EuroFGI Workshop on IP QoS and Traffic Control
P. Pereira (Ed.)
Lisbon, Portugal, December 6–7, 2007

CONSTRAINED STEINER PROBLEM WITH DIRECTIONAL METRICS

Stefano Secci[1,2], **Jean-Louis Rougier**[1], **Achille Pattavina**[2]

[1] Départment Informatique et Réseaux, ENST Paris, France.
e-mail: rougier@enst.fr, secci@enst.fr

[2] Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy
e-mail: pattavina@elet.polimi.it

Keywords: QoS Routing, directional policies, constrained Steiner problem

Abstract. *There is a growing demand for multipoint multimedia services over IP and an explosion of VPN services. Nowadays, multipoint QoS services need to be extended beyond provider boundaries. This manuscript studies the tree-constrained tree-optimization problem with directional metrics. Weights and constraints are not applied to an arc but to a 3uple* (a, b, c) *of vertices named a directional arc, composed of two simple arcs: one incoming and one outgoing, to and from a given transit node* b*. This cost and routing model is well suited to represent complex Autonomous System (AS) interactions. In this way an AS can apply different policies based on the direction of the flow.*

Although the constrained Steiner problem with directional metrics may be relaxed to its classical form by extending the graph with directional metrics to a larger graph with legacy metrics, this transformation increases the complexity of the problem. We formulate the problem directly and devise an ad-hoc breadth-first search approach with limited depth for its resolution. We analyze the time computational complexity and the optimality of our heuristic. We show that taking advantage of pre-computation, our algorithm reaches solutions very close to optimality and appears to be more competitive, less complex than other well known multipoint routing algorithms.

1 INTRODUCTION

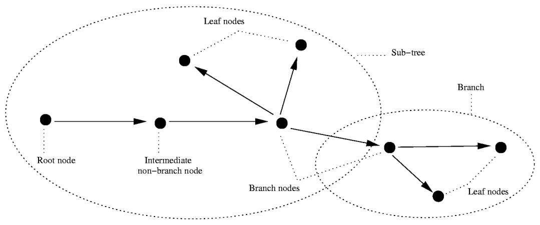
Let a point-to-multipoint tree be a directed tree selected with single source and several destinations over a weighted and directed graph. We differentiate between point-to-multipoint tree and multicast tree since this latter is usually associated with best-effort transmissions. Point-to-multipoint connections are intended to be used for single-source connections with some Quality of Service (QoS) guarantees, for instance using MPLS point-to-multipoint tunnels [1]. From now on instead of "point-to-multipoint" we will use the shorter "multipoint" term; no confusion should arise with multipoint-to-multipoint connections which are not considered in this manuscript (it is still a challenging issue).

When the metrics are applied to arcs, and end-to-end bounds are imposed, the extraction of a constrained tree from a graph is a tree-optimization tree-constrained problem [2], also called constrained Steiner problem [3].

Let a *directional arc* be a chain of two adjacent arcs in a graph. Then, let a directional metric be an *additive metric* applied to a directional arc. In other words, given a node connected to at least two other nodes, the metric is different depending on the incoming and outgoing nodes. This routing behavior appears to be appropriate for policy routing problems such as inter-domain route computation for inter-domain LSPs [4]: an Autonomous System (AS) can thus fix different *per-direction* policies such as transit cost or guaranteed performance to different neighbouring ASs, some of which can be also grouped to apply per-group specific policies. Or, more generally, directional metrics can be useful even for overlay, multi-layer or hierarchical routing. In all these applications, the lower layer paths are computed over the upper-layer route (e.g. an inter-AS LSP over a given AS path).

An agreed taxonomy is needed to unequivocally identify the elements of a multipoint tree:

Root node: source node of a multipoint connection;
Leaf node: node destination of the transmission;
Branch node: node that performs data replication;
Intermediate node: non-branch and non-root node;
Bud node: a leaf-and-branch node.



Multipoint tree elements

Furthermore, a set of nodes can be classified as:
Multipoint sub-tree: part of a tree such that a root or an intermediate node is connected to a subset of leaves;
Multipoint branch: part of a sub-tree such that a single branch is connected to a subset of leaves.

