Link Load Balancing Optimization of Telecommunication Networks: a Column Generation based Heuristic Approach

Dorabella Santos*, Amaro de Sousa†, Filipe Alvelos‡ and Michał Pióro§,¶
*Instituto de Telecomunicações, 3810-193 Aveiro, Portugal
Email: dorabella@av.it.pt
†Instituto de Telecomunicações/DETI
Universidade de Aveiro, 3810-193 Aveiro, Portugal
Email: asou@ua.pt
‡Centro Algoritmi/DPS
Universidade do Minho, 4710-057 Braga, Portugal
Email: falvelos@dps.uminho.pt
§Institute of Telecommunications
Warsaw University of Technology, 00-665 Warsaw, Poland
Email: mpp@tele.pw.edu.pl
¶Department of Electrical and Information Technology
Lund University, 221-00 Lund, Sweden
Email: mpp@eit.lth.se

***Abstract*—This paper deals with optimal load balancing in telecommunication networks. For a capacitated telecommunications network with single path routing and an estimated traffic demand matrix, we wish to determine the routing paths aiming at min-max optimization of link loads. To solve this problem, we propose a column (path) generation based heuristic. In the first step, we use column generation to solve a linear programming relaxation of the basic problem (obtaining a lower bound and a set of paths). In the second step, we apply a multi-start local search heuristic with path-relinking to the search space defined by the paths found in the first step. In order to assess the merits of this approach, we also implemented a search heuristic which is equivalent to the second step of the proposed one but with no constraints on the set of paths that can be used. Through a set of computational results, we show that the proposed heuristic is efficient in obtaining near optimal routing solutions within short running times. Moreover, the comparison of the two heuristics show that constraining the search space to the columns given by column generation gives better results since this solution space contains good quality solutions and, due to its size, enables to find them in short running times.**

***Index Terms*—Link Load Balancing Optimization, Column Generation based Heuristics, Routing, Traffic Engineering**

I. Introduction

Consider a given capacitated telecommunications network that supports a set of traffic flows. Each traffic flow has an estimated demand bandwidth which must be routed through a single network path. Examples of such telecommunication networks are MPLS networks [1], [2] and more recent Ethernet networks based on PBB-TE technology [3]. In both cases, an explicit route must be configured in the network for each traffic flow. The aim is to determine a routing path for each traffic flow so that the traffic load is balanced as much as possible over all network links. The single path routing variant is preferred by many network operators for different reasons: it minimizes the size of the routing tables, it avoids processing overhead of traffic splitting rules, it introduces less jitter, etc.

The link load balance optimization is an important traffic engineering objective to maximize the robustness of the network to unpredictable traffic growth. Assume the worst case scenario where all traffic demands grow simultaneously. If the worst link load is a, with $0 \leq a \leq 1$, then all traffic demands can simultaneously grow up to $(1-a)/a$ before the network becomes saturated and, therefore, the lower the value of a is, the more robust the network becomes to unpredictable traffic growth. Moreover, for a solution with the minimum worst link load a, if the second worst link load is b, with $0 \leq b \leq 1$, then all traffic demands that do not use the worst load link can uniformly grow up to $(1-b)/b$ before the network becomes saturated. Therefore, the value b should also be minimized provided that the worst link load a is kept at its minimum. This idea can be generalized to all other link loads, defining the min-max optimization of link loads: first minimize the worst case link load; among all such solutions, minimize the second worst case link load; among all such solutions, minimize the third worst case link load; and so on.

One approach proposed in [4], [5] to define a load balance optimization function (which has been used by other authors) is to minimize the summation of the individual link costs

978-1-4244-6705-1/10/$26.00

where the cost of a link is an increasing piecewise linear function of its load (it was originally proposed for shortest path routing). The main merit of this proposal is that the optimization problem can be addressed by integer linear programming. Nevertheless, the resulting optimization models are hard to solve and their solutions may not correspond to the optimal solutions of the min-max optimization of link loads. For example, consider a network with 5 links and a set of traffic flows such that there are 2 possible routing solutions: A with link load values 0.7, 0.4, 0.4, 0.4 and 0.3, and B with link load values 0.5, 0.5, 0.5, 0.5 and 0.4. With the objective function proposed in [4], A is the selected solution since it has a value of 3.57 while B has a value of 3.87 but, clearly, B is the best solution since it has a much lower worst load.

The min-max optimization of link loads is a non-linear objective but it can be theoretically solved through a set of integer linear programming models in sequence. This objective has been previously addressed for multiple spanning tree based routing [6], [8] and for routing with path protection [7]. Note that we have seen in [8] that the optimization of the average link load is very penalizing for the worst link load values, while the min-max optimization of link loads usually lead to very small penalties on the average link load values.

