GMPLS Routing Strategies based on the Design of Hypergraph Layouts

Jean-Claude Bermond, David Coudert, Joanna Moulierac, Stéphane Pérennes, Ignasi Sau Valls, Fernando Solano Donado

▶ **To cite this version:**

Jean-Claude Bermond, David Coudert, Joanna Moulierac, Stéphane Pérennes, Ignasi Sau Valls, et al.. GMPLS Routing Strategies based on the Design of Hypergraph Layouts. [Research Report] RR-6842, INRIA. 2009. inria-00360576

HAL Id: inria-00360576

https://hal.inria.fr/inria-00360576

Submitted on 11 Feb 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

GMPLS Routing Strategies based on the Design of Hypergraph Layouts

Jean-Claude Bermond — David Coudert — Joanna Moulierac — Stéphane Pérennes — Ignasi Sau — Fernando Solano Donado

N° 6842

February 2009

Thème COM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--6842--FR+ENG

GMPLS Routing Strategies based on the Design of Hypergraph Layouts

Jean-Claude Bermond*†, David Coudert*†, Joanna Moulierac*†, Stéphane Pérennes*†, Ignasi Sau*‡†, Fernando Solano Donado§¶

Thème COM — Systèmes communicants
Projets Mascotte

Rapport de recherche n° 6842 — February 2009 — 17 pages

Abstract:

All-Optical Label Switching (AOLS) is a new technology that performs packet forwarding without any Optical-Electrical-Optical (OEO) conversions. In this paper, we study the problem of routing a set of requests in AOLS networks using GMPLS technology, with the aim of minimizing the number of labels required to ensure the forwarding. We first formalize the problem by associating to each routing strategy a logical hypergraph whose hyperedges are dipaths of the physical graph, called *tunnels* in GMPLS terminology. Such a hypergraph is called a *hypergraph layout*, to which we assign a cost function given by its physical length plus the total number of hops traveled by the traffic. Minimizing the cost of the design of an AOLS network can then be expressed as finding a minimum cost hypergraph layout.

We prove hardness results for the problem, namely $C \log n$ hardness for directed networks and non-existence of PTAS for undirected networks, where C is a a positive constant and n is the number of nodes of the network. These hardness results hold even is the traffic instance is a partial broadcast. On the other hand, we provide an $\mathcal{O}(\log n)$-approximation algorithm to the problem for a general network. Finally, we focus on the case where the physical network is a path, providing a polynomial-time dynamic programming algorithm for a bounded number of sources, thus extending the algorithm of [2] for a single source.

Key-words: AOLS, GMPLS, routing, labels, hypergraphs

This work has been partly funded by the European project IST FET AEOLUS and the project "Optimization Models for NGI Core Network" (Polish Ministry of Science and Higher Education, grant N517 397334).

* MASCOTTE, INRIA, I3S, CNRS, Univ. Nice Sophia, Sophia Antipolis, France.

† `firstname.lastname@sophia.inria.fr`

‡ Combinatorics Group at Applied Maths. IV Dept. of UPC, Barcelona, Spain

§ Institute of Telecommunications, Warsaw University of Technology, Poland

¶ `fs@tele.pw.edu.pl`

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Stratégies de routage dans les réseaux GMPLS basées sur des représentations d'hypergraphes

Résumé : La commutation d'étiquette tout optique (*All-Optical Label Switching*, AOLS) est une nouvelle technologie permettant de faire suivre des paquets sans conversion optique-electronique-optique (OEO). Dans ce rapport, nous étudions le problème du routage d'un ensemble de requêtes dans les réseaux AOLS utilisant la technologie GMPLS, avec pour objectif de minimiser le nombre totale d'étiquettes utilisées.

Nous commençons par formaliser le problème en associant à chaque stratégie de routage un hypergraphe logique dont les hyperarêtes modélisent des chemins dans le graphe physique, appelés *tunnels* dans la terminologie GMPLS. Un tel hypergraphe est appelé un *hypergraphe support*. Nous lui associons la fonction de coût suivante: longueur physique plus le nombre total de sauts nécessaires au routage du trafic. Minimiser le coût de la conception d'un réseau AOLS peut alors s'exprimer comme la minimisation du coût de l'hypergraphe support.

Nous établissons des résultats sur l'inapproximabilité du problème. En particulier, nous montrons que la version orientée du problème n'est pas approximable à un facteur $C \log n$, et que la version non-orientée du problème n'admet de schéma d'approximation polynomial, où C est une constante positive et n est le nombre de sommets du réseau. Ces résultats d'inapproximabilité sont valables même si l'instance est une diffucion partielle. D'autre part, nous proposons un algorithme d'approximation de facteur $\mathcal{O}(\log n)$ pour un réseau général. Finallement, lorsque la topologie physique est un chemin orienté et que le nombre de sources est borné, nous proposons un algorithme polynomial basé sur la programmation dynamique. Ceci étend l'algorithme proposé dans [2] pour une unique source.

Mots-clés : AOLS, GMPLS, routage, étiquettes, hypergraphes

1 Introduction

All-Optical Label Switching (AOLS) [11] is an approach to route packets transparently and all-optically, thus allowing a speed-up of the forwarding. This very promising technology for the future Internet applications also brings new constraints and new problems. Indeed, since the forwarding functions are implemented directly at the optical domain, a specific correlator (device) is needed for each optical label processed in the node. Therefore, it is of major importance to reduce the number of employed correlators in every node, implying a reduction in the number of labels (as referred in the rest of the paper) that are going to be used by the traffic. Due to its flexibility as a control plane and to the fact that it handles traffic forwarding, the Generic MultiProtocol Label Switching (GMPLS) is the most promising protocol to be applied in AOLS-driven networks.