In this paper, we treat the constrained Steiner problem in the case of directional metrics and propose a novel heuristic for its resolution. In Sect.2 we introduce the problem by discussing some heuristics for the classical constrained Steiner problem, explaining how they could be applied to graphs with directional metrics. Then, in Sect.3, we formulate the problem with multiple metrics by linear programming. Our approach is presented in Sect.4, and compared to other heuristics for the case with single metric (apart from the cost) in Sect.5.

2 SOME HEURISTICS

We discuss possible algorithms for the tree-optimization tree-constrained problem.

2.1 Irrespective Route Computation with Post Merging

A simple algorithm is the following: compute the shortest route subject to all constraints for each leaf; join the sub-route parts of the routes sharing arcs.

We refer to this algorithm with the acronym IRC-PM. The resulting tree has sparse branches in non-optimal positions. It is important to remark that resources (e.g. bandwidth) can be shared on common links. Hence it is better to adopt algorithms which reduce tree cost by encouraging arc sharing.

2.2 Iterative Point to Point Selection

Dividing the multipoint problem into multiple point-to-point route selections, the routes tend to share arcs. An alternative algorithm is: compute the shortest route subject to all constraints from the root to a first leaf node; assign null cost to all directional arcs taken by the first route and compute the route to the second leaf; repeat the process for every remaining leaf.

We refer to this algorithm with the acronym I-P2P. An advantage is that it still does not require the knowledge of all leaves during the tree computation, while being sensitive to link sharing with respect to the IRC-PM algorithm. However, the solution (and its optimality) strongly depends on the order in which routes to leaf nodes have been computed.

2.3 Steiner Tree

To avoid the dependency on leaf ordering, it is needed to compute directly the optimal tree that spans all the destinations, i.e., the Steiner tree. Its optimization directly pushes data replication points (branch nodes) as close as possible to termination points (leaf nodes), seeking optimality. The optimization problem is known to be NP-hard [3], and is much more complex in the case of additive constraints. The problem not being tractable for large instances, heuristics are needed.

In order to run classical heuristics over a graph with directional metrics, it should be extended, as depicted in the example of Fig.1: each node is to be exploded in a number of virtual nodes equal to the number of neighbours to which it is connected. Then, directional metrics are applied to simple arcs connecting these new virtual nodes, while null metrics are to be applied to arcs connecting virtual nodes related to different originating nodes.

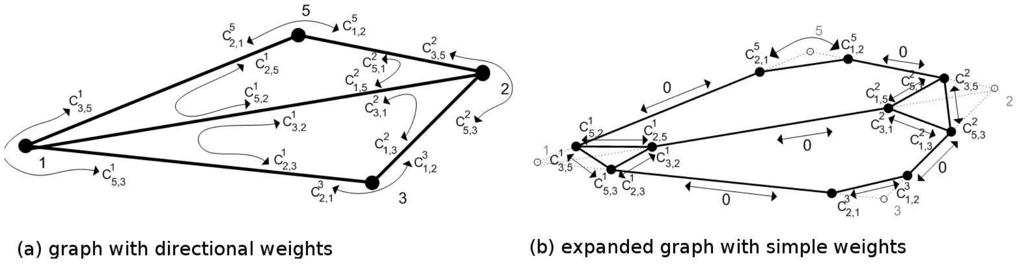

(a) graph with directional weights
(b) expanded graph with simple weights

Figure 1: Example of the graph explosion needed to apply classical algorithms

The application that inspired this work was the inter-domain routing problem at the AS level. Because of policies, an AS should profit by applying directional costs and by announcing performance in function of the directions taken to its neighbours [4]. Indeed, the AS graph having a scale-free nature (i.e. a few nodes attract most of the arcs), its few connected ASs that occupy a key position in the graph for tunnel transit would find in directional policies the most proper means to optimize their gains and to benefit from their positions. We empirically discovered that in the AS graph, an optimistic approximation for the average degree of the AS-node can be $\sqrt[3]{n}$. This finding suggests that the aforementioned extension for AS graphs requires approximately $n\sqrt[3]{n}$ new nodes and arcs, for a graph of n nodes with directional metrics.

To compare different algorithms, authors in [5] select a set of heuristics for the tree-optimization problem with a single additive constraint (end-to-end delay). Two of them have execution times and optimality gaps almost independent of the multicast group size, and execution times that scale better than other heuristics offering the same level of optimality (up to 15%). We consider these two algorithms for performance comparison over the AS graph in Sect.5. The first algorithm is a I-P2P type algorithm employing a constrained version of the Bellman Ford algorithm.