The min-max optimization of link loads is related to the concept of lexicographical minimization and this objective function is similar to that of max-min fairness (MMF) previously applied to routing and allocation of network resources [9], [10], [11]. General issues on MMF are also discussed in [12]. More applications of the MMF solutions in telecommunication network design can be found in [13], [14], [15].

To solve our problem, we propose a column generation based heuristic which runs in two steps. In the first step, we use column generation (CG) to solve the linear programming relaxation of the problem (it corresponds to the splittable path routing variant of the problem). In this step, we still have to solve a set of optimization problems in sequence (one for each load) but since the problems are linear, the whole set of problems run in short running times. Moreover, the solution obtained in this step is a lower bound for the original problem which is used to assess the quality of the heuristic solutions. In the second step, we apply a multi-start local search heuristic with path-relinking [17], to the search space defined by the columns found in CG.

The proposed approach is part of a more general effort to explore the combination of CG and meta-heuristics. Although the combination of exact and approximate methods has been very active in recent years (see, for example, [18]), attempts to systematically combine CG and meta-heuristics are rare (exceptions are [19] and [20]). The rationale for this combination lies in the fact that often CG provides tight lower bounds (in minimization problems) on the optimal value and that the columns belonging to the restricted master problem (RMP) containing the optimal solution define a search space where high-quality solutions exist. In order to assess the merits of such approach, we also implemented two variants of a search heuristic which is equivalent to the second step of the proposed one but with no constraints on its search space.

This paper is organized as follows. Section II defines the optimization problem and describes how it can be solved through mathematical programming. Section III presents the proposed heuristics. Section IV describes a set of case studies and presents the computational results obtained by the heuristics together with their analysis. Finally, Section V presents the main conclusions and identifies topics for future research.

II. Problem Formulation

Consider a network given by a graph $G(N, A)$ where N is the set of nodes and A is the set of links between the nodes. The edge between nodes $i \in N$ and $j \in N$ is denoted by $\{i, j\}$ and each edge $\{i, j\} \in A$ has a given capacity $c_{\{ij\}}$. There is a set of commodities K, where each commodity is to be routed in a single path through the network. Each commodity $k \in K$ is characterized by its origin node $o_k \in N$, its destination node $d_k \in N$ and its demand $b_k > 0$.

Let $\mathcal{P}_k$ be the set of paths available in graph G between the end nodes of $k \in K$ and let $\delta^{pk}_{\{ij\}}$ be a binary parameter that is 1 if edge $\{i, j\} \in A$ belongs to path $p \in \mathcal{P}_k$. Consider the following decision variables: the binary variable φ^p_k which is 1 if path $p \in \mathcal{P}_k$ is chosen as the routing path of commodity $k \in K$; and the real variable $\mu_{\{ij\}}$ which accounts for the load on link $\{i, j\} \in A$. The set of constraints defining the optimization problem are given by

$$\sum_{p \in \mathcal{P}_k} \varphi^p_k = 1 \qquad \forall k \in K \tag{1}$$

$$\sum_{k \in K} \sum_{p \in \mathcal{P}_k} b_k \delta^{pk}_{\{ij\}} \varphi^p_k = c_{\{ij\}} \mu_{\{ij\}} \quad \forall \{i, j\} \in A \tag{2}$$

$$\varphi^p_k \in \{0, 1\},\ \mu_{\{ij\}} \in [0, 1] \tag{3}$$

where constraints (1) guarantee that exactly one path of $\mathcal{P}_k$ is chosen for every commodity $k \in K$, and constraints (2) account for the link loads.

The min-max optimization of link loads is closely related to the concept of lexicographical optimization. Given two vectors $\boldsymbol{a} = (a_1, ..., a_m)$ and $\boldsymbol{b} = (b_1, ..., b_m)$, vector $\boldsymbol{a}$ is lexicographically smaller that $\boldsymbol{b}$, if either $a_1 < b_1$ or there exists an index $l \in \{1, ..., m-1\}$ such that $a_i = b_i$ for all $i \leq l$ and $a_{l+1} < b_{l+1}$. Now consider the vector of link loads $\boldsymbol{\mu} = (\mu_{\{ij\}} : \{i, j\} \in A)$ and let $[\boldsymbol{\mu}]$ be the vector obtained from $\boldsymbol{\mu}$ by rearranging its elements in non-increasing order. The min-max optimization problem can be defined in a non-linear manner as

$$\mathbf{lexmin}[\boldsymbol{\mu}]$$
$$\text{s.t. } (1) - (3)$$

where $\mathbf{lexmin}$ denotes the lexicographical minimization, i.e. finding a vector $[\boldsymbol{\mu}^*]$ which is lexicographically minimal among all possible vectors $[\boldsymbol{\mu}]$. An optimal solution of the problem can be obtained by solving a sequence of mixed integer linear problems, using the conditional means approach

