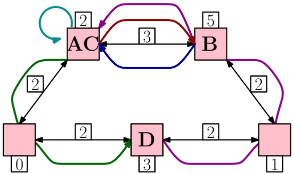
Approximating the Virtual Network Embedding Problem: Theory and Practice



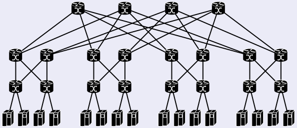
23rd International Symposium on Mathematical Programming 2018
Bordeaux, France

Matthias Rost
Technische Universität Berlin, Internet Network Architectures

Stefan Schmid
Universität Wien, Communication Technologies

A Short Introduction to the Virtual Network Embedding Problem

Operators offer their Network Resources



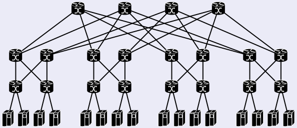
Data Center Network

Wide-Area Network

Substrate (Physical Network)

- Directed graph $G_S = (V_S, E_S)$
- Capacities $c_S : G_S \to \mathbb{R}_{\geq 0}$

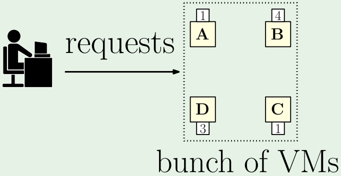
Operators offer their Network Resources



Substrate (Physical Network)

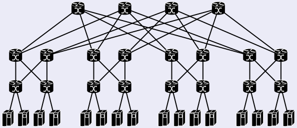
- Directed graph $G_S = (V_S, E_S)$
- Capacities $c_S : G_S \to \mathbb{R}_{\geq 0}$

'Classic' Cloud Computing



- User requests virtual machines
- No guarantee on network performance

Operators offer their Network Resources



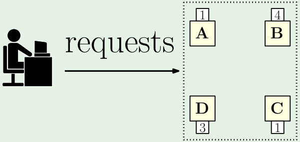
Data Center Network

Wide-Area Network

Substrate (Physical Network)

- Directed graph $G_S = (V_S, E_S)$
- Capacities $c_S : G_S \to \mathbb{R}_{\geq 0}$

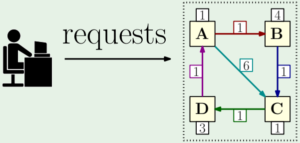
'Classic' Cloud Computing



bunch of VMs

- User requests virtual machines
- No guarantee on network performance

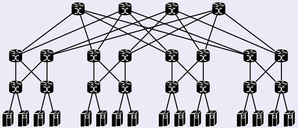
Goal: Virtual Networks (since ≈ 2006)



Virtual Network

- Communication requirements given
- Network performance will be guaranteed

Operators offer their Network Resources



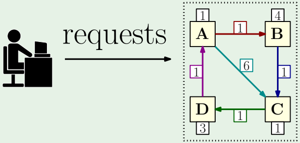
Data Center Network

Wide-Area Network

Substrate (Physical Network)

- Directed graph $G_S = (V_S, E_S)$
- Capacities $c_S : G_S \to \mathbb{R}_{\geq 0}$

Goal: Virtual Networks (since $\approx$ 2006)

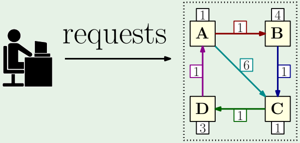


Virtual Network

Virtual Network Request $G_r = (V_r, E_r)$

- demands $d_r : G_r \to \mathbb{R}_{\geq 0}$
- mapping restrictions
  - $V_S^i \subseteq V_S$ for $i \in V_r$
  - $E_S^{i,j} \subseteq E_S$ for $(i,j) \in E_r$

Goal: Virtual Networks (since $\approx$ 2006)



Virtual Network

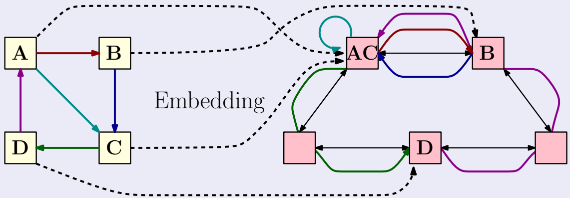
Virtual Network Request $G_r = (V_r, E_r)$

- demands $d_r : G_r \to \mathbb{R}_{\geq 0}$
- mapping restrictions
  - $V_S^i \subseteq V_S$ for $i \in V_r$
  - $E_S^{i,j} \subseteq E_S$ for $(i,j) \in E_r$

Valid Mapping

Virtual Network

Substrate (Physical Network)



Def: Valid mapping $m_r = (m_V, m_E)$ ...