The second one is the Kompella's centralized algorithm [6]: first, it computes all the constrained shortest paths; then, it builds the closure graph of shortest paths from the root to the leaves, and it finds the constrained spanning tree of the closure graph; finally, it expands the constrained spanning tree avoiding possible loops. The overall time complexity of the Kompella's algorithm is $O(n^3D)$, where n is the number of vertexes, and D is the integer value for the delay bound. For graphs with directional metrics, the time complexity becomes $O(n^4D)$.

3 PROBLEM FORMULATION

We formulate the constrained Steiner tree optimization problem with directional metrics by Integer Linear Programming (ILP). The following formulation is independent of multiplicative constraints (e.g., the bandwidth) that can be discarded by a preliminary pruning of the graph. The following notation is used:
N is the node set.
M is the destinations group; $d \in M$ is a single destination.
(a, b, c) is the directional composed arc from a to c passing through b, where $a, b, c \in N$.
$c^0_{a,b,c}$ is the directional cost over (a, b, c).

$C^i_{a,b,c}$ is the directional additive metrics i over (a,b,c).
C^i is the end-to-end bound on metrics i.
$x^d_{a,b}$ is a binary variable equal to 1 if the arc (a,b) is used by the route for destination $d \in M$.
$f^d_{a,b,c}$ is a binary variable equal to 1 if (a,b,c) is used by the route for destination d.
$f_{a,b,c}$ is a binary variable equal to 1 if the directional arc (a,b,c) is used by the selected tree.

The objective is the minimization of the tree cost:

$$\phi(f) = \min \sum_{(a,b,c)} c^0_{a,b,c}\, f_{a,b,c} \tag{1}$$

We have $|M|$ flow conservation constraints (one route per destination):

$$\sum_{b\in N} x^d_{a,b} - \sum_{b\in N} x^d_{b,a} = \begin{cases} 1 & \text{if } i = \text{root} \\ -1 & \text{if } i = \text{d} \\ 0 & \text{otherwise} \end{cases} \quad \forall (i,d) \in (N,M) \tag{2}$$

These constraints enforce the additive end-to-end bounds:

$$\sum_{(a,b,c)} f_{a,b,c}\, c^i_{a,b,c} \leq C^i,\ \forall (i,d) \in (N,M) \tag{3}$$

These constraints enforce the hop bound:

$$\sum_{(a,b,c)} f^d_{a,b,c} \leq H_m,\ \ \forall d \in M \tag{4}$$

The per-destination directional variables are enabled by:

$$f^d_{a,b,c} \geq x^d_{a,b} + x^d_{b,c} - 1\,, \qquad \forall (a,b,c),\ \ \forall d \in M \tag{5}$$

The directional variables are enabled by:

$$|M|\ f_{a,b,c} \geq \sum_{d\in M} f^d_{a,b,c},\ \ \forall (a,b,c) \tag{6}$$

Finally, the binary domain on variables is imposed by:

$$x^d_{a,b}, f^d_{a,b,c}, f_{a,b,c} \in \{0,1\} \tag{7}$$

During the simulations, it is possible to speed up the simplex solving by: (i) setting as upper bound the objective obtained by suboptimal heuristics; (ii) indicating the priority on variables; (iii) setting the steepest edge dual simplex gradient.

4 RCOM APPROACH

We have seen that the tree-constrained tree-optimization problem with directional metrics can be relaxed to the classical constrained Steiner tree problem only by creating a larger graph with simple metrics. Alternatively, to solve this specific problem we devise an ad-hoc heuristic called Route Collection and Optimal Matching (RCOM), composed of two steps:

1. Route Collection: some feasible point-to-point routes towards each leaf node are collected
2. Optimal Matching: optimal routes are matched by minimizing the tree cost.

As opposed to IRC-PM algorithms, RCOM retains a subset of feasible routes instead of only one route per destination. With respect to I-P2P algorithms, RCOM should be more flexible in branch and bud node placement, since it can reach a wider set of solutions. Last but not least, in Sect.4.3 we explain how with pre-computation a big slice of the time-complexity can be cut off-line by computation elements.