[16], [10], in the following way. Consider the vector $\boldsymbol{\theta} = (\theta_l : l = 1, ..., |A|)$ whose elements are given by the accumulated elements of $[\boldsymbol{\mu}]$, $\theta_l = \sum_{t=1}^{l} [\boldsymbol{\mu}]_l$, for $l = 1, ..., |A|$. The min-max optimization problem is equivalent (in the sense that the optimal solution set is the same) to

$$\textbf{lexmin}\, \boldsymbol{\theta}$$
$$\text{s.t. } (1) - (3)$$

Following [10], we consider the additional real variables r_t for $t = 1, ..., |A|$, and $d_{t\{ij\}}$ for $t = 1, ..., |A|$ and $\{i, j\} \in A$. Then, the l^{th} element of $\boldsymbol{\theta}$ is given by the optimal solution of the following mixed integer linear problem

$$\theta_l^* = \min \left(l\, r_l + \sum_{\{i,j\} \in A} d_{l\{ij\}} \right) \tag{4}$$
s.t.
$$(1) - (3)$$
$$\mu_{\{ij\}} \leq r_t + d_{t\{ij\}} \qquad \forall t \in 1, 2, ..., l, \forall \{i, j\} \in A \tag{5}$$
$$t\, r_t + \sum_{\{i,j\} \in A} d_{t\{i,j\}} \leq \theta_t^* \forall t \in \{1, 2, ..., l-1\} \tag{6}$$
$$r_t \geq 0,\, d_{t\{ij\}} \geq 0 \tag{7}$$

where the set of constraints (6) guarantee that the previously obtained objectives are not jeopardized. At the end, the worst link load value $[\boldsymbol{\mu}]_1$ is given by θ_1^*, and the l^{th} worst link load value $[\boldsymbol{\mu}]_l$, for $l > 1$, is given by $\theta_l^* - \theta_{l-1}^*$.

III. Heuristics

The min-max optimization of link loads objective is computationally hard. This has motivated the search for heuristics to obtain good solutions. We propose a CG based heuristic which is a hybridization between CG and local search with path-relinking. The proposed heuristic (named Algorithm A) has two phases: COLUMN_GENERATION and SEARCH.

In the COLUMN_GENERATION phase, we solve through CG the linear programming relaxation of the problem, i.e., we let variables φ_k^p be real between 0 and 1 (please see [9] for a general description of CG technique). The optimal solution is a lower bound for the original problem and, therefore, can be used to assess the quality of the heuristic solutions. In this phase, we apply CG in solving the same set of optimization problems as defined in section II, but since all problems are linear, the whole set is solved in short running times. The columns determined in the COLUMN_GENERATION phase define the sets $\mathcal{P}_k$ which are to be input to the SEARCH phase.

In the SEARCH phase, we use a multi-start local search heuristic with path-relinking on the solution space defined by the sets $\mathcal{P}_k$ computed in the first phase. At each local search procedure, we first build an initial solution by selecting randomly a path $p \in \mathcal{P}_k$ for each commodity $k \in K$ and, then, we apply local search using a neighbor set defined as all solutions which differ from the current one in a single path. At the end of the first local search procedure, the solution is kept as the elite solution. At the end of each local search procedure after the first, we apply path relinking between its solution and the elite solution (path-relinking is done in both directions), and keep the best one as the elite solution. Note that in general, path relinking might use a list of elite solutions (please see [17] for a general description of these heuristic techniques) but our computational experience shows that good performance is achieved with a single elite solution.

In order to assess the merits of Algorithm A, we also implemented two variants of a multi-start local search heuristic with path-relinking (named Algorithm B1 and Algorithm B2) with no constraints on its search space. Note that it is not possible to use directly the SEARCH phase algorithm in this case since the complete sets $\mathcal{P}_k$ of a given graph are exponentially sized (which is the reason why this problem is combinatorial in nature, and, therefore, hard to solve). We have to make appropriate adaptations on how the initial solutions of each local search procedure are computed and on how neighbor solutions are defined at each local search step (the path relinking remains the same).

