Efficient Routing Heuristics for Internet Traffic Engineering

Ronaldo M. Salles *, Vitor G. Rolla 1
Department of Systems Engineering, Military Institute of Engineering, 22290-270 Rio de Janeiro, Brazil
Received 22 August 2006; received in revised form 8 February 2007; accepted 12 February 2007
Available online 17 February 2007

Abstract

This work presents an alternative analysis for the shortest path optimal routing problem and proposes novel heuristics methods to solve it. In practical terms, a solution for the shortest path optimal routing problem determines the link weights that optimize a computer network operating under standard routing protocols (e.g. OSPF). We base our solution on traffic engineering techniques that respect the shortest path routing model. We consider the main objectives of traffic engineering, such as: load balancing, efficient use of available resources and capacity to support growing traffic demands. Our proposals present better results than traditional approaches and follow closely theoretical optimal points.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Traffic Engineering; Routing; Optimization; Heuristics; OSPF

1. Introduction

The potential growth of Internet traffic demands given by high speed access technologies, the deployment of QoS architectures and protocols, and the necessity to increase network reliability in order to attend competitive markets have pushed service providers to take special care about the management and control of network traffic. Internet Traffic Engineering has emerged as an effort to investigate mechanisms, policies, measurement procedures and operations to tackle such issues [1]. One of the principal goals of Internet Traffic Engineering is to optimize the routing function, which is the focus of this work.

The Open Shortest Path First (OSPF) [2] is one of the most widely used routing protocol in the Internet today due to its reliability, scalability and robustness. The OSPF operates by exchanging link state information among network nodes and computing shortest paths through Dijkstra’s procedures. From link state information, each node in the OSPF domain maintains a database that represents the topology of the network. Applying shortest path algorithms in the topology each node constructs a shortest path tree to all destinations in the domain, where the weight of each link is configured by the network operator. For a good network performance, router manufacturers suggest that each link weight should be set proportionally to \(1/C\), where \(C\) is the capacity of the given link [3].

One of the main limitations of the OSPF protocol is the use of destination based paradigm – the source of traffic does not have control and can not affect the procedure of path selection. Routing decisions are made on each router along the path. In case a path is overloaded, other paths that are not shortest will not be used even if they are underutilized. Given the apparent limitations of the OSPF protocol, most of the Internet Traffic Engineering solutions are commonly implemented using MPLS technology [4].

MPLS technology has been considered as the main tool to meet the requirements for Internet Traffic Engineering [5]. In general terms, MPLS provides a connection oriented sub-layer to IP networks, which facilitates the establishment of alternative paths and the support of constraint-based routing. Despite all the advantages gained with MPLS, the majority of service providers still operates...
under standard IGP routing protocols given their large operational experience.

Fortz and Thorup in [6] studied the behavior of standard routing procedures in the implementation of traffic engineering functions. Their results show that standard routing procedures may achieve equivalent performance when compared to MPLS solutions if the algorithms operate using pre-computed network link weights. Such conclusion backs up recent efforts to provide traffic engineering capabilities to the OSPF protocol [7].

The determination of optimal link weights involves the solution of the shortest path optimal routing problem. Since this problem is NP-complete [8], heuristics solutions are envisaged. In this sense, the central issue of this work is the investigation of alternative solutions for the shortest path optimal routing problem so that standard routing protocols (e.g. OSPF) could be efficiently employed to support internet traffic engineering operations.

We propose three routing heuristics (HeurCoupling, HeurRosa and HeurOne – please refer to Section 4) to return network link weights, and evaluate the heuristics according to: (i) relative distance to the optimal routing solution with bifurcation (most general case), (ii) improvement over router manufacturers recommendation (weight set to 1/C), and (iii) running time. We also compare the performance of our proposals with other efficient heuristics found in the literature.

This paper is organized as follows. Section 2 reviews routing optimization problems and formulation. Section 3 comments about related work on the topic. Section 4 presents our proposed algorithms, which are then evaluated through numerical results in Section 5. Section 6 discusses some important aspects about our work. Finally, in Section 7 the paper is concluded and some points to future work are highlighted.