4.1 Route Collection

To collect the per-destination routes set, we devise an ad-hoc breadth-first-search algorithm with limited depth. It starts at the root, moves to unvisited neighbours, collects the routes if a destination is attained, and so on, until no more routes can be collected. It stops at a given number of hops and during the search it prunes branches on the base of metric bounds.

This approach was inspired by the A*prune algorithm [7], proposed to solve the constrained k-shortest paths problem. Our approach differs in that : (i) since the final objective is the selection of the optimal tree, a further pruning (besides that on the additive metrics) on the basis of the route cost is performed, giving priority to the least hop routes; (ii) given that there is no need to sort the candidate routes (as A*prune does when choosing the next path to expand during the graph exploration), the number k of shortest routes is not fixed and all the found (feasible) routes are collected (i.e., we do not need a best-first-search approach).

Algorithm 4.1: ROUTE COLLECTION(G)

```
procedure POP($\overline{c}^i_\pi, h, \pi$)
  – $\overline{f}$ : per-destination vector with counters of found routes so far
  – $a$, $c^i_a$ : next 1-hop arc, $i^{th}$ metric of $a$
  – $M$ : destination group (set of leaf nodes)
  if $h = H$
    then
      if $\exists!$ leaf $d \mid c^0_\pi + SPC(\pi[h], d) < \upsilon(d)$
        then add $\pi$ to $\zeta_{cand}$
      if $\pi[h] \in M$
        then
          if $c^0_\pi < \upsilon(\pi[h])$
            then
              add $\pi$ to $\zeta_{sel}$
              $f(\pi[h]) \leftarrow f(\pi[h]) + 1$
              if $f(\pi[h]) \geq F$
                then update $\upsilon(\pi[h])$
    else
      for $i \leftarrow 1$ to $N$
        do
          if $i$ adjacent to $\pi[h]$, and $i \notin \pi$
            then
              $\pi[h+1] \Leftarrow i$
              $a \leftarrow (\pi[h-1], \pi[h], \pi[h+1])$
              if $h = 0$
                then POP($\overline{c}^i_\pi, h+1, \pi$)
                else if $c^i_\pi + c^i_a < C^i \; \forall i > 0$
                then POP($\overline{c}^i_\pi + \overline{c}^i_a, h+1, \pi$)

main
  $H \leftarrow 1$, $\overline{\upsilon} \leftarrow \overline{\infty}$, $\zeta_{cand} \leftarrow \{\pi_0 = (root)\}$
  while $\zeta_{cand} \neq \emptyset$ or $H < H_m$
    do
      extract a subroute $\pi$ from $\zeta_{cand}$
      POP($\overline{c}^i_\pi, H-1, \pi$)
      $H \leftarrow H + 1$
```

Collection algorithm

Let $\overline{\upsilon}$ be the threshold cost vector with one entry per destination. Each entry is a threshold re-calculated at each new route collection. The starting values are infinite. An entry is initialized when at least F routes have been collected for that destination; F has to be chosen conveniently (we use $F = \sqrt[3]{n}$). Each threshold is calculated as the average cost of those routes with a variance on the average cost less than the average of this variance: within the first F routes, those with a very high cost with respect to the others are not kept. In this way, the threshold has a decreasing trend, with a starting value not excessively high. The least hop routes are thus privileged because the cost bound is higher in the first hops. Favouring routes of few hops is a suitable approach for our specific problem, since long routes crossing several ASs may only have a small number of arcs in common with those previously selected, which tends to increase the cost. In this way we try to cut a lot of branches that would have been considered by general purpose solvers for (1)-(7).

We define the *projected cost* of a sub-route as the cost given by the sum of the current sub-route cost and the cost of the shortest path from the tail toward the destination. To determine the projected costs, the costs of the shortest paths need to be pre-calculated.

The pseudo-code is shown in Alg.4.1. The search starts looking for feasible routes at 2 hops, then 3, and so on. It proceeds with a graph exploration by evaluating for feasibility, at each iteration, only the routes of an equal number of hops H, up to a given hop bound H_m.