In GMPLS, traffic is forwarded through logical connections called Label Switched Paths (LSPs). When GMPLS is used with packet-based network, packets are associated to LSPs by means of a label, or tag, placed on top of the header of the packet. In this way, routers - called Label Switched Routers (LSRs) - can distinguish and forward packets.

The GMPLS standards allow packets to carry a set of labels in their header, conforming a stack of labels. Even though a packet may contain more than one label, LSRs must only read the first (or top) label in the stack in order to take forwarding decisions. This helps to reduce both the number of labels that need to be maintained on the core LSRs and the complexity of managing data forwarding across the backbone.

Stacking labels and label processing, in general, are standardized by the following set of operations that an LSR can perform over a given stack of labels:

- SWAP: replace the label at the top by a new one,
- PUSH: replace the label at the top by a new one and then push one or more onto the stack, and
- POP: remove the label at top in the label stack.

The labels stored in the forwarding table are significant only locally at the node and swapped all along the LSP (See Fig. 1).

Solutions deployed by GMPLS for reducing the number of labels are *label merging* [5, 13, 15] (not discussed here) and *label stacking* [14, 17]. With label stacking, when two or more LSPs follow the same set of links, they can be routed together "inside" a higher-level LSP, henceforth a *tunnel*. In order to setup a tunnel, multiple labels are placed in the packet's header.

Fig. 1 represents the general operations needed to configure a tunnel with the use of label stacking. At the entrance of the tunnel, λ PUSH are performed in order to route the λ units of traffic through the tunnel. Then, only one operation (either a SWAP or a POP at the end of the tunnel) is performed in all the nodes along the tunnel, regardless of λ. In this figure, a stack of size 2 is used to route the λ LSPs in one tunnel from node A to node E. The top label l is swapped and replaced at each hop: by l_1 at node B, by l_2 at node C,

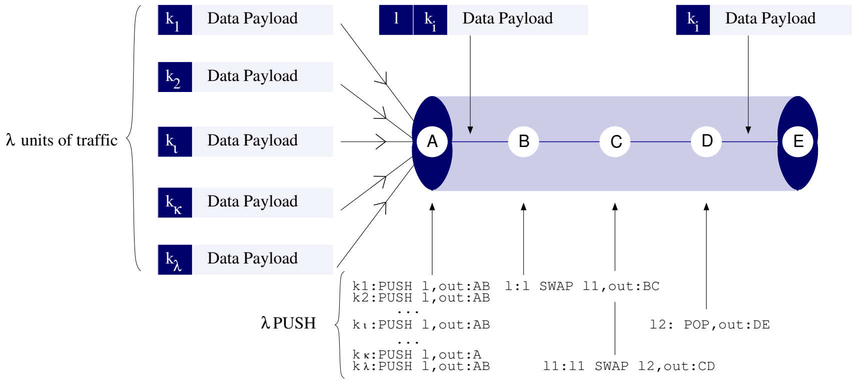

Figure 1: GMPLS operations performed at the entrance and at the exit of a tunnel.

and is finally popped at node D. The λ units of traffic, at the exit of the tunnel at node E can end or follow different paths according to their bottom label k_i, for all $i \in \{1, 2, ..., w\}$ in the stack.

A consequence of the way in which the GMPLS operations can be configured at LSRs is the following: traffic can enter in any node of a tunnel but can exit in only one point, the last node of the tunnel. In other words, when some traffic is carried by a tunnel, it follows the tunnel until its end.

Since the number of labels used for GMPLS forwarding affects the cost of the AOLS architecture, in this paper we mainly focus on the minimization of the number of labels used. In our previous example, the total cost $c(T)$ of this tunnel T from node A to node E in terms of number of labels is $c(T) = \lambda + \ell(T) - 1$, where λ is the number of units of traffic forwarded through this tunnel and $\ell(T)$ is its length in terms of number of hops (which is 4 on this example). We will formally define the cost function of the problem in Section 2.

Previous work and our contribution. The label minimization problem in GMPLS networks has been widely studied in the literature during the last few years [14, 17, 5, 13, 15, 16]. All these articles focus mainly on proposing and analyzing heuristics to the problem, but there is a lack of theoretical results, like computational complexity or bounds on the approximation ratio of the proposed algorithms. For instance, in [16] the authors propose heuristics for routing a set of demands in AOLS networks when routers have limited number of available optical correlators. Very recently [2], the problem has been studied for the directed path from a more theoretical point of view. Namely, in [2] the authors present a polynomial-time optimal algorithm for the case when all traffic issues from a single source and an $\mathcal{O}(\log n)$-approximation algorithm with arbitrary number of sources, where n is the number of nodes of the network.

In this article we provide the first theoretical framework for the label minimization problem in general GMPLS networks. We translate the problem into finding a set of dipaths in a directed hypergraph. With this new formulation, it turns out that the problem is very similar to classical Virtual Path Layout (VPL) problems originating from ATM networks. We provide hardness results and approximation algorithms for the problem in general graphs. The approximation algorithms strongly rely on the already known algorithms for VPL problems. Finally, we focus on the path topology, extending the dynamic programming approach presented in [2] to any bounded number of sources. If there are k sources, the main result is an optimal algorithm with running time $n^{\mathcal{O}(k)}$. That is, the problem is polynomial in the path for any fixed number of sources.