2. Routing optimization problems

2.1. The Optimal Routing Problem (ORP)

The general Optimal Routing Problem (ORP) consists of distributing the traffic demand of each origin node to destination node (OD pair) using all the available network paths so that a certain global function is minimized. It is also known as optimal routing problem with bifurcation. Several works in the literature refer to this problem, it follows next the formulation as found in [9]

\[
\min \sum_{(i,j)} D_{ij}(F_{ij}) 
\]  

such that

\[
F_{ij} = \sum_{p \neq (i,j)} x_p 
\]

\[
\sum_{p \in P_w} x_p = r_w \quad \forall w \in W 
\]

\[
x_p \geq 0 \quad \forall p \in P_w, \quad w \in W
\]

where \( D_{ij}(F_{ij}) \) is the cost function associated to link \((i,j)\), \( F_{ij} \) is the overall flow that goes through link \((i,j)\), \( p \) is a given end-to-end path, \( x_p \) is the flow on path \( p \), \( W \) the set of all OD pairs, \( P_w \) the set of all network paths connecting OD pair \( w \) and \( r_w \) the demand of OD pair \( w \). Each term of the summation in (1) is usually given by

\[
D_{ij}(F_{ij}) = \frac{F_{ij}}{C_{ij} - F_{ij}} 
\]

where \( C_{ij} \) is the capacity of link \((i,j)\).

Eq. (5) acts as a barrier function for link \((i,j)\) – as the flow \( F_{ij} \) gets close to \( C_{ij}, D_{ij} \) grows without bound. By minimizing this function traffic is shifted away from overloaded links. As a consequence, flows are better spread along the network while preserving the same end-to-end demands \( r_w \). Note that the barrier function has also a physical interpretation, it represents the mean number of packets in each link if modeled as a M/M/1 system. Hence, another way to see the objective of problem (1) is to distribute network loads so that the mean number of packets in the network is minimized. The optimal routing problem was the subject of several previous works and the most well known solutions can be found in [10] (flow deviation method) and [9] (gradient projection method).

2.2. Single Path Optimal Routing (Single-ORP)

The Single Path Optimal Routing problem (Single-ORP) is the non-bifurcation version of the optimal routing problem studied in the last subsection. For each OD pair a single path is selected out of all the available alternatives. The problem is to choose a combination of paths (one for each OD pair) that minimizes the overall cost function. Using the formulation introduced before constraint (3) is now replaced by

\[
\sum_{p \in P_w} \delta_p x_p = r_w \quad \forall w \in W 
\]

\[
\sum_{p \in P_w} \delta_p = 1, \quad \delta_p = \{0,1\}
\]

where \( \delta_p \) indicates if path \( p \) is selected (=1) or not (=0).

The Single-ORP is a combinatorial optimization problem, which is NP-hard [11]. Given its high complexity the solution to this problem is usually obtained through the employment of some appropriate heuristics. It is important to notice that in the Single-ORP solution all the demand \( r_w \) of a given OD pair \( w \) must be carried over a single path, depending on the values of \( r_w \) feasibility may be a problem.

2.3. Shortest Path Optimal Routing Problem (Short-ORP)

The objective of the Shortest Path Optimal Routing Problem (Short-ORP) is to find weights to network links that allow routing procedures (e.g. OSPF) to minimize a certain cost function. The Short-ORP differs from previous formulations, here the unknowns are link weights. When weights are set to their optimal values, shortest path
The optimization of OSPF link weights for traffic engineering has received great interest lately. It has been proved that an optimal solution for any traffic engineering problem can always be converted to a set of shortest-paths with respect to some link weights [13].

Fortz and Thorup presented one of the first works on link weight optimization for traffic engineering [8] using a local search technique. Numerical results obtained for some special cases show that the solution proposed gets close to ORP points. The authors have also investigated solutions for this problem in [6,14].

In [15] the authors propose a combinatorial algorithm to solve Short-ORP. The heuristics proposed is based on the analysis of path coupling, a condition that must be enforced in shortest path routing algorithms (our first heuristics also tackle this problem as it will be shown in the next section). Their paper provides a thorough taxonomy about routing problems and characterizes Short-ORP as a special case of the inverse shortest path [16].

Another approach to solve Short-ORP can be found in [12]. The authors compared their deterministic approach to other proposals based on traditional meta-heuristics: Simulated Annealing and Simulated Allocation, studied in [17]. Their results show an improvement of 5–10% over such meta-heuristics. Some other related proposals based on meta-heuristics are also available. Genetic algorithms were used in [18–20].

More recently, two interesting papers also tackled problems related to OSPF and Traffic Engineering (TE). In [24] the authors investigated basic requirements for practical TE over OSPF networks. They formulated an integer linear programming problem together with its dual, and used a different cost function that maximizes sessions throughput. One of the main advantages of their approach is that OSPF TE can be done without the knowledge of accurate traffic matrices, which is often difficult to obtain in practice. The same authors in [25] studied another important issue, which is how to restrict the number of superfluous shortest paths generated by common OSPF TE schemes. The paper proposes an algorithm to return link weights that provide a minimal shortest path representation (mSPR) of the designated TE paths.

