in [11].

Figures 2 to 5 present the values of the objective functions when N goes from 1 to 20 ($\frac{OBJ_N}{OBJ_{OPT}}$). The values are relative to the optimal value of the objective function computed with an LP solver. Be aware that the vertical scale is not the same for all the graphs. $MinHop$ and $InvCap$ are not on the figures because obviously these have a value of 1 $\forall N$. Results are presented on figures 2 and 3 for Abilene network and on figures 4 and 5 for the other operational network. We see that in many cases, we do not have to divide the traffic matrix into many strata to obtain a routing scheme close to the optimum.

Please note that the optimal routing for each objective function is different. So it is impossible to compare two points of two different objective functions on the figures presented in this section. Only two points of the same objective function can be compared. In [10] we can find an analysis and comparison of all our objective functions.

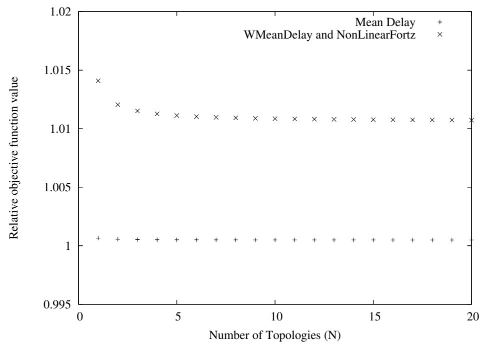

Fig. 2. Abilene Topology ("low" TM)

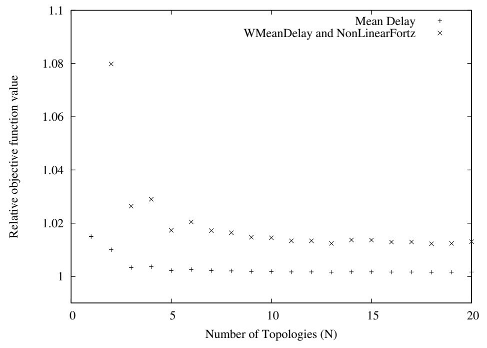

Fig. 3. Abilene Topology ("high" TM)

If we agree to have a precision ϵ of 1.8%, we can use only $N = 1$ for Abilene network. On the other operational network, we have to use at most $N = 3$ to achieve this precision, depending on the chosen objective function.

On Abilene Topology, $WMeanDelay$ and $NonLinearFortz$ provide exactly the same relative values because all the links of this topology have the same capacities. Thus $NonLinearFortz = \sum_{a\in A} \frac{l_a}{1-\frac{l_a}{c_a}} = \sum_{a\in A} \frac{c_a.l_a}{c_a-l_a} = c \times \sum_{a\in A} \frac{l_a}{c-l_a} = c \times WMeanDelay$

On figures 2 and 3 (Abilene network), we can see that there is a gap of about 1% between our solution and the optimum value for $WMeanDelay$ and $NonLinearFortz$[7], while there is almost no gap for $MeanDelay$ objective function. On figures 4 and 5 (operational network), there is almost no gap for any objective function. This highlights that the gap that may be observed depends on the topology and the traffic matrix. For information, the gap is also at most 1% on the other topologies of section VII. We think this is a quite low value.

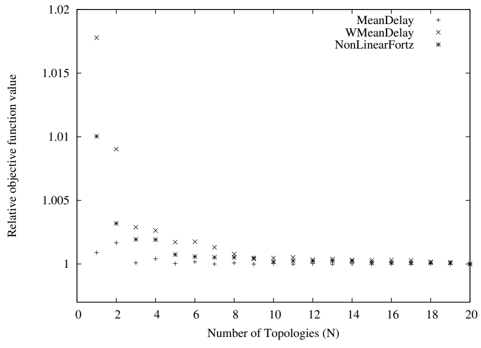

Fig. 4. Operational network Topology ("low" TM)

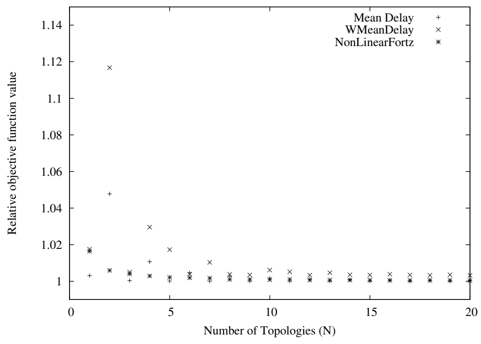

Fig. 5. Operational network Topology ("high" TM)