Organization of the paper. In Section 2 we formally state the problem in terms of hypergraph layout and fix the notation to be used throughout the article. In Section 3 we prove that the problem is NP-hard to approximate within a factor $C \log n$ for some positive constant $C > 0$ in the directed case, and that it does not accept a PTAS in the undirected case, unless P=NP. In Section 4 we provide an approximation algorithm to the problem in general undirected graphs with an approximation ratio $\mathcal{O}(\log n)$, where n is the number of nodes of the network. In Section 5 we focus on the directed path topology and present a dynamic programming approach solving the problem in polynomial time when the number of sources is fixed.

2 GMPLS Logical Network Design as a Hypergraph Layout Problem

The logical network design problem that we address can be roughly described as follows: we are given a (directed or undirected) graph G together with a set of traffic demands between pairs of vertices in G, and we must find a set of tunnels of minimum cost allowing to route all traffic demands. Let us now precise each one of these terms.

A *tunnel* is simply a directed path (or dipath) in G, and due to the technological constraints discussed in Section 1, traffic can enter anywhere in the tunnel but must leave only at the end of the tunnel. To define the problem formally we need following notation:

- $G = (V, E)$ is the underlying graph, which can be directed or undirected.
- d_i is the demand from $a_i \in V$ to $b_i \in V$, with multiplicity m_i. D is the set of all demands.
- $P(G)$ is the set of all simple dipaths in G.
- t stands for a tunnel, and T is the set of tunnels, that is $t \in T \subseteq P(G)$.
- we are given a length function $\ell(e) : E \to \mathbb{R}^+$.
- for a tunnel t, $\ell(t) = \sum_{e \in t} \ell(e)$ is its length and $w(t)$ is the amount of traffic it carries.

Note that a priori $w(t)$ depends on the routing policy. The cost of a tunnel t is then $w(t) + (\ell(t) - 1)$, and the cost of a set of tunnels T is

$$\sum_{t \in T} (w(t) + \ell(t) - 1)\,. \tag{1}$$

Each tunnel can be seen as a directed hyperedge on the vertex set of G. This observation naturally leads to the definition of a hypergraph layout.

Definition 1 (Hypergraph layout) *Given a graph G and a set $T \subseteq P(G)$, $H(T)$ is the directed hypergraph with $V(H(T)) = V(G)$, and where for each tunnel $t \in T \subseteq P(G)$ there is a directed hyperedge in $H(T)$ connecting any vertex of t to the end of t. $H(T)$ is called a* hypergraph layout*.*

Note that a hypergraph layout is always directed, regardless of whether the underlying graph G is directed or not. Note also that a hypergraph $H(T)$ defines a virtual topology on G. A hypergraph layout $H(T)$ is said to be *feasible* if for each demand $d_i \in D$ there exists a dipath in $H(T)$ from a_i to b_i. The problem can then be simply expressed as finding a feasible hypergraph layout of minimum cost. Let us now simplify the cost function of Equation (1).

Given a hypergraph layout $H(T)$, let $L(d_i)$ be the number of hyperedges that demand i uses, and let $d_H(a_i, b_i)$ be the distance from vertex a_i to vertex b_i in $H(T)$. Then the term $\sum_{t \in T} w(t)$ of Equation (1) can be rewritten as $\sum_{d_i \in D} L(d_i) \cdot m_i$ and, since $L(d_i) \geq d_{H(T)}(a_i, b_i)$, we conclude that in an optimal solution the routing is necessarily using shortest paths in the hypergraph layout. It follows that the cost function of Equation (1) can be rewritten w.l.o.g. as

$$\sum_{t \in T} (\ell(t) - 1) + \sum_{d_i \in D} d_H(a_i, b_i) m_i. \tag{2}$$

The cost of a solution is of bicriteria nature. The first part is the cost of the hypergraph structure; we call it the *length* of the layout. The second part is the total distance that the traffic travels in the hypergraph; we call it the *total hop count*. Both cost function parts are very much conflicting. On the one hand, to minimize the hop count, it is enough to take a shortest tunnel connecting any source to any destination. On the other hand, to minimize the length of the layout, it is enough to use a minimum edge-weighted connected hypergraph H such that for each request $d_i \in D$, vertices a_i and b_i lie on the same connected component of H. Summarizing, the problem can be stated as follows.

> MINIMUM COST HYPERGRAPH LAYOUT: Given a graph G with a length function and a set D of traffic demands, find a feasible hypergraph layout of minimum cost, where the cost of a hypergraph layout is defined as in Equation (2).

Depending on whether the underlying network G is directed or undirected, the problem is denoted MINIMUM COST DIRECTED HYPERGRAPH LAYOUT or MINIMUM COST UNDIRECTED HYPERGRAPH LAYOUT, respectively. It makes sense also to consider the decision version in which we are also given two positive integers C_L, C_H and the objective is to decide whether there exists a layout with length less than C_L and total hop count less than C_H.

The cost function of Equation (2) can be naturally generalized to

$$\alpha \cdot \sum_{t \in T} c(t) + \beta \cdot \sum_{d_i \in D} d_H(a_i, b_i) m_i \ , \tag{3}$$

where α and β are positive constants and $c(t)$ is a general cost function $c : P(G) \rightarrow \mathbb{R}^+$.

Relation with VPL problems. This layout design problem defined above is quite similar to well studied VPL problems in ATM networks, in which one imposes a constraint on the logical structure and then wishes to minimize either the maximum distance [3] or the average distance [7] traveled by the traffic. Concerning hardness and approximation, we shall see in the sequel of the article that the problem we study inherits most of the characteristics of the classical VPL problems studied since the 80's. It is not surprising that, even if new technologies like GMPLS are proposed to cope with the increasing bandwidth of communication networks, the computational complexity of the problems associated to these technologies remains essentially the same.