4. Optimizing link weights: proposed algorithms

This section proposes three different heuristics to solve Short-ORP, each heuristics uses a particular strategy to find a solution. Numerical results are left to the next section where comments about the recommended use of each heuristics are made.

The proposed algorithms are based only on deterministic operations, we avoid employing probabilistic procedures such as simulated annealing, genetic algorithms, and other evolutionary approaches. We believe that it is possible to obtain efficient algorithms by exploiting the structure of the problem and without having to rely on generic meta-heuristics.
4.1. HeurCoupling

The first heuristics proposed, HeurCoupling, is based on a two-step approach. In the first step, a simple algorithm is used to find an approximated solution for the Single-ORP. The additional requirement for the algorithm is that it returns paths that satisfy the coupling condition: any computed path that has two nodes in common with previous paths must use the same sub-path between these two nodes since IGP protocols work on single shortest paths.

In the second step a linear programming problem is employed to determine the weights that should be used to secure the solution obtained in the first step. In this way the weights returned by HeurCoupling serve as a solution for Short-ORP, or in other words, they serve for optimizing the performance of traditional routing protocols such as the OSPF.

4.1.1. HeurCoupling first step: Finding Paths

For this step we propose a greedy heuristics that routes demands one by one from the largest to the smallest. We use the available bandwidth \( A_l = C_l - F_l \) to represent the cost on each link, i.e. the cost on link \( l \) is \( c_l = 1/A_l \), and route demands through shortest cost paths. The rationale behind this approach is that large demands will be routed over a less congested network and using links with large amounts of available resources (low costs \( c_l \)). This procedure attempts to reduce the impact of large demands on the barrier functions (5), which compounds the overall optimal routing cost (1).

The first step of HeurCoupling algorithm is presented next, where: \( W \) is the set of \( r_w \) demands to be routed, \( L \) is the set of network links, \( C_l \) the capacity of link \( l \), \( A_l \) the available bandwidth, and \( c_l \) the cost.

HeurCoupling: Finding Paths
0. for all \( l \in L \): \( c_l \leftarrow 1/C_l \)
1. find \( r_j \in W \) such that \( r_j \) is maximum
2. route \( r_j \) through shortest path \( p_j \) (Dijkstra’s)
3. if \( p_j \) is coupled goto 4.
   else recompute \( p_j \)
4. update all link costs: \( c_l \leftarrow 1/A_l \)
5. \( W \leftarrow W - \{r_j\} \)
6. if \( W = 0 \); END
   else goto 1.

In 0 the algorithm initially starts all link costs to \( 1/C_l \). Then in 1 and 2 the largest demand \( r_j \) in \( W \) is selected and the corresponding shortest cost path \( p_j \) is determined. Path \( p_j \) may be changed in 3 to be coupled with previous determined paths. If there is a previous path \( p_i \) which has at least two nodes in common with \( p_j \), the corresponding sub-path of \( p_j \) is changed to match the same sub-path of \( p_i \). After that all paths are coupled and the algorithm may continue. In 4 costs are updated to reflect demands that have already been routed and consumed link resources. Then in 5 set \( W \) is also updated and the procedure ends in 6 if there is no more demands to be routed, else it goes back to 1.

At the end of the algorithm we have a set of end-to-end paths to route all demands so that the overall routing cost is reduced. The solution is never worse than router’s manufactures recommended approach \((1/C)\) and in practice produces results which are close to the optimal as it will be seen in the numerical results section. However, to use traditional IGP routing protocols it is still necessary to determine the corresponding link weights.

4.1.2. HeurCoupling Second Step: Determining Weights

The second step of HeurCoupling consists of determining weights to all network links in order to secure the solution returned in the first step. Hence, standard routing protocols using such weights on network links would route demands through paths \( p_j \) obtained in the previous step.

One way to tackle this problem is to formulate it as a linear programming problem. Link weights are variables and constraints are such that for each path \( p_j \) returned from the first step, the sum of the weights along \( p_j \) should be made less than the sum of the weights along any other alternative path available for demand \( r_j \). Let us use the example in [21] to illustrate this procedure.

