Designing Hypergraph Layouts to GMPLS Routing Strategies

Jean-Claude Bermond, David Coudert,
Joanna Moulierac, Stéphane Pérennes, **Ignasi Sau**
INRIA/CNRS/UNSA, Sophia-Antipolis, France
Fernando Solano
Warsaw University of Technology, Poland

ignasi.sau@gmail.com

SIROCCO 2009
27th of May 2009, Piran, Slovenia

Outline

Outline

1. Introduction
   - Concepts and Motivations
   - The Label Stack
2. Model
3. Hardness Results
4. Approximation Algorithms
5. The Case of the Path
6. Conclusions

Concepts in brief... GMPLS/AOPS

About GMPLS switching...

- GMPLS = Generic MultiProtocol Label Switching.
- G/MPLS is a tag-switching technology (packet-based networks).
- Each packet is tagged (labeled), so it can be associated and treated as a single flow.
- Packet forwarding is based on the content of the label solely.

Concepts in brief... GMPLS/AOPS

About GMPLS switching...

- GMPLS = Generic MultiProtocol Label Switching.
- G/MPLS is a tag-switching technology (packet-based networks).
- Each packet is tagged (labeled), so it can be associated and treated as a single flow.
- Packet forwarding is based on the content of the label solely.

About AOPS...

All-Optical Packet Switching (AOPS) is an all-optical hardware implementation of GMPLS switching for packet forwarding.

1. Label processing and packet forwarding decisions are all performed completely optically
2. No need for OEO regeneration... faster packet forwarding.
3. *One 'decoding' device is needed for each used label at each node*
4. Few labels...

Concepts in brief... GMPLS/AOPS

About GMPLS switching...

- GMPLS = Generic MultiProtocol Label Switching.
- G/MPLS is a tag-switching technology (packet-based networks).
- Each packet is tagged (labeled), so it can be associated and treated as a single flow.
- Packet forwarding is based on the content of the label solely.

About AOPS...

All-Optical Packet Switching (AOPS) is an all-optical hardware implementation of GMPLS switching for packet forwarding.

1. Label processing and packet forwarding decisions are all performed completely optically
2. No need for OEO regeneration... faster packet forwarding.
3. *One 'decoding' device is needed for each used label at each node*
4. Few labels... Labels are extremely valuable resources

Outline

1. Introduction
   - Concepts and Motivations
   - The Label Stack
2. Model
3. Hardness Results
4. Approximation Algorithms
5. The Case of the Path
6. Conclusions

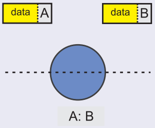
Handling Labels

Packets contain a stack of labels...
Nodes may alter the stack of labels performing one of the following operations:

Handling Labels

Packets contain a stack of labels...
Nodes may alter the stack of labels performing one of the following operations:

Swap the Top

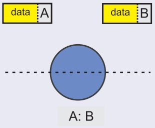


... with the neighbor's locally 'understandable' label.

Handling Labels

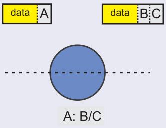
Packets contain a stack of labels...
Nodes may alter the stack of labels performing one of the following operations:

Swap the Top



... with the neighbor's locally 'understandable' label.

Swap & Push

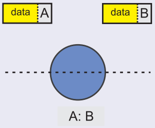


... as many as you want.

Handling Labels

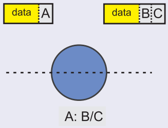
Packets contain a stack of labels...
Nodes may alter the stack of labels performing one of the following operations:

Swap the Top



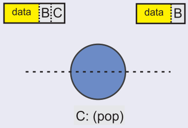
... with the neighbor's locally 'understandable' label.

Swap & Push



... as many as you want.

Pop the Stack

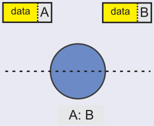


... only one.

Handling Labels

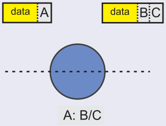
Packets contain a stack of labels...
Nodes may alter the stack of labels performing one of the following operations:

Swap the Top



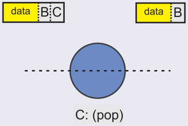
... with the neighbor's locally 'understandable' label.

Swap & Push



... as many as you want.