Concerning the initial solutions of each local search procedure of Algorithms B1 and B2, they are computed by first selecting randomly an order of the commodities $k \in K$ and then applying the following greedy procedure:

1. **For** all $\{i, j\} \in A$ **do:**
2. $\quad \gamma_{\{ij\}} = 0$
3. **For** all $k \in K$, following the selected order **do:**
4. $\quad$ **For** all $\{i, j\} \in A$ **do:**
5. $\quad\quad \mu_{\{ij\}} = \gamma_{\{ij\}} + b_k / c_{\{ij\}}$
6. $\quad$ Compute $[\boldsymbol{\mu}]$
7. $\quad$ Set $\alpha = 1.0$
8. $\quad$ **For** $l = 1, ..., |A| - 2$ **do:**
9. $\quad\quad$ **If** $[\boldsymbol{\mu}]_l \neq [\boldsymbol{\mu}]_{l+1}$ **then:**
10. $\quad\quad\quad$ **If** $([\boldsymbol{\mu}]_l)^\alpha \leq (|A| - l)\, ([\boldsymbol{\mu}]_{l+1})^\alpha$ **then:**
11. $\quad\quad\quad\quad \alpha = z \cdot \log(|A| - l) / \log([\boldsymbol{\mu}]_l / [\boldsymbol{\mu}]_{l+1})$
12. $\quad$ Set $\delta_{\{ij\}}$ equal to 1 for the links in the minimum cost path of k using link costs $\left(\mu_{\{ij\}}\right)^\alpha$
13. $\quad$ **For** all $\{i, j\} \in A$ **do:**
14. $\quad\quad \gamma_{\{ij\}} = \gamma_{\{ij\}} + \delta_{\{ij\}} b_k / c_{\{ij\}}$

Initially, the link loads are set to zero (lines 1-2). Following the previous selected order (line 3), a minimum cost path is computed for each k (line 12) and the bandwidth of k is added to the load of the edges of the minimum cost path (lines 13-14). To compute the minimum cost path (line 12), we use the well known Dijkstra algorithm. Nevertheless, this algorithm assumes that the cost of a path is given by the sum of all link costs composing it. Since our aim is the min-max optimization of link loads, the edge costs are set to $\left(\mu_{\{ij\}}\right)^\alpha$. This value is the load that the edge will have, if it is used to route commodity k (lines 4-5), to the power of a value α which is calculated in such a way (lines 6-11) that the result is a min-max cost path (see Appendix for details). In this procedure, z (line 11) is a parameter that must be set to a value slightly larger than 1.0 (in our implementation, we have set $z = 1.0000001$).

The difference between Algorithms B1 and B2 is on how the order of the commodities $k \in K$ is computed to obtain the initial solution of each local search procedure. In Algorithm B1, all orders have equal probability while in Algorithm B2, the commodities k with higher values of demand b_k have more probability of being positioned before the others.

Concerning how neighbor solutions are defined at each local search, both algorithms B1 and B2 use the following strategy. For each current solution, there is one neighbor solution for each $k \in K$ which is computed by: (i) removing its bandwidth demand from its current path and (ii) computing a new minimum cost path using the same method as described in the previous greedy algorithm (lines 4-14).

All algorithms (A, B1 and B2) run until a running time limit (defined at the beginning) is reached. At the end, the last elite solution is the result of the algorithm.

IV. Computational Results

In order to test the proposed algorithms, we have defined a set of 4 case studies based on the well known network topology of NSF network (depicted in Figure 1) where all links are considered to have a capacity of 10000 Mbps.

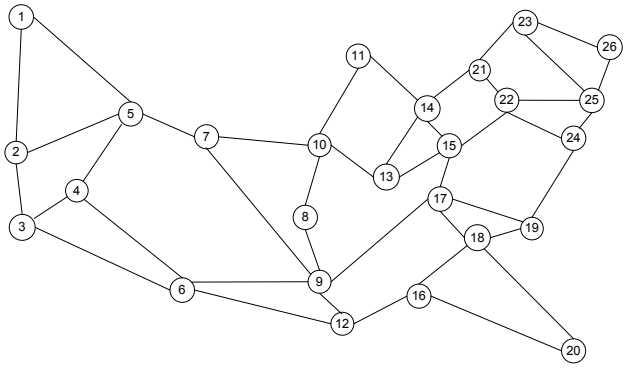

Fig. 1. NSF Network: 26 nodes and 42 links

Based on this network, we have generated 4 different traffic matrices, with commodities between all origin-destination pairs (giving a total of 325 commodities).