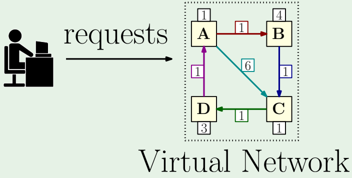
$m_V : V_r \to V_S$ and $m_E : E_r \to \mathcal{P}(E_S)$ satisfies

valid connectivity: $m_V(i) \overset{m_E(i,j)}{\rightsquigarrow} m_V(j)$

valid node mapping: $m_V(i) \in V_S^i$

valid edge mapping: $m_E(i,j) \subseteq E_S^{i,j}$

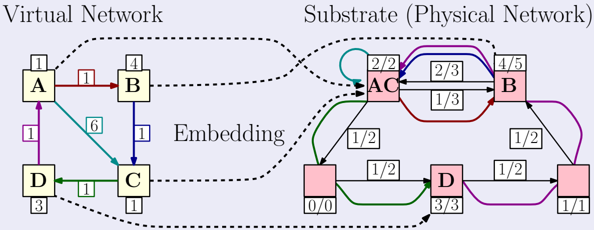
Goal: Virtual Networks (since ≈ 2006)



Virtual Network Request $G_r = (V_r, E_r)$

- demands $d_r : G_r \to \mathbb{R}_{\geq 0}$
- mapping restrictions
  - $V_S^i \subseteq V_S$ for $i \in V_r$
  - $E_S^{i,j} \subseteq E_S$ for $(i,j) \in E_r$

Feasible Embedding



Def: Valid mapping $m_r = (m_V, m_E)$ ...

$m_V : V_r \to V_S$ and $m_E : E_r \to \mathcal{P}(E_S)$ satisfies

valid connectivity: $m_V(i) \overset{m_E(i,j)}{\rightsquigarrow} m_V(j)$

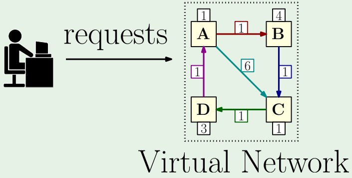
valid node mapping: $m_V(i) \in V_S^i$

valid edge mapping: $m_E(i,j) \subseteq E_S^{i,j}$

Def: Feasible embedding m_r ...

... is **valid** and **respects capacities**.

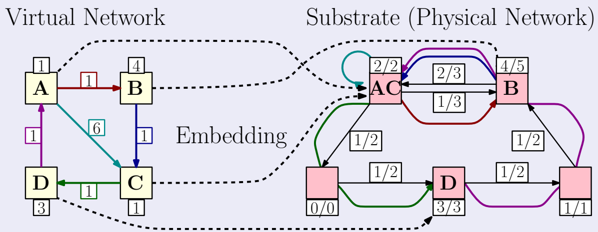
Goal: Virtual Networks (since $\approx$ 2006)



Virtual Network Request $G_r = (V_r, E_r)$

- demands $d_r : G_r \rightarrow \mathbb{R}_{\geq 0}$
- mapping restrictions
  - $V_S^i \subseteq V_S$ for $i \in V_r$
  - $E_S^{i,j} \subseteq E_S$ for $(i,j) \in E_r$

Feasible Embedding



Def: Feasible embedding m_r ...

... is **valid** and **respects capacities**.

Virtual Network Embedding Problem

Setting	Online vs. Offline
Objectives	resource minimization, profit maximization, energy minimization, ...

Related Work & Overview of Contributions

Related Work

Computational Complexity

Andersen [2002]
$\mathcal{NP}$-hardness (argument)

Amaldi et al. [2016]
$\mathcal{NP}$-hardness and inapproximability for offline VNEP (profit)

Heuristics & Exact Algorithms

Generally
$\gg$ 100 works, e.g. ...

Chowdhury et al. [2009]
Heuristics based on **Linear Programming**; hoped for *approximations*...

Approximations

None for general graphs!

Bansal et al. [2011] for trees

Even et al. [2016] for chains

Related Work

Computational Complexity

Andersen [2002]
$\mathcal{NP}$-hardness (argument)

Amaldi et al. [2016]
$\mathcal{NP}$-hardness and inapproximability for offline VNEP (profit)

Heuristics & Exact Algorithms

Generally
$\gg$ 100 works, e.g. ...

Chowdhury et al. [2009]
Heuristics based on **Linear Programming**; hoped for *approximations*...

Approximations

None for general graphs!

Bansal et al. [2011] for trees

Even et al. [2016] for chains

VNEP is of **crucial** importance, yet is **hardly understood**!

Related Work

Computational Complexity

Andersen [2002]
$\mathcal{NP}$-hardness (argument)

Amaldi et al. [2016]
$\mathcal{NP}$-hardness and inapproximability for offline VNEP (profit)

Heuristics & Exact Algorithms

Generally
$\gg$ 100 works, e.g. ...

Chowdhury et al. [2009]
Heuristics based on **Linear Programming**; hoped for *approximations*...