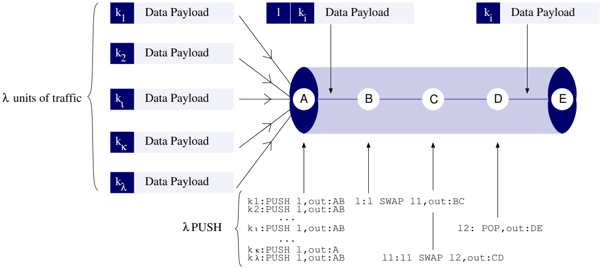
Pop the Stack



... only one.

... but solely the top one can be processed!

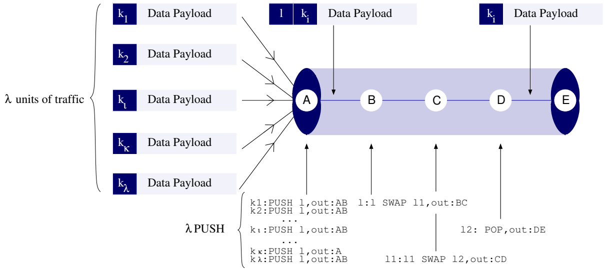
Label Stacking & Tunnels



By pushing a label, we can create a higher hierarchy LSP (Label Switched Path), called TUNNEL.

- Using a stack size of 2, the number of used labels can be decreased
- The larger the stack, fewer labels are needed...

Label Stacking & Tunnels



By pushing a label, we can create a higher hierarchy LSP (Label Switched Path), called TUNNEL.

- Using a stack size of 2, the number of used labels can be decreased
- The larger the stack, fewer labels are needed...

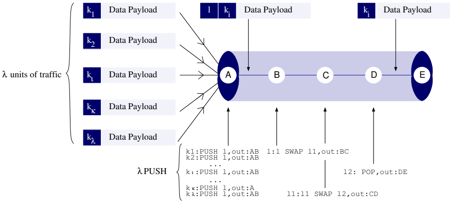
The problem we tackle is the minimization of the total number of labels using tunnels.

Outline

Some notation

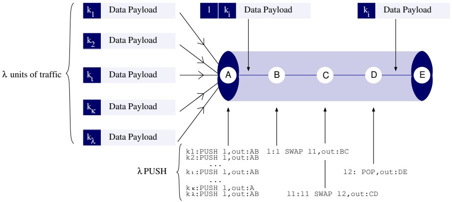
- $G = (V, E)$ is the underlying digraph (which can be symmetric or not).
- $|V| = n$, and vertices are numbered $1, \ldots, n$.
- r_{ij} is the request from $i \in V$ to $j \in V$, with multiplicity m_{ij}. R is the set of all requests.
- $P(G)$ is the set of all simple dipaths in G.
- t stands for a tunnel, and T is the set of tunnels, that is $t \in T \subseteq P(G)$.
- ℓ is a length function on the arcs, that is $\ell : E \rightarrow \mathbb{R}^+$.
- 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.

Cost of a tunnel = number of labels



Let t be a tunnel.

Cost of a tunnel = number of labels



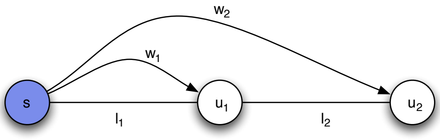
Let t be a tunnel. Its cost is defined as

$$c(t) = w(t) + (\ell(t) - 1).$$

- $w(t)$ is the number of forwarded traffic units (LSPs)
- $\ell(t)$ is the length of the tunnel (usually, the number of hops)

Cost Example - Scenario

We consider the case of a line network with one source and multiple destinations...

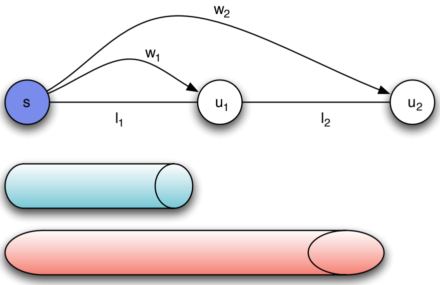


With no tunnels we need $l_1 \cdot (w_1 + w_2) + l_2 \cdot w_2$ labels...

With tunnels?? The optimal solution depends on the values of l_i and w_i...

Cost Example - Solutions

First Solution...