Nevertheless, there is a crucial difference between the GMPLS version and the classical VPL version of ATM networks. Indeed, we have seen that the GMPLS logical network design can be translated into finding a set of paths in a *directed hypergraph*, whereas the existing models for VPL problems deal with *digraphs* without multiple edges. This feature will be exploited in the dynamic programming approach for the path presented in Section 5. Finally, it is important to note that if there is a single source in the the GMPLS version (or, more generally, if the traffic instance is such that in an optimal solution each hyperedge has exactly 2 vertices), then the problem is basically equivalent to a classical VPL problem.

3 Hardness Results

In this section we give hardness results for the Minimum Cost Hypergraph Layout problem. We distinguish two cases according to whether the underlying network is directed or not. We focus on those cases in Section 3.1 and 3.2, respectively.

3.1 Directed version

Theorem 1 *The* Minimum Cost Directed Hypergraph Layout *problem cannot be approximated within a factor $C \log n$ for some constant $C > 0$, even if the instance is a partial broadcast, unless* P = NP.

Proof. The reduction is from Minimum Set Cover[1]. Raz and Safra [12] proved that Minimum Set Cover is not approximable within a factor $C \log n$, for some constant $C > 0$, un-

[1]Given a finite set $\mathcal{S}$ and a collection $\mathcal{C}$ of subsets of $\mathcal{S}$, the aim is to find a subcollection $\mathcal{C}'$ of $\mathcal{C}$ of minimum cardinality that covers all the elements of $\mathcal{S}$.

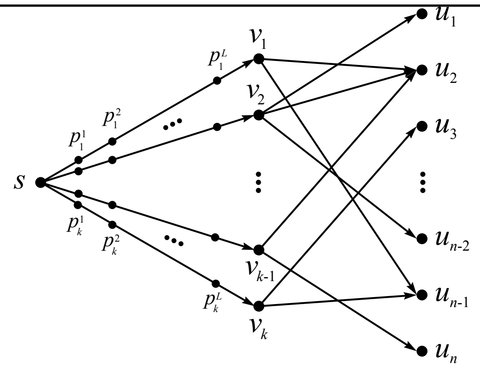

Figure 2: Reduction in the proof of Theorem 1.

less P = NP. To a Set Cover instance with sets $S_1, S_2, \ldots, S_k$, with $S_i \subseteq \{a_1, a_2, \ldots, a_n\}$, we associate the following graph:

- We start with a distinguished node s.
- For each set S_i we introduce a node v_i and a directed path of length $L+1$ (L is a constant to be specified later) from s to v_i through L new vertices $p_i^1, p_i^2, \ldots, p_i^L$.
- For each element a_j we introduce a vertex u_j and, for each vertex v_i we add the arcs (v_i, u_j) if $a_j \in S_i$.
- The requests are from s to u_j, for $i = 1, \ldots, n$.

This construction is illustrated in Fig. 2. Let OPT be the optimal cost to the Minimum Cost Directed Hypergraph Layout instance, and let OPT_{VC} be the optimal cost to the Minimum Vertex Cover instance.

Note that any cover defined by $I \subseteq \{1, 2, \ldots k\}$ induces a solution of Directed Hypergraph Layout obtained as follows: we use a tunnel of cost L connecting node s to each node $v_i, i \in I$ corresponding to a set taken in the cover. Then we connect each node $v_i, i \in I$ to the vertices $u_j, j \in S_i$. Finally, if a node u_j has more than one incoming tunnel (which means that a_j is covered more than once), we remove extra ones. A solution induced by an optimal cover has length cost $L \cdot OPT_{VC}$, and the hop count cost is $2n$, so $OPT \leq L \cdot OPT_{VC} + 2n$.

Conversely, given a layout, the paths from s to v_i used by some tunnel must induce a cover, so $OPT \geq L \cdot OPT_{VC} + n$. Putting all together,

$$L \cdot OPT_{VC} + n \leq OPT \leq L \cdot OPT_{VC} + 2n.$$

By choosing L to be large enough, the gap for the Minimum Cost Directed Hypergraph Layout problem can be made as large as in Minimum Set Cover. Since, unless P = NP, approximating Minimum Set Cover within a factor $C \log n$ for some constant $C > 0$ is NP-hard [12], our result follows. $\square$

3.2 Undirected version

Theorem 2 *The* Minimum Cost Undirected Hypergraph Layout *problem does not accept a* PTAS *even if the instance is a partial broadcast, unless* P = NP.

Proof. The reduction is from Minimum Steiner Tree[2], which is know to be APX-hard [4], hence it does not accept a PTAS unless P = NP.

Given an instance $(G = (V, E), S \subseteq V)$ of Minimum Steiner Tree problem on n vertices, we build an instance of Minimum Cost Undirected Hypergraph Layout problem by subdividing $\Omega(n^3)$ times each edge of G and considering as request set a partial broadcast from any vertex in S to all the others vertices in S. Note that subdividing edges is equivalent to setting $\alpha >> \beta$ in the cost function of Equation (3). In other words, the total hop count is negligible compared to the length of the layout. It is then clear than any optimal solution to the Minimum Cost Undirected Hypergraph Layout corresponds to a minimum cost Steiner tree in G spanning all the elements in S. Let OPT be the optimal cost to the Minimum Cost Undirected Hypergraph Layout instance, and let OPT_{ST} be the optimal cost to the Minimum Steiner Tree instance. Let M be the number of times we have subdivided the edges of G. Summarizing,

$$OPT = M \cdot OPT_{ST} + o(M \cdot OPT_{ST}).$$

The existence of a PTAS for Minimum Cost Undirected Hypergraph Layout would yield a PTAS for Minimum Steiner Tree, which is impossible unless P = NP. $\square$

4 Approximation Algorithms

In this section we provide approximation algorithms for Minimum Cost Hypergraph Layout problem. Unless said otherwise, we focus on the undirected version, for which the description of the algorithms is easier, although the main ideas can be adapted to the directed version with slight modifications. For the sake of presentation, we describe our algorithms when the network is a path, a tree, and a general graph in Sections 4.1, 4.2, and 4.3, respectively. The approximation algorithm for the directed path network appeared also in [2], we include it here for the sake of completeness.