In the first case study, named NSFa, all commodity demand values were randomly generated with a uniform distribution with values multiple of 10 Mbps between 20 and 180 Mbps.

In the other 3 case studies, named NSFb, NSFc and NSFd, we have generated the traffic matrices in the following way. First, we have selected a set $N_s \in N$ composed by 6 nodes. Then, we have set the commodity demand values with: 400 Mbps (when both end nodes belong to N_s), 140 Mbps (when one of the end nodes belong to N_s) and 50 Mbps (when none of the end nodes belong to N_s). Sets N_s are $\{1, 3, 5, 6, 10, 12\}$ for NFSb (nodes concentrated on the left part of the network), $\{5, 6, 10, 12, 14, 18\}$ for NFSc (nodes concentrated on the central part of the network) and, $\{2, 8, 11, 20, 22, 26\}$ for NFSd (nodes distributed evenly over all the network).

All algorithms were developed in C++ programming language and were run on the same computational platform with a running time limit of 10 minutes. The running time of COLUMN_GENERATION phase of Algorithm A was never higher than 10 seconds for all 4 case studies, which confirms that the linear programming relaxation can be easily solved by CG. Since the proposed algorithms are stochastic processes, they give, in general, different solutions in different runs. Therefore, we have run each algorithm 10 times for all case studies. The computational results are summarized in Table I for NFSa, Table II for NFSb, Table III for NFSc and Table IV for NFSd. All tables show the 10 worst link load values of the obtained solutions.

Worst	A				B1	B2
Load	LP	RMP	Best	Worst	Best	Best
1st	43,70%	43,70%	**43,70%**	**43,70%**	**43,70%**	**43,70%**
2nd	43,60%	43,60%	**43,60%**	**43,60%**	**43,60%**	**43,60%**
3rd	43,60%	43,60%	**43,60%**	**43,60%**	**43,60%**	**43,60%**
4th	43,60%	43,60%	**43,60%**	**43,60%**	**43,60%**	**43,60%**
5th	42,40%	42,40%	**42,40%**	42,50%	43,00%	42,90%
6th	42,30%	42,30%	**42,30%**	42,40%	42,90%	42,70%
7th	42,30%	42,30%	**42,30%**	42,40%	42,80%	42,70%
8th	42,30%	42,30%	**42,30%**	42,40%	42,40%	42,60%
9th	41,60%	41,60%	**41,60%**	41,60%	42,40%	42,50%
10th	41,40%	41,40%	**41,40%**	41,40%	42,30%	42,50%

TABLE I
COMPUTATIONAL RESULTS FOR NSFA

Worst	A				B1	B2
Load	LP	RMP	Best	Worst	Best	Best
1st	52,20%	52,20%	**52,20%**	**52,20%**	**52,20%**	**52,20%**
2nd	52,10%	52,10%	**52,10%**	**52,10%**	**52,10%**	**52,10%**
3rd	52,10%	52,10%	**52,10%**	**52,10%**	**52,10%**	**52,10%**
4th	42,90%	43,00%	**43,00%**	**43,00%**	43,40%	43,50%
5th	42,90%	42,80%	**42,80%**	**42,80%**	43,10%	43,10%
6th	42,40%	42,40%	**42,40%**	42,50%	43,00%	43,10%
7th	42,40%	42,40%	**42,40%**	42,40%	42,90%	43,00%
8th	42,40%	42,40%	**42,40%**	42,40%	42,60%	43,00%
9th	42,40%	42,40%	**42,40%**	42,30%	42,50%	42,90%
10th	42,30%	42,30%	**42,30%**	42,20%	41,20%	42,80%

TABLE II
COMPUTATIONAL RESULTS FOR NSFB

All tables show the best and the worst among all 10 solutions of algorithm A and only the best among all 10 solutions of algorithms B1 and B2. In order to understand the meaning of the two additional columns (LP and RMP), note first that the demand values of all traffic matrices are multiple of 10 Mbps and, since all link capacities are 10000 Mbps, the possible link load values $\mu_{\{ij\}}$ are multiples of 0,1%. We have used this fact to improve the lower bounds obtained in the COLUMN_GENERATION phase of Algorithm A using a lifting technique previously proposed in [6]. We substitute constraints (2) with their inequality version

$$\sum_{k \in K} \sum_{p \in \mathcal{P}_k} b_k \delta^{pk}_{\{ij\}} \varphi^p_k \leq c_{\{ij\}} \mu_{\{ij\}} \quad \forall \{i,j\} \in A$$

and apply the following lifting technique: the optimal solution value θ^*_l obtained by solving model (4) in iteration l is lifted