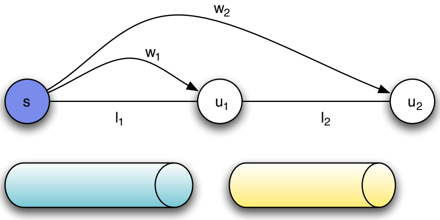


$$c(T_{(s,u_1)}) = w_1 + l_1 - 1$$
$$c(T_{(s,u_2)}) = w_2 + (l_1 + l_2) - 1$$

Total cost is: $w_1 + w_2 + 2l_1 + l_2 - 2$

Cost Example - Solutions

Second Solution...

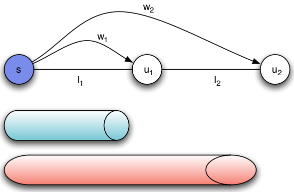


$$c(T_{(s,u_1)}) = w_1 + w_2 + l_1 - 1$$
$$c(T_{(u_1,u_2)}) = w_2 + l_2 - 1$$

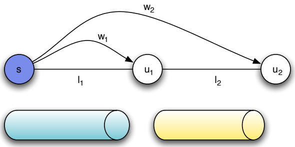
Total cost is: $w_1 + 2w_2 + l_1 + l_2 - 2$

Cost Example - Solutions

Which one is the best solution?

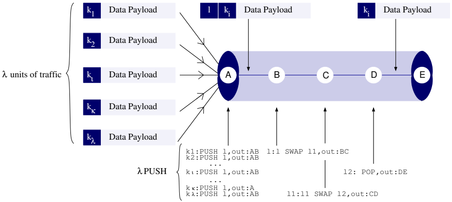


If $l_1 \leq w_2$



If $l_1 \geq w_2$

Cost of a tunnel = number of labels (II)

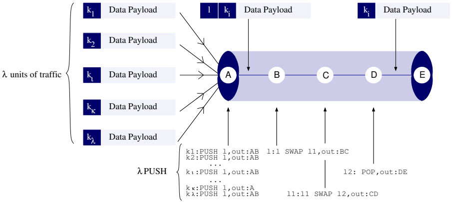


Let t be a tunnel. Its cost is defined as

$$c(t) = w(t) + (\ell(t) - 1).$$

- $w(t)$ is the number of forwarded traffic units (LSPs)
- $\ell(t)$ is the length of the tunnel (usually, the number of hops)

Cost of a tunnel = number of labels (II)



Let t be a tunnel. Its cost is defined as

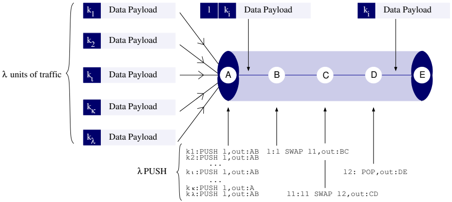
$$c(t) = w(t) + (\ell(t) - 1).$$

- $w(t)$ is the number of forwarded traffic units (LSPs)
- $\ell(t)$ is the length of the tunnel (usually, the number of hops)

The cost of a set of tunnels T is

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

Cost of a tunnel = number of labels (II)



Let t be a tunnel. Its cost is defined as

$$c(t) = w(t) + (\ell(t) - 1).$$

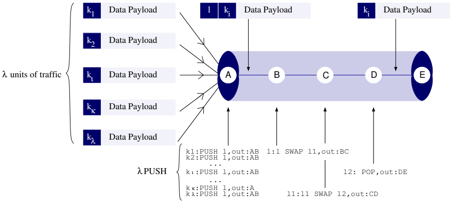
- $w(t)$ is the number of forwarded traffic units (LSPs)
- $\ell(t)$ is the length of the tunnel (usually, the number of hops)

The cost of a set of tunnels T is

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

Each tunnel can be seen as a directed hyperarc on the vertex set of G.

Cost of a tunnel = number of labels (II)



Let t be a tunnel. Its cost is defined as

$$c(t) = w(t) + (\ell(t) - 1).$$

- $w(t)$ is the number of forwarded traffic units (LSPs)
- $\ell(t)$ is the length of the tunnel (usually, the number of hops)

The cost of a set of tunnels T is

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

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

Hypergraph layout

Definition (Hypergraph layout)

Given a graph G and a set $T \subseteq P(G)$ of dipaths, the associated hypergraph layout $H(T)$ 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 hyperarc in $H(T)$ connecting any vertex of t to the end of t.