Fig. 2 sketches a 7-node network along with the respective weights on links. Assume path A-D-G was returned from the first step of HeurCoupling, hence in order to guarantee that it will be used by standard shortest path routing protocols we must have the following relations:

\[
\begin{align*}
(1) \quad w_{ad} + w_{dg} & < w_{ab} + w_{be} + w_{eg} \\
(2) \quad w_{ad} + w_{dg} & < w_{ac} + w_{cg} \\
(3) \quad w_{ad} + w_{dg} & < w_{ab} + w_{bd} + w_{dt} \\
(4) \quad w_{ad} + w_{dg} & < w_{ab} + w_{bd} + w_{de} + w_{eg} \\
(5) \quad w_{ad} + w_{dg} & < w_{ad} + w_{cf} + w_{fg} \\
(6) \quad w_{ad} + w_{dg} & < w_{ad} + w_{de} + w_{eg}
\end{align*}
\]

Fig. 2. 7-Node network.
Therefore, constructing similar constraints for each path and using a cost function such as \[ \sum_{(i,j)} w_{ij}, \] it is possible to obtain the weights from the solution of a LP problem.

The problem with this approach is the size of the LP that has to be solved given the high number of constraints in the formulation. However, in [21] the authors studied the inverse shortest path problem (which is the problem tackled in this step) and proposed a simple and efficient criterion to reduce the number of constraints. In this way it is possible to solve large instances of the problem.

The criterion can be explained as follows. Assume \( p_{od} \) the selected path between nodes O and D, and \( V(p_{od}) \) the overall weight along the path. Let \( p_{ai}p_{id} \) be an alternative path that pass through intermediate node I, thus \( V(p_{ai}p_{id}) \) is the cost of the alternative path. The criterion states that if \( V(p_{ai}p_{id}) < V(p_{ai}p_{id}) \) is the cost of the alternative path. The criterion states that the criterion can be explained as follows. Assume \( p_{od} \) the selected path between nodes O and D, and \( V(p_{od}) \) the overall weight along the path. Let \( p_{ai}p_{id} \) be an alternative path that pass through intermediate node I, thus \( V(p_{ai}p_{id}) \) is the cost of the alternative path. The criterion states that if constraint \( V(p_{ai}p_{id}) < V(p_{ai}p_{id}) \) is redundant if at least one of the following is true: (i) \( p_{ai} \) is not the selected path between nodes O and I, (ii) \( p_{id} \) is not the selected path between nodes I and D.

By adopting this simple criterion the number of constraints in the problem is now polynomial. For the example in Fig. 2 it can be observed that constraints from (3)-(6) are redundant if the chosen paths between directly connected nodes is the link between them, and the shortest path between B and G is B–E–G. We adopted this criterion on our calculations.

### 4.2. HeuRoSa

HeuRoSa differs from HeurCoupling since it works directly with link weights. The basic idea behind HeuRoSa is to change a given link weight at each iteration so that this change will cause the re-routing of a single demand OD. The demand will effectively be routed if this action decreases the overall cost function. By using such deterministic approach we have a greater control over the evolution of the algorithm.

In order to reroute a single demand at each algorithm iteration it is necessary to first compute \( \delta \). This parameter represents the value \( \delta_{sup} (\delta_{inf}) \) that must be added (subtracted) to the weight of a given link so that one demand \( w \) that uses (does not use) this link is rerouted. The highest (lowest) utilized link is used to compute \( \delta_{sup} (\delta_{inf}) \).

The rationale behind this approach is the following: by increasing the weight of a congested link, demands will tend not to use this link and the overall cost will strongly reduce since variations on the cost function due to congested links are large. On the other hand, by decreasing the weight of an underutilized link demands will be attracted to it, however the increase in the cost function due to this link is often smaller than due to other more congested links. The reason to reroute one demand at a time by computing the corresponding \( \delta \) is to avoid oscillations, which may be experienced if weights are varied abruptly and altogether.

The algorithm operates with two distinct phases. The first phase works with \( \delta_{sup} \) by increasing the weights of overloaded links. The second phase works with \( \delta_{inf} \) by decreasing the weights of underutilized links. The algorithm finishes the first phase and starts the second phase only when it is not possible to improve the cost function anymore. The whole procedure terminates when both first and second phases cannot improve cost. By using the two phases in a row we decrease chances of being stalled too early in the procedure and very frequently achieve better results than using a single phase.

The first step of the algorithm is to configure link weights to \( \frac{1}{n} \), which is the router manufacture’s recommendation. Hence, we start with an initial cost \( \sum_{(i,j)} D_{ij}(F_{ij}) \) obtained from the \( \frac{1}{n} \) settings and our algorithm never returns a worse solution than this one.

It follows below the pseudo code for the \( \delta_{sup} \) phase, the second phase (\( \delta_{inf} \)) is analogous and also commented next.