Worst	A				B1	B2
Load	LP	RMP	Best	Worst	Best	Best
1st	43,80%	43,80%	**43,80%**	**43,80%**	**43,80%**	**43,80%**
2nd	43,70%	43,70%	**43,70%**	**43,70%**	**43,70%**	**43,70%**
3rd	43,70%	43,70%	**43,70%**	**43,70%**	**43,70%**	**43,70%**
4th	42,50%	42,50%	**42,50%**	**42,50%**	42,60%	42,60%
5th	42,50%	42,50%	**42,50%**	**42,50%**	42,50%	42,50%
6th	42,50%	42,50%	**42,50%**	**42,50%**	42,40%	42,40%
7th	42,40%	42,40%	**42,40%**	**42,40%**	42,40%	42,40%
8th	42,10%	42,10%	**42,10%**	42,20%	42,20%	42,30%
9th	41,50%	41,50%	41,60%	41,80%	42,10%	41,90%
10th	41,50%	*	41,60%	41,70%	41,90%	41,70%

* Reached timelimit of 24 hours

TABLE III
COMPUTATIONAL RESULTS FOR NSFC

Worst	A				B1	B2
Load	LP	RMP	Best	Worst	Best	Best
1st	43,80%	43,80%	**43,80%**	**43,80%**	**43,80%**	**43,80%**
2nd	43,70%	43,70%	**43,70%**	**43,70%**	**43,70%**	**43,70%**
3rd	43,70%	43,70%	**43,70%**	**43,70%**	**43,70%**	**43,70%**
4th	42,40%	42,40%	**42,40%**	**42,40%**	42,80%	42,70%
5th	42,40%	42,40%	**42,40%**	**42,40%**	42,50%	42,30%
6th	42,40%	42,40%	**42,40%**	**42,40%**	42,30%	42,30%
7th	42,30%	42,30%	**42,30%**	**42,30%**	42,00%	42,30%
8th	42,00%	42,00%	**42,00%**	**42,00%**	42,00%	42,30%
9th	41,90%	41,90%	**41,90%**	**41,90%**	41,90%	42,30%
10th	41,90%	41,90%	**41,90%**	**41,90%**	41,80%	42,20%

TABLE IV
COMPUTATIONAL RESULTS FOR NSFD

(rounded up) to the lowest multiple of 0,1% higher than θ_l^*, and then used in constraint (6) in the subsequent iterations. The LP columns present the lower bounds obtained by CG with this lifting technique.

The RMP column is the solution obtained by solving through mathematical programming (we have used CPLEX 12.1) the original problem with the paths obtained by CG, i.e., solving the integer restricted master problem (RMP). Although in the general case, this problem is still very hard to solve, we have used the granularity of 0,1% of link load values $\mu_{\{ij\}}$ to set the absolute MIP gap tolerance parameter of CPLEX. Using this setting, we were able to find within a time limit of 24 hours many of the worst link load values. The worst case was NSFc where we have found only the 9 worst link load values. Note that this solution is the best that can be obtained by the heuristics. In order to help the analysis, the link load values of the heuristics that are equal to the values given by the integer RMP are highlighted in bold on the tables.

The first important observation from the results is that the optimal solutions of the constrained space (RMP results) are quite close to the lower bounds (LP results), which clearly indicates that the constrained search space includes high-quality solutions.

The second important observation is that Algorithm A is able to find high-quality solutions within the given time limit, since that even the worst solution (out of 10 runs) of each case study has always the best values in at least the 4 worst link loads, although on average the solutions exhibit an higher number of best values.

The third important observation is that the worst solution out of 10 runs of Algorithm A is always better than the best solutions of Algorithms B1 and B2. Note that the search space of algorithms B1 and B2 necessarily contains the search space of algorithm A and, therefore, the only reason for the superior performance of algorithm A is because its search space contains high-quality solutions that, due to its size, can be found in short running times.

Note that the total number of paths given by the COLUMN_GENERATION phase of Algorithm A in our runs were 1333 for NSFa, 1063 for NSFb, 1095 for NSFc and 1178 for NSFd. These numbers represent an average number of paths per commodity around 4 in all case studies, which confirms that the constrained search space is a very small subset of the whole search space.

As a concluding remark, we observe that the results of Algorithm B1 have an average quality similar to the results of Algorithm B2 (tables show only the best out of 10 solutions), leading to the conclusion that the two strategies to generate the order of the commodities in the greedy procedure (that computes the initial solutions) produce no significant differences on the obtained solutions.