Hypergraph layout

Definition (Hypergraph layout)

Given a graph G and a set $T \subseteq P(G)$ of dipaths, the associated hypergraph layout $H(T)$ 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 hyperarc in $H(T)$ connecting any vertex of t to the end of t.

- Note that a hypergraph $H(T)$ defines a virtual topology on G.

Hypergraph layout

Definition (Hypergraph layout)

Given a graph G and a set $T \subseteq P(G)$ of dipaths, the associated hypergraph layout $H(T)$ 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 hyperarc in $H(T)$ connecting any vertex of t to the end of t.

- Note that a hypergraph $H(T)$ defines a virtual topology on G.
- A hypergraph layout $H(T)$ is said to be feasible if for each request $r_{ij} \in R$ there exists a dipath in $H(T)$ from i to j.

Hypergraph layout

Definition (Hypergraph layout)

Given a graph G and a set $T \subseteq P(G)$ of dipaths, the associated hypergraph layout $H(T)$ 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 hyperarc in $H(T)$ connecting any vertex of t to the end of t.

- Note that a hypergraph $H(T)$ defines a virtual topology on G.
- A hypergraph layout $H(T)$ is said to be feasible if for each request $r_{ij} \in R$ there exists a dipath in $H(T)$ from i to j.
- The problem can then be simply expressed as finding a **feasible hypergraph layout of minimum cost**.

Simplifying the cost function

The cost of a set of tunnels T was

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

Simplifying the cost function

The cost of a set of tunnels T was

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

Given a hypergraph layout $H(T)$,

- let $L(r_{ij})$ be the number of hyperarcs that request r_{ij} uses, and

Simplifying the cost function

The cost of a set of tunnels T was

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

Given a hypergraph layout $H(T)$,

- let $L(r_{ij})$ be the number of hyperarcs that request r_{ij} uses, and
- let $d_H(i, j)$ be the distance from vertex i to vertex j in $H(T)$.

Simplifying the cost function

The cost of a set of tunnels T was

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

Given a hypergraph layout $H(T)$,

- let $L(r_{ij})$ be the number of hyperarcs that request r_{ij} uses, and
- let $d_H(i,j)$ be the distance from vertex i to vertex j in $H(T)$.

Then the term $\sum_{t \in T} w(t)$ of Equation (1) can be rewritten as

$$\sum_{r_{ij} \in R} L(r_{ij}) \cdot m_{ij}\,.$$

Simplifying the cost function

The cost of a set of tunnels T was

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

Given a hypergraph layout $H(T)$,

- let $L(r_{ij})$ be the number of hyperarcs that request r_{ij} uses, and
- let $d_H(i,j)$ be the distance from vertex i to vertex j in $H(T)$.

Then the term $\sum_{t \in T} w(t)$ of Equation (1) can be rewritten as

$$\sum_{r_{ij} \in R} L(r_{ij}) \cdot m_{ij}\,.$$

Since $L(r_{ij}) \geq d_H(i,j)$, we conclude that **in an optimal solution the routing necessarily uses shortest dipaths in the hypergraph layout**.

Statement of the problem

It follows that the cost function can be rewritten w.l.o.g. as

$$\underbrace{\sum_{t \in T} (\ell(t) - 1)}_{\text{total length}} + \underbrace{\sum_{r_{ij} \in R} d_H(i,j) m_{ij}}_{\text{hop count}} . \qquad (2)$$

Statement of the problem

It follows that the cost function can be rewritten w.l.o.g. as

$$\underbrace{\sum_{t \in T} (\ell(t) - 1)}_{\text{total length}} + \underbrace{\sum_{r_{ij} \in R} d_H(i,j) m_{ij}}_{\text{hop count}} . \tag{2}$$

MINIMUM COST HYPERGRAPH LAYOUT

- Input: A digraph $G = (V, E)$ with a length function $\ell : E \to \mathbb{R}^+$, and a set R of traffic requests.
- Output: A feasible hypergraph layout of minimum cost, where the cost of a hypergraph layout is defined as in Equation (2).