At every iteration, the sub-routes in the set ζ_{cand} are the starting point of the search. At every call of Pop(), c and d are the cumulative cost and delay of the route handled by the current route vector π with h hops. $\overline{c}^i_\pi$ indicates the cumulative metrics vector for the route handled by π, where c^i_π is the i^{th} metric of the vector, and c^0_π is the cost. When visiting the root neighbours (h=0), π is only the root, and the delay is not verified. Then, the function recursively visits every neighbour of the sub-route tail node, updating π, and evaluating the route feasibility on the cumulative delay. At the H^{th} hop, the route is collected in the set ζ_{sel} if a leaf is visited, if its cost is less than the threshold, and if the delay bound is respected; it is also added to ζ_{cand} for further expanding and possible selection in the next hop only if, for at least one destination, its projected cost is equal to or less than the threshold.

4.2 Routes Matching

The routes in ζ_{sel} subtend a subgraph built as the superimposition of their directional arcs. The final least cost tree is a composition of directional arcs linking the root to the leaves.

Let $p \in A$ indicating the affiliation of route p to subgraph A, $a \in A$ and $a \in p$ the affiliation of directional arc a to subgraph and to route. Let y^d_p be a parameter equal to 1 if the leaf node d is a tail or an intermediate node of p, 0 otherwise. The following binary variables are used: x_a is 1 if a is enabled in the tree solution; f_p is 1 if p has been selected to reach its destination. The objective is the minimization of the tree cost as the sum of the costs of the directional arcs:

$$g(x) = \min \sum_{a \in A} c_a x_a \tag{8}$$

By imposing that each destination is crossed by at least one of the enabled routes, we can assure that all the leaves are reached and that the delay constraint is satisfied.

$$\sum_{p \in \zeta_{sel}} y^d_p f_p = 1, \quad \forall d \in M \tag{9}$$

The route enabling is linked with the enabling of the directional arcs it is composed of by:

$$x_a \geq f_p \;\; \forall p \in \zeta_{sel}, \; \forall a \in p \tag{10}$$

The binary domain of the variables is imposed:

$$x_a, f_p \in \{0, 1\} \tag{11}$$

4.3 Pre-computation and Complexity

In the collection algorithm, the majority of the time is spent in computing the (unconstrained) shortest path costs, which are needed to determine the projected costs. Our proposition is to pre-compute the shortest paths, prior to any request, and after any topological and cost update. This can stand when the cost and topology update arrival rate is much less than the request arrival rate. Hence prior to the request (characterized by root, leaves, and end-to-end constraints) a simplified version of the Floyd-Warshall algorithm [8] can be pre-computed to calculate the cost of the shortest paths (SPC matrix in Alg.4.1) from any node to any node (A2ASP). For sake of completeness, it is worth mentioning that when the destination group M can be known in advance, $|M||N|$ executions of the Dijkstra algorithm could fill sufficiently the SPC matrix; nevertheless, in the following we assume that M is not known in advance.

Floyd's algorithm takes $O(n^3)$ time to compute, which becomes $O(n^4)$ for graphs with directional metrics assuming an average degree of $\sqrt[3]{n}$. The subsequent breadth-first search would have, without pruning, a time complexity of $O(n^{\frac{1}{3}H_m})$ for the worst case, approximating the base (branching factor) to $\sqrt[3]{n}$, but, because of pruning, it is more efficient than that.

To improve the execution time, A2ASP computation should be run off-line prior to any request, and triggered by each topology and costs update. In this way the post-request complexity of the collection becomes $\mathrm{O}(n^{\frac{1}{3}H_m})$ for the worst case. Indeed, computation elements as the Path Computation Element (PCE) [10] currently studied at the IETF for inter-AS LSPs computation, or the AS Selection Element (ASE) proposed in [4] for constrained disjoint inter-AS routes computation, could perform such operations prior to requests coming with a different time scale to the A2ASP computation without overloading the router.

The centralized heuristics proposed so far for constrained multicast routing, such as those in [5], do not have a sub-algorithm independent from the constraint values. For example, Kompella's algorithm computes the constrained A2ASP to build the closure graph with a complexity proportional to the delay bound (see Par.2.3); Zhu's algorithm [9] uses as a starting point a least-delay spanning tree. Both Kompella's and Zhu's algorithms have an overall complexity equal to the post-request complexity, which is, for our graph, more than $\mathrm{O}(n^4)$.

Then, the optimal route matching is solved through the ILP formulation (8)-(11) that has a number of variables $\ll (1 + H_m)|\zeta_{sel}|$ and a number of constraints $\ll H_m|\zeta_{sel}||M|$ since the number of shared arcs is expected to be $\ll H_m|\zeta_{sel}|$. The limited number of variables in the objective, and the relaxation of (9), guarantee that the subproblem is tractable.