V. CONCLUSIONS

For a capacitated telecommunications network with single path routing and an estimated traffic demand matrix, we have addressed the problem of how to determine the routing paths aiming the min-max optimization of link loads. We have first described the problem by mathematical programming. Then, we have proposed a column generation based heuristic which is a hybridization between column generation and local search with path-relinking. In our proposal, we use column generation to solve the linear programming relaxation of the problem and, then, we apply a multi-start local search heuristic with path-relinking to the solution space defined by the paths computed in column generation.

In order to assess the merits of our approach, we have also implemented two variants of a multi-start local search heuristic with path-relinking with no constraints on its solution space. The computational results have shown that constraining the search space of the heuristic to the paths given by the column generation gives much better results since this solution space contains good quality solutions and, due to its size, enables to find them in much shorter running times. Moreover, the computational results have also shown that the proposed heuristic is quite efficient in obtaining near optimal routing solutions within short running times.

Concerning future work, note first that, in the general case, the lower bounds given by the column generation might be not so close to the obtained solutions as the ones obtained in our case studies. Therefore, we should test the algorithms in other

case studies with different topological and/or traffic demand characteristics.

Finally, note that the min-max optimization of link loads has been previously addressed as a traffic engineering objective in the cases of (i) multiple spanning tree based routing networks [6], [8] and (ii) single path routing with path protection [7]. It will be interesting to investigate how the column generation based heuristic approach proposed here for the simplest case of single path routing can be generalized to these more complex traffic engineering variants and if its efficiency will be at least as good as in the present case.

ACKNOWLEDGMENT

This work has been conducted under the project PTDC/EIA-EIA/100645/2008 "SearchCol: Meta-heuristic Search by Column generation" (funded by FCT) and under the European FP7 Network of Excellence "Euro-NF". Dorabella Santos was funded by Portuguese FCT under post-doc grant SFRH/BPD/41581/2007. Michał Pióro was funded by Polish Ministry of Science and Higher Education under research grant N517 397334.

REFERENCES

[1] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell and J. McManus, "Requirements for Traffic Engineering Over MPLS", RFC 2702, September 1999

[2] X. Xiao, A. Hannan, B. Bailey and L. Ni, "Traffic Engineering with MPLS in the Internet", IEEE Network, Vol. 14, No. 2, pp. 28-33, 2000

[3] IEEE Standard 802.1Qay, "Provider Backbone Bridge - Traffic Engineering", 2009

[4] B. Fortz and M. Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights", in Proc. 19th IEEE Conf. on Computer Communications (INFOCOM), pp. 519-528, 2000

[5] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS Weights in a Changing World", IEEE Journal on Selected Areas in Communications, Vol. 20, No. 4, pp. 756-767, 2002

[6] D. Santos, A. de Sousa, F. Alvelos, M. Dzida and M. Pióro, "Optimization of link load balancing in multiple spanning tree routing networks", Telecommunication Systems, Springer, available online, June 2010, doi: 10.1007/s11235-010-9337-8

[7] A. de Sousa, D. Santos, P. Matos and J. Madeira, "Load Balancing Optimization of Capacitated Networks with Path Protection", in Proc. of International Symposium on Combinatorial Optimization, Hammamet, Tunisia, 2010

[8] D. Santos, A. de Sousa, F. Alvelos, M. Dzida, M. Pióro and M. Zagożdżdon, "Traffic Engineering Of Multiple Spanning Tree Routing Networks: the Load Balancing Case", Next Generation Internet Networks (NGI 09), IEEE Xplore, Aveiro, Portugal, 2009

[9] M. Pióro and D. Medhi, "Routing, Flow and Capacity Design in Communication and Computer Networks", Morgan Kaufmann, 2004

[10] W. Ogryczak, M. Pióro, and A. Tomaszewski, "Telecommunications network design and max-min optimization problem", Journal of Telecommunications and Information Technology, vol. 3, 2005

[11] D. Nace and M. Pióro, "Max-Min Fairness And Its Applications to Routing and Load-Balancing in Communication Networks: A Tutorial", IEEE Surveys and Tutorials, vol. 10, no. 4, 2008

[12] B. Radunovic and J.-Y. L. Boudec, "A unified framework for max-min and min-max fairness with applications", ACM/IEEE Transactions on Networking, vol. 15, no. 5, October 2007

[13] M. Dzida, M. Pióro, and M. Zagożdżon, "The Application of Max-Min Fairness Rule to Bandwidth Allocation in Telecommunication Networks", The 3rd Polish-German Teletraffic Symposium (PGTS), Dresden, 2004