Statement of the problem

It follows that the cost function can be rewritten w.l.o.g. as

$$\underbrace{\sum_{t\in T}(\ell(t)-1)}_{\text{total length}}+\underbrace{\sum_{r_{ij}\in R} d_H(i,j)m_{ij}}_{\text{hop count}}\,. \qquad (2)$$

MINIMUM COST HYPERGRAPH LAYOUT

- Input: A digraph $G=(V,E)$ with a length function $\ell: E\longrightarrow \mathbb{R}^+$, and a set R of traffic requests.
- Output: A feasible hypergraph layout of minimum cost, where the cost of a hypergraph layout is defined as in Equation (2).

If G is a **symmetric** digraph, the problem is denoted MINIMUM COST SYMMETRIC HYPERGRAPH LAYOUT.

Outline

General (directed) network

Theorem

The Minimum Cost 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* $\mathrm{P} = \mathrm{NP}$.

[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 $\mathcal{S}$.

General (directed) network

Theorem

The MINIMUM COST 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* $\mathrm{P} = \mathrm{NP}$.

- The reduction is from MINIMUM SET COVER[1].

[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 $\mathcal{S}$.

General (directed) network

Theorem

The MINIMUM COST 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* $\mathrm{P} = \mathrm{NP}$.

- The reduction is from MINIMUM SET COVER[1].
- MINIMUM SET COVER is not approximable within a factor $C \log n$, for some constant $C > 0$, unless $\mathrm{P} = \mathrm{NP}$.
  [Raz and Safra, STOC 1997]

[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 $\mathcal{S}$.

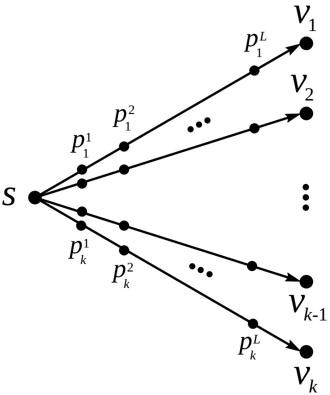
Idea of the reduction

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.

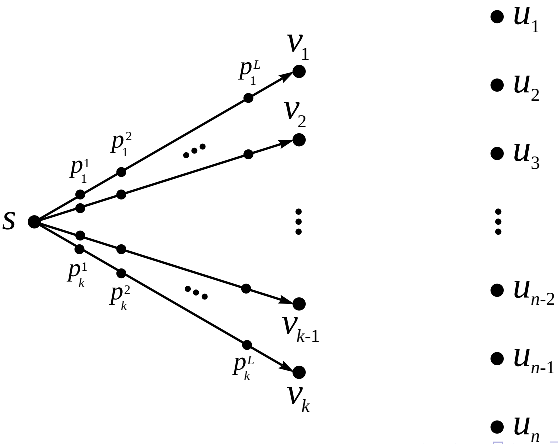
Idea of the reduction (II)

- For each set S_i we introduce a node v_i and a directed path of length $L+1$ (L is a big constant) from s to v_i through L new vertices $p_i^1, p_i^2, \ldots, p_i^L$.



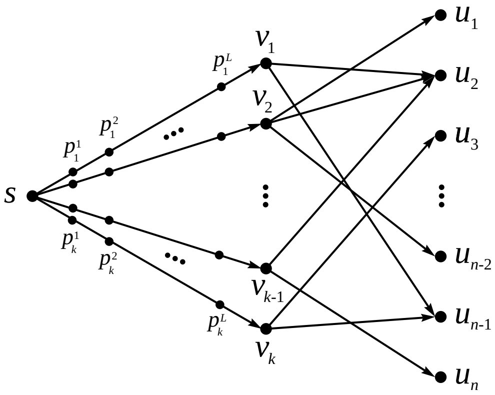
Idea of the reduction (III)

- 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 $a_j \in S_i$.



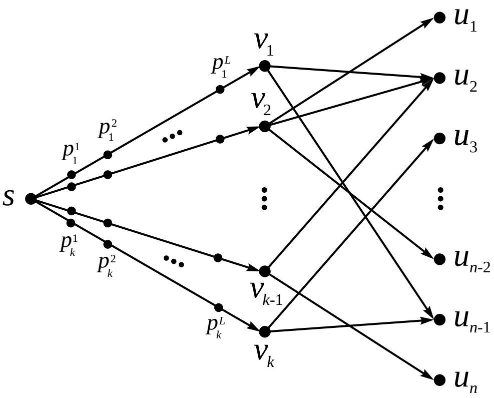
Idea of the reduction (IV)

- 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 $a_j \in S_i$.