Approximations

None for general graphs!

Bansal et al. [2011] for trees

Even et al. [2016] for chains

Idea of this Talk: Give Overview on Our Results

Complexity results showing $\mathcal{NP}$-completeness and inapproximability.

(FPT-)Linear Programs for computing *convex combinations* of *valid mappings*.

(FPT-)Approximations for *offline* VNEP based on randomized rounding.

Computational evaluation of *derived* heuristics for *offline profit* VNEP.

Related Work

Computational Complexity

Andersen [2002]
$\mathcal{NP}$-hardness (argument)

Amaldi et al. [2016]
$\mathcal{NP}$-hardness and inapproximability for offline VNEP (profit)

Heuristics & Exact Algorithms

Generally
$\gg$ 100 works, e.g. ...

Chowdhury et al. [2009]
Heuristics based on **Linear Programming**; hoped for *approximations*...

Approximations

None for general graphs!

Bansal et al. [2011] for trees

Even et al. [2016] for chains

Idea of this Talk: Give Overview on Our Results

Complexity results showing $\mathcal{NP}$-completeness and inapproximability.

(FPT-)Linear Programs for computing *convex combinations* of *valid mappings*.

(FPT-)Approximations for *offline* VNEP based on randomized rounding.

Computational evaluation of *derived* heuristics for *offline profit* VNEP.

Related Work

Computational Complexity

Andersen [2002]
$\mathcal{NP}$-hardness (argument)

Amaldi et al. [2016]
$\mathcal{NP}$-hardness and inapproximability for offline VNEP (profit)

Heuristics & Exact Algorithms

Generally
$\gg$ 100 works, e.g. ...

Chowdhury et al. [2009]
Heuristics based on **Linear Programming**; hoped for *approximations*...

Approximations

None for general graphs!

Bansal et al. [2011] for trees

Even et al. [2016] for chains

Idea of this Talk: Give Overview on Our Results

Complexity results showing $\mathcal{NP}$-completeness and inapproximability.

(FPT-)Linear Programs for computing *convex combinations* of *valid mappings*.

(FPT-)Approximations for *offline* VNEP based on randomized rounding.

Computational evaluation of *derived* heuristics for *offline profit* VNEP.

Related Work

Computational Complexity

Andersen [2002]
$\mathcal{NP}$-hardness (argument)

Amaldi et al. [2016]
$\mathcal{NP}$-hardness and inapproximability for offline VNEP (profit)

Heuristics & Exact Algorithms

Generally
$\gg$ 100 works, e.g. ...

Chowdhury et al. [2009]
Heuristics based on **Linear Programming**; hoped for *approximations*...

Approximations

None for general graphs!

Bansal et al. [2011] for trees

Even et al. [2016] for chains

Idea of this Talk: Give Overview on Our Results

Complexity results	showing $\mathcal{NP}$-completeness and inapproximability.
(FPT-)Linear Programs	for computing *convex combinations* of *valid mappings*.
(FPT-)Approximations	for *offline* VNEP based on randomized rounding.
Computational evaluation	of *derived* heuristics for *offline profit* VNEP.

Idea of this Talk: Give Overview on Our Results

Complexity results	showing $\mathcal{NP}$-completeness and inapproximability[a].
(FPT-)Linear Programs	for computing *convex combinations* of *valid mappings*[b,c].
(FPT-)Approximations	for *offline* VNEP based on randomized rounding[b,c].
Computational evaluation	of *derived* heuristics for *offline profit* VNEP[b].

---

a Matthias Rost and Stefan Schmid. Charting the Complexity Landscape of Virtual Network Embeddings. In *Proc. IFIP Networking*, 2018c

b Matthias Rost and Stefan Schmid. Virtual Network Embedding Approximations: Leveraging Randomized Rounding. In *Proc. IFIP Networking*, 2018d

c Matthias Rost and Stefan Schmid. (FPT-)Approximation Algorithms for the Virtual Network Embedding Problem. Technical report, March 2018a. URL http://arxiv.org/abs/1803.04452

Complexity of the VNEP[1]

[1] Matthias Rost and Stefan Schmid. Charting the Complexity Landscape of Virtual Network Embeddings. In *Proc. IFIP Networking*, 2018c

Reminder: 3-SAT and $\mathcal{NP}$-Completeness

3-SAT-Formula ϕ

$\phi = \bigwedge_{\mathcal{C}_i \in \mathcal{C}_\phi} \mathcal{C}_i$ with $\mathcal{C}_i \in \mathcal{C}_\phi$ being disjunctions of at most 3 (possible negated) literals.