**HeuRoSa:**

1. **\( \delta_{sup} \) Phase**
   0. for all \( l \in L \), \( w_l \leftarrow \frac{1}{n} \).
   1. find \( \hat{l} \in L \) such that link utilization is maximum
   2. compute \( \delta_{sup} \) for link \( \hat{l} \)
   3. \( w_{\hat{l}} \leftarrow w_{\hat{l}} + \delta_{sup} \)
   4. compute newcost
   5. if newcost < cost:
      1. cost \( \leftarrow \) newcost
      goto 1
   else
      \( L \leftarrow L \setminus \{\hat{l}\} \)
      \( w_{\hat{l}} \leftarrow w_{\hat{l}} - \delta_{sup} \)
   6. if \( \hat{L} = \emptyset \); END
   else goto 1

In step 0 network link weights are initially set to \( 1/C \), then in step 1 the link of maximum utilization \( \hat{l} \) is selected to have its weight updated. Step 2 computes \( \delta_{sup} \) for this link and the corresponding weight is updated in step 3. Step 4 computes the new cost \( \sum_{(i,j)} D_{ij}(F_{ij}) \) after rerouting due to change in link weight \( w_{\hat{l}} \). A test is carried out in step 5: if cost is decreased the algorithm goes back to step 1 and the procedure is repeated, otherwise this link will be no more considered in future steps and the corresponding weight returns to its previous value (no progress achieved with this change). Step 6 tests for termination, if there is no more links to try the procedure ends, otherwise the procedure is repeated from step 1.

The computation of \( \delta_{sup} \) in step 2 is done in the following way. First, for each demand that uses link \( \hat{l} \) (candiates to be rerouted) we calculate the cost of the shortest path obtained if link \( \hat{l} \) is not available in the network. Then, by subtracting this cost to the cost of the original path we have the difference in cost \( \Delta \) for each demand. The parameter \( \delta_{sup} \) should be set above the minimum among all \( \Delta \)'s and below the second minimum in order to reroute just one demand. If the minimum and second minimum are the same two demands will be rerouted after link weight update.

For the \( \delta_{inf} \) phase we have a similar procedure. First, we search for the least utilized link \( l \), then we compute the
corresponding $\delta_{ij}$ and try the new weight: $w_j \leftarrow w_j - \delta_{ij}$. At this time one demand has been re-routed and now crosses link $l$. If a reduction in cost is observed we keep this solution and repeat the procedure, otherwise we go back to the previous state and search for the second least utilized link until all links are tested.

4.3. HeurOne

This approach also works directly with link weights as HeuRoSa. Each iteration of the algorithm solves an one-variable optimization problem (OneORP): the only variable is the weight of the most congested link, all other weights are fixed to their current values. The iteration problem is then to find the weight for this link that minimizes the cost function $\sum_{(i,j)} D_{ij}(F_{ij})$. If cost is improved the weight is set to the optimal value and the procedure is repeated in the next iteration, otherwise the algorithm searches for the second most congested link until cost is reduced or all links are tested. It follows below the pseudo code for HeurOne.

HeurOne

0. for all $l \in L : w_l \leftarrow \frac{1}{C_l}$
1. compute routing cost
2. find $l \in L$ such that utilization is maximum $w \leftarrow w_l$
3. solve OneORP for $\hat{l}$ (return new $w_l$)
4. compute newcost
5. if newcost $<$ cost
   $\begin{cases} 
   cost \leftarrow \text{newcost} \quad \text{goto 2} \\
   \text{else} 
   \quad \text{w} \leftarrow \text{w}_l 
   \quad L \leftarrow L \setminus \{\hat{l}\}
   \end{cases}$
6. if $L = \emptyset$: END 
   \text{else goto 2}

The procedure is simple and similar to what has been done before. The only different part is given in step 3 where OneORP is solved to return the optimal weight for the most congested link. OneORP can be formulated as a problem of this single variable:

$$
\min D(x) = \sum_{(i,j)} D_{ij}(F_{ij}(x)), \quad x \geq 0 
$$

(9)

where $F_{ij}(x)$ represents the overall flow on link $(i,j)$ as a function of the weight on link $\hat{l}$ (most congested) with all other weights fixed.

To illustrate the approach let us use the 8-node network topology in Fig. 1 with same demands and link capacities. OneORP is solved to return the optimal weight for the most congested link. OneORP results were obtained using KNITRO from neos server [22] a mathematical programming tool for solving large scale optimization problems. HeurCoupling used lp-solve [23] to carry out the second step, while all other algorithms were coded using standard programming language.