Finally, it is worth mentioning that given the breath-first-search nature of the collection algorithm and the additive constraint transparency of the route matching, an extension of the RCOM approach to multiple additive constraints would scale with the number of constraints.

5 PERFORMANCE ANALYSIS

We compare all the described algorithms in terms of optimality and execution time, and characterize the selected trees, in the case of a single metric. We chose to use realistic topologies: we dumped the AS whois database with interconnections available at [11]. A first topology is selected in the following way: among all the ASs, only those with at least 7 adjacencies are kept; then, only those ASs with more than 2 adjacencies within the selection are kept in. The final topology, called ATL7, has 643 AS-nodes. A second and a third topologies, TOP100 and TOP300, are built with the 100 and 300 most connected nodes of ATL7. Then we generated the directional metrics to apply. We ranked the nodes on the basis of their degree. We then assigned transit delays and inter-AS capacities normally around different values on the basis of AS ranking. The transit costs are calculated with a $log(x)/x$ law where x is the minimal directional capacity between two neighbours ASs. More details can be found in [4] (not included because of page limit).

5.1 Algorithms' performance

We run one new instance for each size of the multicast group. Root and leaves are generated randomly. The delay bound is fixed to 1.5 s and the bandwidth is randomly set bigger than 5 Mb/s.

5.1.1 Execution times

Figs.2ace display the execution times obtained for TOP100, TOP300 and ATL7 as a function of $|M|$. For ATL7 the upper hop bound is fixed to 8 (that is a sufficient value for this topology [4]), while for TOP100 and TOP300 it is fixed to 5 (also sufficient because of the smaller diameter). The case of the optimal approach is not plotted: it grows more than exponentially with $|M|$. For RCOM we display: the total time (RCOM), the times of the collection (.RC) and matching (.OM) procedures. The time of the A2APC (Floyd) is separated since we assume that it can be pre-computed. Then, the case of IRC-PM, I-P2P and Kompella's (KOMP) algorithms is also plotted. We can affirm that: (i) the complexity of the RCOM due to matching becomes more negligible as the topology grows; (ii) as expected, KOMP is lower bounded by Floyd since it implements a constrained version of Floyd; (iii) including the ASAPC, RCOM has an execution time comparable to that of KOMP; without (assuming A2APC pre-computation) it has the lowest time almost always; (iv) I-P2P and IRC-PM have a similar behaviour, and both seem to scale worst with $|M|$ and the topology size than the other algorithms; (v) an increase of M does not worsen the complexity of RCOM and of KOMP.

Hence RCOM and KOMP are the least complex algorithms, independently of the group size, and RCOM is much more competitive if the A2ASPs can be pre-computed.

5.1.2 Optimality

Figs.2bd display the excess cost ratio w.r.t. the optimal solution for TOP100 and TOP300. For ATL7 this could not be computed, but Fig.2f displays the excess cost ratio w.r.t. RCOM for ATL7. We can affirm that: (i) RCOM is the best option since it gave with TOP100 the optimal solution always, and with TOP300 often optimal, largely under 10%; (ii) KOMP had always 50% excess cost w.r.t. RCOM; (iii) I-P2P and IRC-PM gave very close solutions.

5.2 Tree Characterization

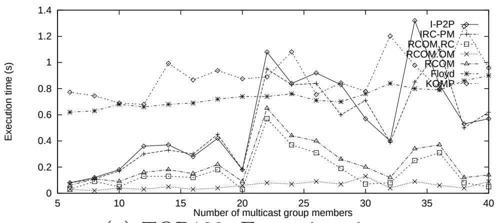
5.2.1 Nodes type

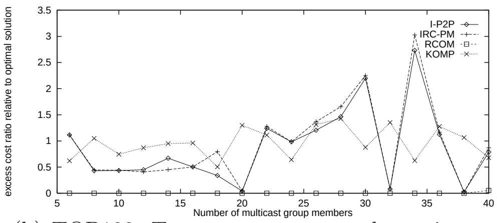
Fig.3 displays the number of intermediate, branch and bud nodes as defined in the Introduction. The ATL7 results are considered.