4.1 Case of the path

First assume that the instance is a weighted all-to-all (i.e., there is a traffic demand between each pair of nodes), and that n is a power of two (otherwise, just add dummy vertices). Then one simply uses the following binary layout: We connect node 0 to node $n/2$, node

[2]Given an edge-weighted graph $G = (V, E)$ and a subset $S \subseteq V$, find a connected subgraph with minimum edge-weight containing all the vertices in S. We can assume, by subdividing edges, that all edge-weights equal 1.

$n/2$ to node n, and we use recursively the binary layout for $n/2$ on the subpaths $[1, n/2]$ and $[n/2, n]$. It is clear that any traffic demand can be routed in this layout with at most $\log n$ hops, and that the length of this layout is bounded above by $\log n \cdot \ell([1, n])$, where $\ell([1, n])$ denotes the length of the tunnel going from node 1 to node n. Therefore the cost of this layout is $\log n \cdot \sum_{d_i \in D} m_i + \log n \cdot \ell([1, n])$. Since any layout costs at least $\sum_{d \in D} m_i + \ell([1, n])$, this provides a $\log n$-approximation in the all-to-all case.

Now, for a general traffic pattern, it is not always the case that $\ell([1, n])$ is a lower bound on the length of the layout. We define the *span* of an instance as the minimum set of edges such that any demand can be routed using only those arcs. Note that the span is indeed a set of intervals such that any traffic demand is routed within one of these intervals. Let ℓ_0 denote the length of the span. Then any layout costs at least $\sum_{d_i \in D} m_i + \ell_0$, and using the binary layout on each interval of the span we can define a layout with length $\log n \cdot \ell_0$ and total hop count $\log n \cdot \sum_{d_i \in D} m_i$. Summarizing,

Proposition 1 *When the network is a path, there exists a polynomial-time approximation algorithm for* Minimum Cost Hypergraph Layout *problem with an approximation ratio* $\mathcal{O}(\log n)$.

4.2 Case of the tree

In [3] the authors studied the design of virtual layouts in ATM networks. Their model deals with point-to-point connections in the virtual graph, whereas in Minimum Cost Hypergraph Layout problem, a tunnel can carry more than one request. Nevertheless, we can use the results of [3] to obtain good approximation algorithms. Namely, we are interested in the following result which establishes the trade-off between the maximum load c and the diameter of a virtual layout allowing to route an all-to-all traffic in a general tree.

Theorem 2 (Bermond *et al.* [3]) *In a general tree on n nodes with all-to-all traffic, for each value of* $c \in \{1, \ldots, n\}$ *there exists a virtual layout allowing to route all traffic with diameter at most* $10c \cdot n^{\frac{1}{2c-1}}$ *and load at most c. In addition, such a layout can be constructed in polynomial time.*

In particular, if we set $c = \frac{\log n + 1}{2}$, Theorem 2 implies that we can find in polynomial time a layout with load $\mathcal{O}(\log n)$ and diameter at most $(5 \log n + 5) \cdot n^{\frac{1}{\log n}} = 5 \log n + 5 = \mathcal{O}(\log n)$.

Suppose first that the instance of Minimum Cost Hypergraph Layout problem is a weighted all-to-all traffic. It is clear that each edge must be used by some tunnel, hence $n - 1$ is a lower bound on the length of any layout. On the other hand, the hop count is at least $\sum_{d_i \in D} m_i$. In the layout described above, each edge is used at most $\frac{\log n + 1}{2}$ times, and therefore the length of this layout is $\mathcal{O}(n \log n)$. Since the diameter is also $\mathcal{O}(\log n)$, the total hop count is $\mathcal{O}(\log n \cdot \sum_{d_i \in D} m_i)$, yielding an $\mathcal{O}(\log n)$-approximation.

If the instance is not all-to-all, we repeat the argument of the *span* discussed in Section 4.1, obtaining again an $\mathcal{O}(\log n)$-approximation. Summarizing,

Proposition 3 *When the network is a tree, there exists a polynomial-time approximation algorithm for* Minimum Cost Hypergraph Layout *problem with an approximation ratio* $\mathcal{O}(\log n)$.

4.3 General network

In the Minimum Generalized Steiner Network problem, we are given a graph $G = (V, E)$, a weight function $w : E \to \mathbb{N}$, a capacity function $c : E \to \mathbb{N}$, and a requirement function $r : V \times V \to \mathbb{N}$. The objective is to find a *Steiner network* over G that satisfies all the requirements and obeys all the capacities, i.e., a function $f : E \to \mathbb{N}$ such that, for each edge e, $f(e) \leq c(e)$ and, for any pair of nodes i and j, the number of edge disjoint paths between i and j is at least $r(i, j)$, where for each edge e, $f(e)$ copies of e are available. We want to minimize the cost of the network, i.e., $\sum_{e \in E} w(e)f(e)$. The problem is approximable within $\mathcal{O}(\log R)$, where R is the maximum requirement [8], and within a constant factor 2 when all the requirements are equal [9]. The directed version of the problem is approximable within $\mathcal{O}(n^{2/3} \log^{1/3} n)$ [6].