The most congested link is $(4,8)$ since utilization is maximum 10/11, and so the weight of this link is selected as variable for OneORP. If $w_{48} = 0$ all flows in (10) will be preserved and overall cost is $D(0) = 11.66$. Actually, for $0 \leq w_{48} \leq 9$ we have $D(w_{48}) = 11.66$ since there will be no change in routing and link flows are preserved. When $w_{48}$ is just above 9, the shortest path for demand $r_1$ will now be $1 \rightarrow 3 \rightarrow 6 \rightarrow 8$, link flows will change and the respective cost is 4.166. In summary:

$$
D(w_{48}) = \begin{cases} 
11.66, \quad 0 \leq w_{48} < 9 \\
4.166, \quad 9 < w_{48} \leq 12 \\
5, \quad 12 < w_{48}
\end{cases} 
$$

(11)

Therefore, to minimize $D(w_{48})$ we must have $9 < w_{48} \leq 12$, for instance $w_{48} = 24/12 = 11.5$. Note that on the boundaries when $w_{48} = 9$ or $w_{48} = 12$ there are two equal cost shortest paths, however we only consider rerouting if there is a strict shortest path.

Step 3 of HeurOne procedure does exactly what has been illustrated above. For each demand $w$ there are only two possibilities depending on the value of $x$: either be routed over a path that does not include link $(p_u)$ or over a path that uses $(p_u)$. Hence, for each demand it is possible to determine the values of $x$ for $p_u$ and $p_w$. By combining the values for all demands we determine the intervals of $x$, the routing setting for each interval, and consequently the cost as in (11).

The algorithm terminates when it is not possible to improve cost anymore and all links were already tested.

5. Numerical results

5.1. Performance evaluation

In this subsection we carry out a performance analysis of the proposed algorithms according to the following performance parameters: cost $\sum_{(i,j)} D_{ij}(F_{ij})$, maximum network utilization, and running time. We compare the performance achieved by our approaches with the performance of three well-known routing schemes: minimum hop count routing (NrHops) where all weights are fixed and unitary, standard 1/C procedure recommended by manufacturers, and also with the absolute optimal points returned by the general optimal routing problem (ORP), NrHops, 1/C and ORP solutions serve as reference points (benchmarks) for our assessment and give us a common ground to relate the performance of our algorithms with any other proposed routing scheme.

ORP results were obtained using KNITRO from neos server [22] a mathematical programming tool for solving large scale optimization problems. HeurCoupling used lp-solve [23] to carry out the second step, while all other algorithms were coded using standard programming language.

We used three network topologies with different number of nodes and links. Each topology was submitted to ten
different scenarios of increasing traffic loads. Scenario 0 was subjected to very light loads such that there are not almost any difference on the performance among all routing schemes. From this scenario on we increased loads gradually, by multiplying each demand by a random number between 1 and 1.2, until Scenario 9 where loads are high and even ORP results were not good.

Figs. 3, 5 and 7 show the overall cost $\sum_{ij} D_{ij}(F_{ij})$ while Figs. 4, 6 and 8 show the maximum link utilization in the network for the three network topologies. Running times are presented in Tables 1 and 2.

Regarding the overall cost (Figs. 3, 5, and 7), it can be seen that for Scenarios 1–4, which represent light loads, the performance of all algorithms were similar. The only exception happened for NrHops in Fig. 5 where just after Scenario 1 costs are very high. In all cases, from Scenario 5 onwards the results for the recommendation $1/C$ and NrHops exacerbate, notably for high loads the overall cost grows unbounded.

Another important point to observe is that HeurCoupling, HeuRosa and HeurOne results were close from the absolute optimal points (ORP), and hence presented a much better performance than the $1/C$ recommendation.

For the 16-node topology routing costs are presented in Fig. 3. HeurOne results are in average 6% worse than optimal (ORP) and 603% better than $1/C$. HeuRoSa and HeurCoupling results are in average 14% and 18% worse than optimal and 556% and 531% better than $1/C$ respectively. For this case, HeurOne got the best performance and HeuRoSa was slightly superior to HeurCoupling. Regarding maximum utilization in Fig. 4, again HeurOne presented the best results being very close to ORP. HeurOne is in average 1% worse than ORP and 16% better than $1/C$, while HeuRoSa and HeurCoupling are approximately 6% worse than ORP and 10% better than $1/C$.