Idea of the reduction (IV)

- 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 $i = 1, \ldots, n$.

Outline

Symmetric network

Theorem

The MINIMUM COST SYMMETRIC HYPERGRAPH LAYOUT *problem is* APX*-hard even if the instance is a partial broadcast. Therefore, it does not accept a* PTAS *unless* P=NP*.*

[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.

Symmetric network

Theorem

The MINIMUM COST SYMMETRIC HYPERGRAPH LAYOUT *problem is* APX*-hard even if the instance is a partial broadcast. Therefore, it does not accept a* PTAS *unless* P=NP*.*

- The reduction is from MINIMUM STEINER TREE[2].
- MINIMUM STEINER TREE is APX-hard, hence it does not accept a PTAS unless P = NP.
  [Bern and Plassmann, IPL 1989]

[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.

Outline

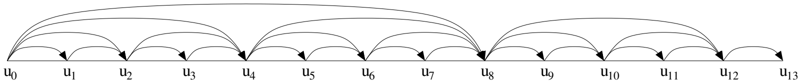
Case of the path

Proposition

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

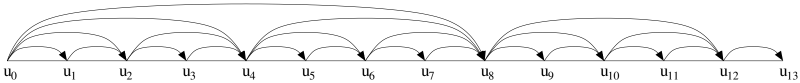
Idea: 2^k-hops long tunnels

We consider tunnels of length a power of 2.



Idea: 2^k-hops long tunnels

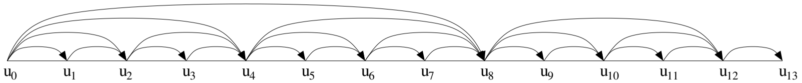
We consider tunnels of length a power of 2.



.... then, we route the demands using shortest paths in this layout.

Idea: 2^k-hops long tunnels

We consider tunnels of length a power of 2.



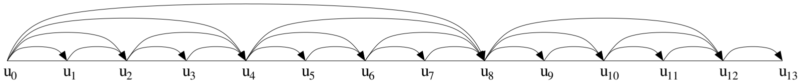
.... then, we route the demands using shortest paths in this layout.

In this layout:

- Total length: $\mathcal{O}(n \log n)$.
- Hop count: each request can be routed in $\mathcal{O}(\log n)$ hops.

Idea: 2^k-hops long tunnels

We consider tunnels of length a power of 2.



.... then, we route the demands using shortest paths in this layout.

In this layout:

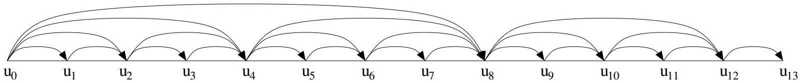
- Total length: $\mathcal{O}(n \log n)$.
- Hop count: each request can be routed in $\mathcal{O}(\log n)$ hops.

In any solution:

- Total length $\geq n$.
- Hop count $\geq \sum m_{ij}$.

Idea: 2^k-hops long tunnels

We consider tunnels of length a power of 2.



.... then, we route the demands using shortest paths in this layout.

In this layout:

- Total length: $\mathcal{O}(n \log n)$.
- Hop count: each request can be routed in $\mathcal{O}(\log n)$ hops.

In any solution:

- Total length $\geq n$.
- Hop count $\geq \sum m_{ij}$.

$\Rightarrow$ $\mathcal{O}(\log n)$-approximation algorithm.

Outline

Case of the tree

Theorem (Bermond, Marlin, Peleg, and Pérennes, TCS 2003)

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.*

Case of the tree

Theorem (Bermond, Marlin, Peleg, and Pérennes, TCS 2003)

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 $c = \frac{\log n + 1}{2}$, the above implies that we can find in polynomial time a layout with load $\mathcal{O}(\log n)$ and diameter at most $(5 \log n + 5) \cdot n^{\frac{1}{\log n}} = 10 \log n + 10 = \mathcal{O}(\log n)$.