Given an instance of Minimum Cost Hypergraph Layout in a general network, consider the associated Minimum Generalized Steiner Network problem where all the requirements are equal to 1 and where the edge capacities are set to $+\infty$. Let H be an optimal solution to this Minimum Generalized Steiner Network instance (note that H may be disconnected). The following easy observation will be useful: since H is the smallest subgraph of G such that any pair source-destination lies on the same connected component, in any solution to the Minimum Cost Hypergraph Layout problem, the number of edges that are used by at least one tunnel is at least $|E(H)|$. Using the algorithm of [9], we can find in polynomial time a Steiner network H' with $|E(H')| \leq 2|E(H)|$. Since the edge capacities are set to ∞, we can assume that such a Steiner network is a forest. The layout is then obtained by applying the algorithm described in Section 4.2 to each connected component of H'.

It is clear that the hop count of this layout is at most $\mathcal{O}(\log n)$ times the lower bound $\sum_{d_i \in D} m_i$. On the other hand, the length of this layout is $\mathcal{O}(\log n \cdot |E(H')|) = \mathcal{O}(\log n \cdot |E(H)|)$. Since the length of any layout is lower-bounded by $|E(H)|$, the $\mathcal{O}(\log n)$-approximation follows. Summarizing,

Theorem 4 *In a general network, there exists a polynomial-time approximation algorithm for* Minimum Cost Hypergraph Layout *problem with an approximation ratio* $\mathcal{O}(\log n)$.

4.4 Case of a single source

When there is a single source, our problem is closely related to the k-hop Minimum Spanning Tree problem [10]...

5 The Hypergraph Layout Problem on the Path

In this section we focus on the case when the underlying graph is a directed path. Our approach consists in a dynamic programming algorithm that computes partial solutions induced on subpaths of the original path. Let us first give some intuition about this algorithm.

Nodes are numbered from left to right $1, \ldots, n$. Loosely speaking, we use the following dynamic program: we consider a cut vertex i and we look at a local solution induced on the subpath $[1, i]$. That is, the tunnels and traffic located on $[1, i]$. The cost of a local solution is defined as the sum of the local tunnels cost plus the hop counts sum taken on the local traffic.

We introduce then node $i+1$ and the potential tunnels finishing at it (note that if no traffic is directed toward $i+1$, there exists optimal solutions with no tunnel ending at $i+1$). In order to update the local solution cost, it is necessary to have enough information to compute the hop counts once this tunnel is introduced in the solution. Consider now how a tunnel $[j, i+1]$ affects the hop count to reach $i+1$, and remark that it depends only on the hop count from its origin j. So for each source $s \in S$ and vertex x, we introduce $h(s, x)$ defined as the hop count from s to x. Each vertex is then characterized by a hop count vector $h(x)$ whose dimension is the number of sources. A partial solution is then fully encoded by its local cost and the hop counts of all its nodes. Moreover, whenever two nodes get the same hop count vectors, only the rightmost node will be used (if any), since it will be closer to node $i+1$.

It follows from the above discussion that we can encode a partial solution by giving, for each of its hop count vectors, the rightmost node associated to that vector. The dynamic program is therefore as follows: we characterize a solution by a table that contains for each hop count vector h the location of the rightmost vertex that can be reached within $h(s)$ hops from a source s. If we denote by h a bound on the hop count (at most n) and by c a bound (at most n) on the cost of a tunnel, we have $(c^k)^{h^k} = c^{kh^k}$ such possible table entries.

By making an error of ε on the two costs (length and hops), we can encode the logarithm in base $1+\varepsilon$ of those quantities, which lead to tables of size $\Theta((\log n)^{k \log n})$. Note that this running time is already subexponential, so the problem is unlikely to be NP-hard to approximate within a constant factor when the number of sources is bounded (because it is widely assumed that algorithms solving 3-SAT require $2^{\Theta(n)}$ time). We shall see now how to improve this first naïve dynamic program.

Refining the dynamic program. Since tunnels are directed hyperedges, whenever a solution uses for node $i+1$ a tunnel going beyond (i.e. to the right) of a node that a source s can reach in h hops, node i will be reachable within $h+1$ hops from source s. From this observation it follows that in order to update a local solution we only need to know where the rightmost node at distance h from source s is located. This leads to tables of size C^{kH} in the exact case and $\Theta((\log n)^{k \log n})$ if we the use an approximation.

We proceed now to give all the details for one and two sources, that suffice to get the intuition for an arbitrary number k of sources.

5.1 Case of a single source and the non crossing property

We summarize the algorithm that appeared first in [1]. In the case of a single source, it is not difficult to see that the tunnel structure is *non crossing*, i.e. two tunnels can only intersect in an optimal solution if one is strictly inside the other [1]. This leads to the following approach:

- We consider the rightmost tunnel originating from the source and assume it ends at i.
- Clearly any tunnel ending in $[i, n]$ and starting in $[1, i-1]$ can be replaced by a tunnel starting at i, this may only decrease the hop count and the length (for this we need only the length function to be increasing).

This approach allow to compute the optimal for a path with n vertices inductively.

- We denote $C([1, i])$ the minimum cost for the instance restricted to the i first nodes.
- We denote $C[j, n]$ the minimum cost for the instace restricted to the last nodes in which the source is replaced by node j.