Routing costs for the 26-node topology are presented in Fig. 5. It can be seen a very poor performance of NrHops even in light load conditions (just after scenario 1). Costs presented for the $1/C$ heuristics exacerbate after scenario 5 and again the proposed heuristics followed ORP results closely. HeurCoupling and HeurOne performed in average

Table 1

<table>
<thead>
<tr>
<th>ORP</th>
<th>HeuRoSa</th>
<th>HeurOne</th>
<th>HeurCoupling</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-Node topology</td>
<td>56.067</td>
<td>0.260</td>
<td>14.601</td>
</tr>
<tr>
<td>26-Node topology</td>
<td>212.000</td>
<td>4.545</td>
<td>204.034</td>
</tr>
<tr>
<td>30-Node topology</td>
<td>1842.006</td>
<td>11.988</td>
<td>1422.445</td>
</tr>
</tbody>
</table>

Table 2

<table>
<thead>
<tr>
<th>HeuRoSa (%)</th>
<th>HeurOne (%)</th>
<th>HeurCoupling (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-Node topology</td>
<td>0.4</td>
<td>26</td>
</tr>
<tr>
<td>26-Node topology</td>
<td>2.1</td>
<td>96</td>
</tr>
<tr>
<td>30-Node topology</td>
<td>0.7</td>
<td>77</td>
</tr>
</tbody>
</table>
8% worse than ORP and and 196% better than 1/C. While HeuRoSa results were 16% and 175% respectively. Fig. 6 shows maximum link utilization results. HeurOne, HeuRosa and HeurCoupling were in average 26%, 33% and 31% above ORP and 21%, 15% and 17% below 1/C.

Finally, routing costs and maximum link utilization for the 30-node topology are shown in Figs. 7 and 8. It can be observed a similar behavior as for the previous topologies. However, in this case HeurOne, HeuRosa and HeurCoupling results were even closer to each other. The average figures for cost are 7% above ORP and 340% below 1/C.

Table 1 below presents the mean running time in seconds to return ORP, HeuRoSa, HeurOne and HeurCoupling results, while Table 2 shows the percentage of processing time in relation to ORP. Times are in seconds and all algorithms run over a Pentium III 1 GHz system.

It can be seen that HeurOne processing time for all topologies is comparable to ORP. For the 26-node topology it is almost the same (96%) saving only 4% of processing time. On the other hand, HeuRoSa and HeurCoupling run much faster than ORP with worst result providing a considerable save of 98% of processing time.

5.2. Comparison with other heuristics

In this subsection we carry out a performance comparison between our proposed algorithms and the heuristics proposed in [28] (GAOSPF) and [8] (F-T). We collected all data and configuration files related to the results presented in the first paper, and then run our algorithms for those scenarios. In other words, all results presented here are for the networks and traffic matrices used in [28].

Figs. 9 and 10 present the overall cost and maximum link utilization for a random network with 50 nodes and 245 arcs, while Figs. 11 and 12 present the results obtained for a hierarchical network with 50 nodes and 148 arcs. As mentioned above, the networks as well as GAOSPF and F-T results were taken directly from [28]. The graphs also show the results we obtained for: router manufacturers recommendation heuristics (1/C), minimum hop count
routing (NrHops), optimal routing (ORP-Delay), our first heuristics (HeurCoupling, Section 4.1), second heuristics (HeuRosa, Section 4.2), and third heuristics (HeurOne, Section 4.3).

From Fig. 9 it can be seen that HeurCoupling and HeuRosa results were very close to GAOSPF, while HeurOne followed F-T points. For all scenarios HeurOne outperformed GAOSPF and was only worse than F-T for high loads. For the maximum link utilization presented in Fig. 10 the situation was similar, HeurCoupling and HeuRosa very close to GAOSPF while HeurOne was better than GAOSPF and close to F-T points.

For the hierarchical network results presented in Figs. 10 and 11, the behavior of the algorithms changed a little. Now, HeurCoupling and HeuRosa curves were no more close to the other heuristics, while GAOSPF, F-T and HeurOne results were almost indistinguishable.

From all the results presented in this section we may conclude that HeurOne is a very efficient heuristics since it outperformed GAOSPF in most of the cases and followed closely F-T results. Although HeurCoupling and HeuRosa did not present good results for the hierarchical network, they performed well for the random network (Figs. 9 and 10).

Moreover, our proposed heuristics were based on simple concepts, which facilitates understanding and implementation. In this section we run all our algorithms on a standard Pentium IV PC, which may also motivates online implementations in other non-expensive platforms.