[14] M. Pióro, M. Dzida, E. Kubilinskas, P. Nilsson, W. Ogryczak, A. Tomaszewski and M. Zagożdżon, "Applications of the Max-Min Fairness Principle in Telecommunication Network Design", Next Generation Internet Networks (NGI 05), IEEE Xplore, Rome, Italy, 2005

[15] W. Ogryczak, M. Milewski and A. Wierzbicki, "Fair and Effcient Bandwidth Allocation with the Reference Point Methodology", International Network Optimization Conference (INOC), Spa, Belgium, 2007

[16] W. Ogryczak and T. Śliwiński, "On Solving Linear Programs with the Ordered Weighted Averaging Objective", European Journal of Operational Research, Vol. 148, pp. 80-91, 2003

[17] M. Resende and C. Ribeiro, "GRASP with Path-Relinking: Recent Advances and Applications" in "Metaheuristics: Progress as Real Problem Solvers", T. Ibaraki, K. Nonobe and M. Yagiura, (Eds.), Springer, pp. 29-63, 2005

[18] J. Puchinger and G.R. Raidl, "Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification" in "Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach", J. Mira and J. R. Álvarez, (Eds.), Lecture Notes in Computer Science 3562, Springer, pp. 41-53, 2005

[19] E. Danna and C.L. Pape, "Branch-and-Price Heuristics: A Case Study on the Vehicle Routing Problem with Time Windows" in "Column Generation", G. Desaulniers, J. Descrosiers and M.M. Solomon (eds.), Springer, New York, 2005

[20] F. Alvelos and J.M. Valério de Carvalho, "A local search heuristic based on column generation applied to the binary multicommodity flow problem", International Network Optimization Conference (INOC), Spa, Belgium, 2007

APPENDIX

Consider a graph $G(N, A)$ where each link $\{i, j\} \in A$ has an associated cost $\mu_{\{ij\}}$. Consider $\boldsymbol{\mu} = \{\mu_{\{ij\}} : \{i, j\} \in A\}$ and let $[\boldsymbol{\mu}]$ be the vector obtained from $\boldsymbol{\mu}$ by rearranging its elements in non-increasing order ($[\boldsymbol{\mu}]_l$ is the l^{th} element of $[\boldsymbol{\mu}]$). For a given $l = 1, ..., |A| - 1$ such that $[\boldsymbol{\mu}]_l > [\boldsymbol{\mu}]_{l+1}$, if we set a value $\alpha \geq 1$ such that:

$$([\boldsymbol{\mu}]_l)^\alpha > \sum_{t=l+1}^{|A|} ([\boldsymbol{\mu}]_t)^\alpha \tag{A.1}$$

then, a standard minimum cost path algorithm (e.g., Dijkstra algorithm), with costs given by $\left(\mu_{\{ij\}}\right)^\alpha$, guarantees that the link with cost $[\boldsymbol{\mu}]_l$ will not be in the solution if there is a path composed only by links with lower costs. An alternative way of ensuring (A.1) is:

$$([\boldsymbol{\mu}]_l)^\alpha > (|A| - l)\,([\boldsymbol{\mu}]_{l+1})^\alpha \tag{A.2}$$

since the right-hand side expression of (A.2) is an upper bound for the right-hand side expression of (A.1).

Note that (A.2) is always true for $l = |A| - 1$. Therefore, if we compute a value of α compliant with (A.2) for all $l = 1, ..., |A| - 2$ such that $[\boldsymbol{\mu}]_l > [\boldsymbol{\mu}]_{l+1}$, we guarantee that the solution of a standard minimum cost path algorithm, with costs given by $\left(\mu_{\{ij\}}\right)^\alpha$, is a solution for the min-max optimization of link costs.

In the greedy procedure of Algorithms B1 and B2 (section III), we start by setting $\alpha = 1.0$ (line 7) and for all $l = 1, ..., |A| - 2$ (line 8) such that $[\boldsymbol{\mu}]_l > [\boldsymbol{\mu}]_{l+1}$ (line 9), we check if (A.2) is fulfilled (line 10) and, if not, we raise α (line 11) to an appropriate value.

If there are indexes $l = 1, ..., |A|-2$ such that $[\boldsymbol{\mu}]_l = [\boldsymbol{\mu}]_{l+1}$, the min-max optimization solution is still guaranteed by any value α compliant with (A.2). To understand this, consider that there are M links with the same cost. In this case, m of such links ($2 \leq m \leq M$) will not be in the solution if there is a path composed by at most $m - 1$ of such links and links with lower costs.