Case of the tree

Theorem (Bermond, Marlin, Peleg, and Pérennes, TCS 2003)

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 $c = \frac{\log n+1}{2}$, the above implies that we can find in polynomial time a layout with load $\mathcal{O}(\log n)$ and diameter at most $(5 \log n + 5) \cdot n^{\frac{1}{\log n}} = 10 \log n + 10 = \mathcal{O}(\log n)$.

Proposition

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

Outline

General graph

Theorem

In a general network, there exists a polynomial-time approximation algorithm for MINIMUM COST HYPERGRAPH LAYOUT *problem with an approximation ratio* $\mathcal{O}(\log n)$.

General graph

Theorem

In a general network, there exists a polynomial-time approximation algorithm for MINIMUM COST HYPERGRAPH LAYOUT *problem with an approximation ratio $\mathcal{O}(\log n)$.*

- **Idea**: We find a MINIMUM GENERALIZED STEINER FOREST H.

  This problem can be approximated within a **constant factor** 2. [Khuller and Vishkin, Journal of the ACM 1994]

General graph

Theorem

In a general network, there exists a polynomial-time approximation algorithm for MINIMUM COST HYPERGRAPH LAYOUT *problem with an approximation ratio* $\mathcal{O}(\log n)$.

- **Idea**: We find a MINIMUM GENERALIZED STEINER FOREST H.

  This problem can be approximated within a **constant factor** 2. [Khuller and Vishkin, Journal of the ACM 1994]

- The we apply the algorithm for the tree to each connected component of $H \Rightarrow$ overall approximation factor $\mathcal{O}(\log n)$.

The case of the path with bounded number of sources

- For a **single source**, a polynomial optimal dynamic programming algorithm has been presented in [Bermond, Coudert, Moulierac, Pérennes, Rivano, and Sau, Networking 2009]

The case of the path with bounded number of sources

- For a **single source**, a polynomial optimal dynamic programming algorithm has been presented in [Bermond, Coudert, Moulierac, Pérennes, Rivano, and Sau, Networking 2009]
- Here we extended the dynamic programming algorithm for any fixed number k of sources, with a running time of $n^{\mathcal{O}(k)}$.

Conclusions

- We modeled a problem raised by label minimization in GMPLS networks as a hypergraph layout problem.
- The problem seems closely related to classical VPL problems.
- We provided hardness results and approximations algorithms.
- Also, we proved that the problem is polynomial on the path for any bounded number of sources.

Conclusions

- We modeled a problem raised by label minimization in GMPLS networks as a hypergraph layout problem.
- The problem seems closely related to classical VPL problems.
- We provided hardness results and approximations algorithms.
- Also, we proved that the problem is polynomial on the path for any bounded number of sources.

A lot of work to be done:

- Improve the hardness results and the approximation algorithms.

Conclusions

- We modeled a problem raised by label minimization in GMPLS networks as a hypergraph layout problem.
- The problem seems closely related to classical VPL problems.
- We provided hardness results and approximations algorithms.
- Also, we proved that the problem is polynomial on the path for any bounded number of sources.

A lot of work to be done:

- Improve the hardness results and the approximation algorithms.
- Is the problem polynomial on the path for **unbounded** number of sources?

Conclusions

- We modeled a problem raised by label minimization in GMPLS networks as a hypergraph layout problem.
- The problem seems closely related to classical VPL problems.
- We provided hardness results and approximations algorithms.
- Also, we proved that the problem is polynomial on the path for any bounded number of sources.

A lot of work to be done:

- Improve the hardness results and the approximation algorithms.
- Is the problem polynomial on the path for **unbounded** number of sources?
- More generally, is the problem polynomial on trees or graphs of bounded treewidth?

Conclusions

- We modeled a problem raised by label minimization in GMPLS networks as a hypergraph layout problem.
- The problem seems closely related to classical VPL problems.
- We provided hardness results and approximations algorithms.
- Also, we proved that the problem is polynomial on the path for any bounded number of sources.

A lot of work to be done:

- Improve the hardness results and the approximation algorithms.
- Is the problem polynomial on the path for **unbounded** number of sources?
- More generally, is the problem polynomial on trees or graphs of bounded treewidth?
- Does the problem remain NP-hard if the **routes** to be followed by the requests are part of the **input**?

Thanks!

Questions?