6. Discussion

The heuristics proposed in this work have as their main objective the minimization of the cost function as given by (1) and (5). Our choice for this cost function was motivated by the Optimal Routing Problem (ORP) [9], which is often used as a benchmark for TE problems. Moreover, this cost function is commonly employed in TE approaches and algorithms as it can be seen in most of the references cited in this work. However, we would like to highlight that the proposed heuristics are general and can be applied to other cost functions of interest.

The use of barrier functions as in (5) has its own advantages. It has long been known that packet networks exhibit a threshold behavior [10]: packet delays on links are reasonably constant, but when throughput approaches link capacity delays exacerbate and grow without bound. The cost function used in this work captures this behavior quite well and avoids maximization of throughput on links, which may cause long delays and high packet drops. Recently, other different cost functions have been proposed for Internet TE, an interesting work on this topic can be found in [24].

Another important aspect to discuss is about the correlation with packets inter-arrival times. It has been shown that both local and wide area network traffic presents long-range dependence and exhibits self-similar behavior.
In principle, our approach does not directly handle such correlations given the formulation presented here. As mentioned in Section 2.1, the cost function adopted in this work represents the mean number of packets if each link is modeled as a M/M/1 system.

In [27] an interesting illustration for the self-similar storage model can be found in Fig. 9.7. It can be seen that the M/M/1 model underestimates queue sizes for self-similar traffic. If we have beforehand an estimate for the hurst parameter \( H \) used to describes the network traffic under study, we may minimize such discrepancies by reescalating our problem. For instance, we may artificially reduce link capacities or increase demands by a factor to compensate the underestimation of our model. Although this solution is not very accurate, it keeps the simplicity required by TE solutions and avoid the complexity involving self-similar problem formulations. Therefore, the proposed heuristics can also be applied in this case.

Finally, we would like to mention about how our work could be applied without the knowledge of precise input traffic matrices. If traffic matrices are not available and/or cannot be measured, we need at least an upper bound estimation of these values. The estimated upper bound traffic matrix can then be used as input for our heuristics. Hence, the returned solution represents an approximation for the optimal weight settings. The advantage on using upper bound estimation is that link weights will be configured for the worst scenario, which is desirable from a TE viewpoint: in most cases the weights will not be optimal but the network will be better conditioned for sudden increase on traffic demands.

7. Conclusion

In this paper we studied optimal routing problems and their applications to the optimization of network resource utilization. We focused on the shortest path optimal routing problem and proposed three different heuristics HeurCoupling, HeuRoSa and HeurOne to return solutions that could be implemented in the network using standard routing algorithms. The proposed heuristics could be used by traffic engineering procedures in order to manage network resources and better distribute network traffic loads.

Numerical results confirmed a good performance of our heuristics when compared to router manufacturer’s recommendations \((1/C)\), our results were also close to absolute optimal points \((\text{ORP})\). In fact, from the results presented in Section 5.1 there were no significant difference on performance among the proposed heuristics which would make any one of them equally recommended. However, for larger network scenarios as the ones presented in Section 5.2, HeurOne outperformed HeurCoupling and HeuRoSa as well as GAOSPF [28], making it an efficient choice in a large variety of conditions. When it comes to processing time HeurCoupling and HeuRoSa provide huge savings when compared to HeurOne. Thus, if processing time is an issue HeurOne should not be employed.

When comparing HeurCoupling with HeuRoSa there is an important design aspect to be considered. HeurCoupling is based on a linear programming problem so a LP solver is necessary to accomplish Step-2. If such a solver is not available for some reason, HeurCoupling can not be applied.

After all the discussions and results presented in this paper we believe that HeuRoSa is the most readily employable heuristics to return link weights for traffic engineering. It shows that even simple traffic engineering procedures may produce significant gains in network performance. Also, the good results and small computational times observed for HeuRoSa in the experiments motivates further investigations on the use of an adaptive procedure that could respond online to traffic changes.

References


Ronaldo M. Salles received the BSc. degree in Electrical Engineering and the MSc. in Computer Science both from the Military Institute of Engineering in 1992 and 1998, respectively. He received the PhD degree from the Imperial College London, United Kingdom, in 2004. His research interests include QoS and traffic engineering, routing, optimization theory and resource allocation. Currently he holds a position of Associate Professor in the Computer Engineering Department at the Military Institute of Engineering, Rio de Janeiro, Brazil.

Vitor G. Rolla received the Technologist degree in Data Processing from Brasília Educational Union (UNEB), Brasília - Brazil in 2003. He received the MSc. degree in Computer Science from the Military Institute of Engineering (IME), Rio de Janeiro - Brazil in 2006. Currently, he holds a position of consultant in Globosat Telecommunication Company also in Rio de Janeiro, Brazil.