Then:

$$C(i) = Min\{k < i, C(k) + w(T_{[1,k]}) + c([1, k]) + d([k + 1, i]) + C([k, n])\}$$

In the particular case of the broadcast $d_i = 1, a_i = 0, b_i = i$ with a cost for the tunnels equal to their length, we denote $OPT(i)$ the minimum cost for a solution on the path with i nodes. The above equation reduces to:

$$C(i) = Min\{k < i, Opt(k) + (k - 1) + n - k - 1 + Opt(n - k - 1)\}$$

5.2 Case of two sources

We use a dynamic program similar to the one used for the single source case. We denote $s_0 = 0$ the lefmost source and s_1 the other. The two sources dynamic program is slightly more complicated. In order to solve it we introduce an auxilliary problem with *pseudo sources*. A pseudo source (h_0, h_1, l) represents a node attached by a path of length l to the path and from which one can reach $s_i, i = 0, 1$ in h_i hops In the induction the following auxilliary problem will appear:

- The traffic is restricted on the interval $[u, v]$ where weither u or v in an end of the original path.
- There is one or two pseudo sources located at the left end of $[u, v]$.
- If there are two pseudo sources they are labeled (j, j, l_0) and $(j+1, j, l_1)$ and we denote the problem $P((j, j), (j + 1, j), l_0, l_1, [u, v])$

- If there a single pseudo source it is labeled (j,k) and we denote the problem $P((j,k),[u,v])$

In both cases we denote $OPT()$ the value of the optimal solution. Note that $P((0,0),[u,v])$ is indeed a single source problem in which a uniuqe source replace both s_0 and s_1. Moreover $P((j,k),[u,v])$ is equivalent to a single source problem since $OPT(P(j,k),[u,v]) = OPT(P(0,0,[u,v]) + \sum_{x\in[u,v]} jt(x,s_0) + kt(x,s_1))$.

We now relate the two sources problem to the auxilliary problem:

We consider the rightmost tunnel having $s_i, i=0,1$ as head and denote E_i its end node. We compute the optimal solution conditionned on those two constraints (i,e the values $E_i, i=0,1$. There are 3 cases to consider (A,B,C see figure 5.2) , we first deal with two "easy ones".

- A) If E_0 is left to s_1. Then on the subpath $[E_0,n]$ we pick an optimal solution with a slightly modified instance: we leave traffic demands toward s_1 unchanged and we replace the source s_0 by a pseudo source at E_0 with hop count 1. On the subpath $[0,E_0-1]$ we use an optimal solution (note that in this subproblem there is only one source).
- B) If E_0 is right to s_1 with E_0 right from E_1. Then E_0 is at distance 1 from both sources and hencefore any pipe entering $[E_0+1,n]$ can be assumed to start at E_0. So the optimal solution is then obtained by using $OPT([0,E_0])$

In both cases the induction is valid because E_0 is in each case the best node too start a tunnel going to its right, starting at E_0 is cheaper and no node closer to the sources can be reached (from the definition of E_0).

The main case (C). We now study the case (C) in which E_1 is right from E_0. Note that E_0 is a $(1,1)$ pseudo-source while E_1 is a $(1,2)$ pseudo-source. Consider a tunnel ending in $]E_1,n]$, the situation is more complicated that in the single source case since E_1 (a $(1,2)$ node) is not anymore the "best" possible node. The only nodes that can beat E_1 are $(1,1)$ nodes and E_0 is the rightmost one. So we can assume that such a tunnel is starting either at E_0 or at E_1. Indeed we have two "best nodes". To perform the induction we have to solve two subproblems:

- the right subproblem on $[0,E_0-1]$ but under a conditionning on the location of the rightmost tunnel from s_1. i.e $Opt([0,E_0-1] \mid (s_1,E_1))$.
- the left subproblem in which we have two pseudo sources $E_0:(1,1)$ and $E_1:(2,1)$. So we pay $Opt(P((1,1),(2,1),l(E_1,E_0),l(E_1,E_1+1),[E_1+1,n]))$

To complete our algorithm we need to show how to compute the dynamic program tables inductively. We miss only the rules to compute $Opt(P((j,j),(j+1,j),l_0,l_1,[u,v]))$.

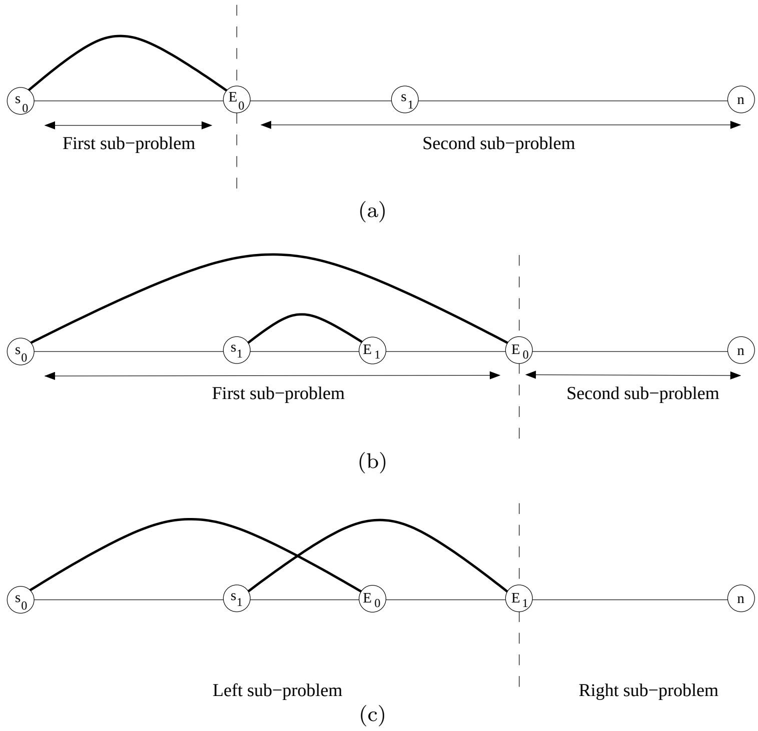

Figure 3: Dynamic programming with two sources: cases A, B, and C, respectively.