VII. Efficiency

We have used an LP formulation of the routing problem (as in [7]) to find the optimal routing scheme so that it was possible to measure the gap between our algorithm and the optimum in preceding section. In this LP formulation, we had to linearize our non linear objective functions. As our objective functions are convex and we have a minimization problem, it is possible to replace these functions by piecewise linear approximations. Indeed we just have to add one constraint for each linear piece. These constraints force the objective variable to be greater than or equal to all the linear pieces. Now we have to choose the number of linear pieces of the approximation. Increasing the number of linear pieces in the approximation will increase the

[7]This means that in this case our algorithm converges to a solution which is at about 1 % of the optimum.

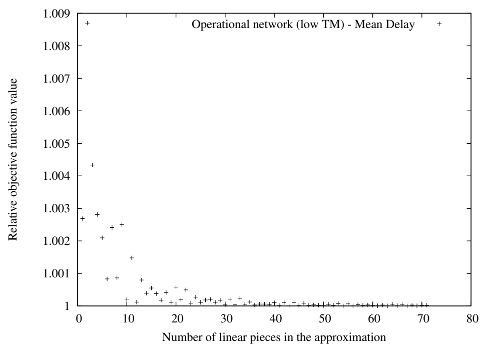

Fig. 6. Precision of LP routing scheme

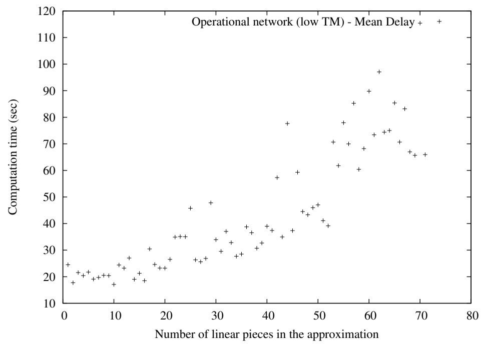

Fig. 7. Computation time of LP routing scheme

quality of the solution found, but it will also increase the running time. Figure 6 shows the precision of LP formulation when we increase the number of lines in the piecewise approximation. We present on this figure relative values to the minimum observed over all tests (in this case the minimum occurs when 51 linear pieces are used). We can see that when more than 30 pieces are used the error is at most 0.02%, which is considered as very good. On figure 7 we can see the computation time when the number of pieces increase. We can notice that the computation time does not increase that much when we increase the number of linear pieces in the approximation. In fact the big problem of LP formulations is the size of the network. Table II shows the computation time of the LP solver compared to our method on Abilene and the operational network, but also on two other generated networks. The two additional networks are generated using the BRITE topology generator ([12]) and the traffic matrix using the TOTEM toolbox ([13]). The results are presented for 20 linear pieces in the approximation for the LP method and when dividing the Traffic Matrix in 20 strata for our method. All the simulation times of this paper are measured on an IBM computer eServer 325 with 2 AMD opteron 2GHz 64 bits processors and 2GB of memory. As LP solver, we have used CPLEX 9.1 (version 64 bits). We can see on the table that our algorithm can be used for large topologies where LP is not usable due to the high running time, although CPLEX is the most powerful solver

Size of the network		Computation time (in sec)	
Nb Nodes	Nb Links	Our method	LP method
11	14	0.549	0.248
~ 20	~ 40	6.563	23.251
30	60	15.475	634.64
35	70	27.410	1322.586

TABLE II

Computation time of our method compared to LP formulation

among those we have tested.

VIII. Implementation on routers

The routing scheme which is found by our method can be implemented on routers using either Multiple Topology Routing ([9]) or multiple MPLS LSP full-mesh.

A. Multiple Topology Routing

In this case, there are two sub-problems. The first problem is to divide the traffic matrix into N sub-matrices. Each router of the network has to map each packet to one of the N topologies. Usually, load balancing is done using a hash function which is based on an identifier of the flow so that all the packets of a flow are forwarded along the same path, thus avoiding packet reordering. The flow id is usually composed of five fields: source and destination addresses, source and destination ports and protocol number.