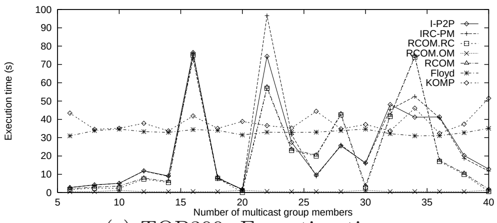
We can affirm that: (i) the number of intermediate nodes increases with $|M|$ (reflecting the fact that more isolated branches are selected as the number of leaves increases), and we can see that because of its irrespective behaviour IRC-PM selects many more isolated branches than the other algorithms, with RCOM having a lower number of intermediate nodes; (ii) the number of branch nodes is interestingly lower bounded by KOMP, and also interesting is the fact that KOMP sets un upper bound on the number of bud nodes; (iii) on the contrary, RCOM has more branch nodes and less bud nodes than KOMP. (ii) and (iii) may be explained as follows. While RCOM uses an unconstrained A2ASP pre-computation for projecting costs during the constrained exploration to pragmatically discard routes, KOMP uses a constrained A2ASP computation for producing a closure graph where the minimum spanning tree is computed: KOMP falls more easily in local minima represented by longer routes, the closure graph not being sensitive to real hop numbers, and thus the possibility of branching at leaves becomes higher.

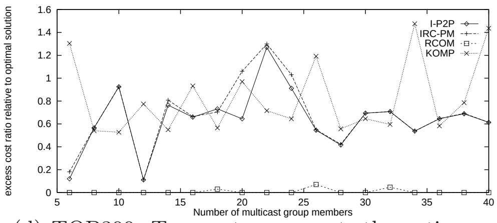
5.2.2 Tree slimness

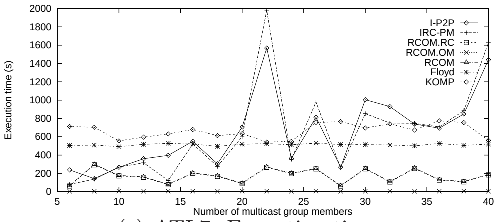
Let the utility of a directional arc be the number of destinations it can serve minus one. Let the tree slimness be defined as the ratio between the sum of all these utilities and the number of directional arcs the tree is composed of. The slimness expresses how much the selected tree is exploited, or how much the selected tree has backbone directional arcs. This is not intended as an overall evaluation parameter of a tree, but just as a silhouette feature; however, the less optimal a tree is, the smaller its slimness is expected to be. We are motivated in analyzing this parameter because in multi-layer network, a major application of these algorithms, a computation in one


(a) TOP100: Execution times


(b) TOP100: Tree cost gaps w.r.t. the optimum


(c) TOP300: Execution times


(d) TOP300: Tree cost gaps w.r.t. the optimum


(e) ATL7: Execution times

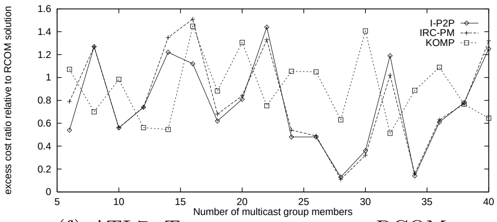
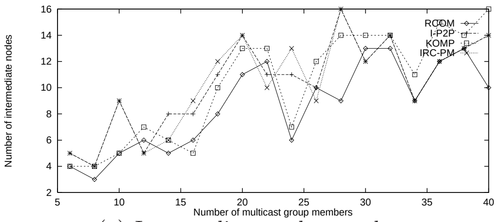
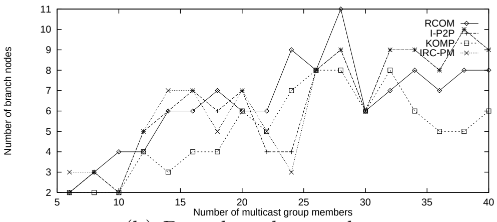
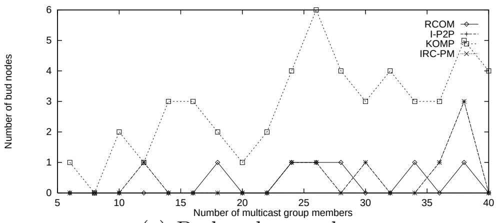