The two pseudo sources tables. Let us denote s_0, s_1 the two pseudo sources, the induction is again on the two rightmost nodes (E_0, E_1). It is almost the same as the one we used in the case of two sources, but case A cannot occur since both pseudo sources are located outside the path.

- B) If E_0 right from E_1. Then E_0 is at distance $J+1$ from both sources and hencefore any pipe entering $[E_0+1, n]$ can be assumed to start at E_0. So the optimal solution is then obtained by using $OPT([0, E_0])$ for the right subproblem and $Opt(P((j,j),(j+1,j),l_0,l_1,[0,E_0-1]))$.
- C) IF E_0 is left from E_1. The situation is similar to the C case for two sources, we can split the problem into left and right subproblems the left being a two pseudo-sources problems reduced to $[u, E_1-1]$ with a condition on the rightmost tunnel from s_0 and the other being a single source problem on $E_1+1, n]$.

Correctness & Complexity

To complete the proof, we must explain how the above induction allow to compute all the tables inductively. here are some explanations:

- first the induction is performed on the length of the path and when the tables for $[u, v]$ are computed all the tables for strict subsegments of $[u, v]$ are known.
- Second when filling the new tables we compute the cost in a consistent way: That is we sum the cost of the left and right subproblems (found in already computed tables) with the cost of the tunnels that are removed and the hop count for traffic toward node that are removed (either E_0or E_1).
- As always we keep only the best cost found when examining all the subcases (A,B,C)
- Last (and this may be the only unclear feature) one may worry about the conditioning on the rightmost tunnel that appear in cases (C). But this never lead to condition on an unbounded number of tunnel since in the induction those rightmost tunnel either disappear or stay.

To evaluate the complexity we use a pessimistic bound on the table size. $Opt(P((j, j), (j+1, j), l_0, l_1, [u, v]))$. The value l_0, l_1 are not pseudo polynomial since they are in bijection with the pseudo sources locations, $j \in [0, n]$ and since $[u, v]$ is either an end or and head segment we can store it in space $2n$. So we get size $\Theta(n^4)$ for the tables, if we add the conditioning on the rightmost tunnel from the rightmost source we get $\Theta(n^5)$.

To improve the complexity we can use classical scaling technics to get space $\frac{\log n}{\varepsilon}^5$ and approximation $1 + \varepsilon$.

References

[1] J.-C. Bermond, D. Coudert, J. Moulierac, S. Pérennes, H. Rivano, I. Sau, and F. Solano. MPLS label stacking on the line network. Technical Report RR-6803, INRIA, 2009.

[2] J.-C. Bermond, D. Coudert, J. Moulierac, S. Perennes, H. Rivano, I. Sau, and F. Solano Donado. MPLS label stacking on the line network. In *Proceedings of IFIP Networking*, Aachen, Germany, May 2009. Research Report INRIA RR-6803 accesible at `http://hal.inria.fr/inria-00354267`.

[3] J.-C. Bermond, N. Marlin, D. Peleg, and S. Pérennes. Directed virtual path layouts in ATM networks. *Theoretical Computer Science*, 291(1):3–28, 2003.

[4] M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32:171–176, 1989.

[5] S. Bhatnagar, S. Ganguly, and B. Nath. Creating Multipoint-to-Point LSPs for traffic engineering. *IEEE Commun. Mag.*, 43(1):95–100, Jan. 2005.

[6] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed Steiner problems. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 192–200, 1998.

[7] O. Gerstel, A. Wool, and S. Zaks. Optimal layouts on a chain ATM network. *Discrete Applied Mathematics*, 83:157–178, 1998.

[8] M. X. Goemans, A. V. Goldberg, S. Plotkin, D. B. Shmoys, E. Tardos, and D. P. Williamson. Improved approximation algorithms for network design problems. In *Proceedings of the 5th annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 223–232, 1994.

[9] S. Khuller and U. Vishkin. Biconnectivity approximations and graph carvings. *Journal of the ACM*, 41:214–235, 1994.

[10] S. Laue and D. Matijevic. Approximating k-hop minimum spanning trees in euclidean metrics.

[11] F. Ramos et al. IST-LASAGNE: Towards all-optical label swapping employing optical logic gates and optical flip-flops. *IEEE J. Sel. Areas Commun.*, 23(10):2993–3011, Oct. 2005.

[12] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th annual ACM Symposium on Theory of Computing (STOC)*, pages 475–484, 1997.

[13] H. Saito, Y. Miyao, and M. Yoshida. Traffic engineering using multiple MultiPoint-to-Point LSPs. In *Proc. IEEE INFOCOM 2000*, pages 894–901, 2000.

[14] F. Solano, R. V. Caenegem, D. Colle, J. L. Marzo, M. Pickavet, R. Fabregat, and P. Demeester. All-optical label stacking: Easing the trade-offs between routing and architecture cost in all-optical packet switching. In *IEEE Conference on Computer Communications (Infocom 08)*, pages 655–663, Phoenix, AZ, USA, Apr. 2008.

[15] F. Solano, R. Fabregat, and J. Marzo. On optimal computation of MPLS label binding for MultiPoint-to-Point connections. *IEEE Trans. Commun.*, 56(7):1056–1059, July 2007.

[16] F. Solano and J. Moulierac. Routing in All-Optical Label Switched-based Networks with Small Label Spaces. In *13th Conference on Optical Network Design and Modeling (ONDM)*, Feb. 2009.

[17] F. Solano, T. Stidsen, R. Fabregat, and J. Marzo. Label space reduction in MPLS networks: How much can one label do? *IEEE/ACM Trans. Netw.*, Feb. 2009.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399