In our case, the hash function has to be the same in all the routers of the network to avoid cycles. Indeed cycles could appear if one node associates one packet with one topology and the following node associates this same packet with another one. The hash function can be $mod(flow\_id, N)$, for example.

The second problem is to find the N sets of metrics. In our case it is simple. The N sets of metrics are the values of the first derivatives at each step of our algorithm.

B. MPLS full-mesh

It is simpler with MPLS. The paths of the multiple full-mesh are the paths computed at each step when running SPF algorithm on updated metrics. We also have to use a hash function to associate a packet with one of the multiple LSPs available, but in this case, the hash function can be different in each router.

With a triple full-mesh, large backbones with 200-300 egress points would require 600-900 LSP heads at an ingress router. Core routers may need an order of magnitude more transit LSPs. These numbers are far below the thousands of LSP heads and tens-of-thousands transit LSPs that equipment vendors can support today.

IX. Conclusion

Our algorithm provides a good way to approach the optimal routing scheme for an objective function which is on the form of $\sum_{a\in A} f_a(l_a)$ and for which $f_a(x)$ is convex. To approach the optimal routing scheme, we divide the traffic matrix into N equal strata. For any chosen N, we have proposed two methods to implement corresponding routing scheme in the routers: Multi-Topology Routing or MPLS multiple full-mesh. We have highlighted the compromise between low N and good precision. Furthermore, our algorithm can be used on large topologies to compute a near-optimal solution where LP solvers are inefficient. The near-optimal value found by our algorithm can also be used to estimate the quality of a heuristic routing scheme. Moreover, the source code of this algorithm is freely available in the TOTEM Toolbox[13], so using it does not require any expensive license as it is the case for professional LP solvers like CPLEX.

Acknowledgments

This work has been partially supported by the Walloon Region (TOTEM project) and the European Network of Excellence E-NEXT.

References

[1] Daniel O. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for traffic engineering over MPLS. RFC 2702, Internet Engineering Task Force, September 1999.

[2] Eric C. Rosen, Arun Viswanathan, and Ross Callon. Multiprotocol Label Switching Architecture, rfc3031. www.ietf.org, January 2001.

[3] Subhash Suri, Marcel Waldvogel, Daniel Bauer, and Priyank Ramesh Warkhede. Profile-based routing and traffic engineering. *Computer Communications*, 26(4):351–365, March 2003.

[4] M. S. Kodialam and T. V. Lakshman. Minimum interference routing with applications to MPLS traffic engineering. In *Proc. of IEEE INFOCOM*, pages 884–893, 2000.

[5] N. Degrande, G. Van Hoey, P. de La Vallée-Poussin, and S. Van den Busch. Inter-area traffic engineering in a differentiated services network. *J. Networks Syst. Manage.*, 11(4), 2003.

[6] F. Blanchy, L. Mélon, and G. Leduc. An efficient decentralized on-line traffic engineering algorithm for MPLS networks. *Proc. of 18th ITC*, pages 451–460, 2003.

[7] B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *Proc. of IEEE INFOCOM*, pages 519–528, 2000.

[8] E. Mulyana and Killat. U. An offline hybrid IGP/MPLS traffic engineering approach under LSP constraints. In *Proceedings of the 1st International Network Optimization Conference INOC 2003*, Evry/Paris France, oct 2003.

[9] Tony Przygienda, Naiming Shen, and Nischal Sheth. M-ISIS: Multi Topology (MT) Routing in IS-IS. Internet draft, draft-ietf-isis-wg-multi-topology-11.txt, work in progress, October 2005.

[10] S. Balon, F. Skivée, and G. Leduc. How Well Do Traffic Engineering Objective Functions Meet TE Requirements? In *Proceedings of IFIP Networking 2006*, Coimbra, May 2006.

[11] S. Uhlig, B. Quoitin, S. Balon, and J. Lepropre. Providing public intradomain traffic matrices to the research community. *ACM SIGCOMM Computer Communication Review*, 36(1):83–86, January 2006.

[12] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. BRITE: An Approach to Universal Topology Generation. In *In Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems - MASCOTS '01*, Cincinnati, Ohio, Août 2001.

[13] http://totem.run.montefiore.ulg.ac.be.