(f) ATL7: Tree cost gaps w.r.t. RCOM

Figure 2: Results for TOP100 (a)-(b), TOP300 (c)-(d) and ATL7 (e)-(f) topologies


(a) Intermediate nodes number


(b) Branch nodes number


(c) Bud nodes number

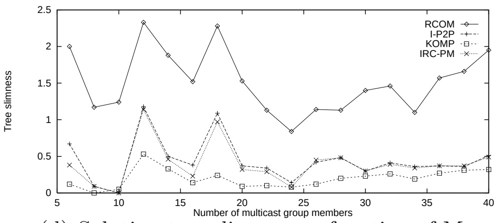

(d) Solution tree slimness as function of M

Figure 3: Characterization of the solution tree

layer can be followed by computations in other lower layers along the routes chosen in the upper layer. Hence the slimmer the tree is, the less the complexity for the under-layer path computation (and maybe signalling) could be in the case of multi-layer networks.

Fig.3d displays the slimness of solution trees obtained for the ATL7 graph. We can affirm that: (i) RCOM offers the best slimness; (ii) KOMP offers the worst slimness; (iii) I-P2P and IRC-PM behave better than KOMP w.r.t. the slimness, but significantly worst than RCOM.

6 CONCLUSIONS

We dealt with the constrained Steiner problem in the case of directional metrics. Directional metrics are more useful than simple metrics in multi-layer routing problems at those layers where business or social policies need to be taken into account. In particular, our study was on the inter-domain AS-level tree selection problem. We noticed that the heuristics proposed so far for the case with simple metrics do not scale well in the case of directional metrics. Furthermore, we remarked the absence of adequate routing algorithms aware of the possibility of pre-computing part of the job by computation elements, and of the subsequent sub-layer path computation characterizing multi-layer networks and hierarchical architectures.

To overcome these deficiencies, we devised a search algorithm called RCOM, and demonstrated that: (i) exploiting pre-computation RCOM is faster than the two best algorithms at the state of the art in the case of a single additive metric; (ii) multiple metrics do not affect RCOM asymptotical time complexity; (iii) it often reaches the optimality and the optimality is always under 10% on realistic AS graphs; (iv) it produces efficient trees w.r.t. under-layer computation issues characterizing multi-layer networks.

REFERENCES

[1] A. Farrel, I. Bryskin, “GMPLS Architecture and Applications”, Morgan Kaufmann, 2006.

[2] Bin Wang, J.C. Hou, “Multicast routing and its QoS extension: problems, algorithms, and protocols”, Network, IEEE Vol. 14, Nb. 1,Pp: 22-36 (2000).

[3] S. Chopra1, M.R. Rao, “The Steiner tree problem I: Formulations, compositions and extension of facets”, Journal Mathematical Programming. Vol.64, pp:209-229 (1994).

[4] S. Secci, J.-L. Rougier, A. Pattavina, “On the Selection of Optimal Diverse AS-Paths for Inter-Domain IP/(G)MPLS Tunnel Provisioning”, to appear in Proceedings of 4^{th} International Telecommunication Networking Workshop on QoS in Multiservice IP Networks (IT-NEWS 2008), 13-15 Feb. 2008, Venezia, Italy.

[5] H.F. Salama, D.S. Reeves, Y. Viniotis, “Evaluation of multicast routing algorithms for real-time communication on high-speed networks”, IEEE J. SAC 15, No. 3, pp::332-345 (1997).

[6] VP Kompella, JC Pasquale, GC Polyzos, “Multicast routing for multimedia communication”, IEEE/ACM Transactions on Networking (1993).

[7] G Liu, KG Ramakrishnan, “A*Prune: an algorithm for finding K shortest paths subject to multiple constraints ” INFOCOM 2001

[8] R. W. Floyd, “Algorithm 97: Shortest Path”, Communications of the ACM 5 (6): 345, (June 1962)

[9] Q. Zhu, M. Parsa, J.J. Garcia-Luna-Aceves, “A source-based algorithm for delay-constrained minimum-costmulticasting”, in Proc. of INFOCOM ’95.

[10] A. Farrel, J. P. Vasseur, J. Ash, “A Path Computation Element (PCE)-based architecture”, RFC 4655, Aug. 2006.

[11] The CIDR report, `http://www.cidr-report.org/`.