Example 3-SAT formula ϕ over literals $\mathcal{L}_\phi = \{x_1, x_2, x_3, x_4\}$

$$\phi = \underbrace{(x_1 \vee x_2 \vee x_3)}_{\mathcal{C}_1} \wedge \underbrace{(\bar{x}_1 \vee x_2 \vee x_4)}_{\mathcal{C}_2} \wedge \underbrace{(x_2 \vee \bar{x}_3 \vee x_4)}_{\mathcal{C}_3}$$

Definition of 3-SAT

Decide whether satisfying assignment $a : \mathcal{L}_\phi \to \{\mathrm{F}, \mathrm{T}\}$ exists for formula ϕ. Output: Yes/No.

Theorem: Karp [1972]

3-SAT is $\mathcal{NP}$-complete.

Methodology: Proving $\mathcal{NP}$-completeness

Proving $\mathcal{NP}$-completeness of the VNEP

1. VNEP lies in $\mathcal{NP}$ (answer can be checked in polynomial time).
2. Reduction from 3-SAT to VNEP.

Outline of Reduction Framework

$$\text{3-SAT instance } \phi \longmapsto \text{VNEP instance } (G_{r(\phi)}, G_{S(\phi)}, \text{restrictions})$$

$$\phi \text{ satisfiable?} \rightleftarrows \text{feasible embedding of } G_{r(\phi)} \text{ on } G_{S(\phi)} \text{ under restrictions?}$$

Our Reduction Framework

Proving $\mathcal{NP}$-completeness of the VNEP

1. VNEP lies in $\mathcal{NP}$ (answer can be checked in polynomial time).
2. Reduction from 3-SAT to VNEP.

Outline of Reduction Framework

3-SAT instance $\phi \longmapsto$ VNEP instance $(G_{r(\phi)}, G_{S(\phi)}, \text{restrictions})$

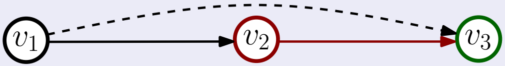
ϕ satisfiable? $\rightleftarrows$ feasible embedding of $G_{r(\phi)}$ on $G_{S(\phi)}$ under restrictions?

Our Reduction Framework

Input: 3-SAT formula $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \bar{x}_3 \vee x_4)$

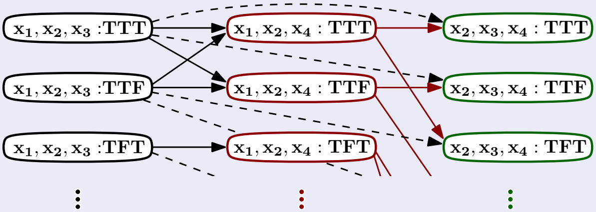
Request $G_{r(\phi)}$

- $V_{r(\phi)} = \{v_i \mid \mathcal{C}_i \in \mathcal{C}_\phi\}$
- $E_{r(\phi)} = \{\ (v_i, v_j) \mid \mathcal{C}_i$ introduces literal used by $\mathcal{C}_j\ \}$



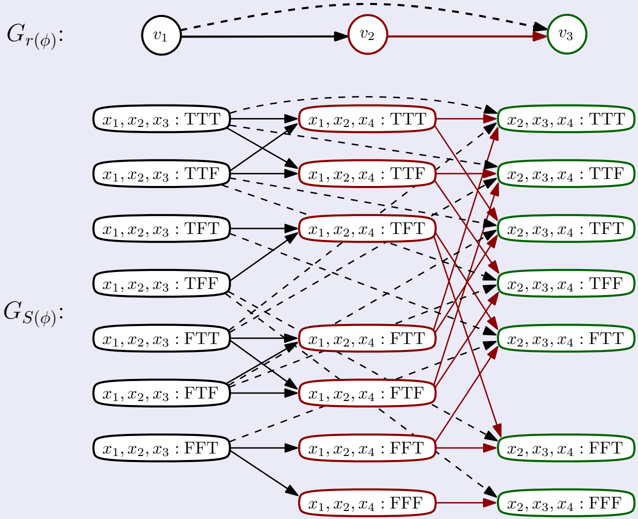
Substrate $G_{S(\phi)}$

- one node per clause and **per satisfying assignment**
- edges as for the requests, **if assignments do not contradict**



Complete Picture

ϕ: $(x_1 \vee x_2 \vee x_3) \quad \wedge \quad (\bar{x}_1 \vee x_2 \vee x_4) \quad \wedge \quad (x_2 \vee \bar{x}_3 \vee x_4)$



Our Reduction Framework

Outline of Reduction Framework

3-SAT instance $\phi \longmapsto$ VNEP instance $(G_{r(\phi)}, G_{S(\phi)}, \text{restrictions})$

ϕ satisfiable? $\rightleftarrows$ feasible embedding of $G_{r(\phi)}$ on $G_{S(\phi)}$ under restrictions?

Our Reduction Framework

Outline of Reduction Framework

3-SAT instance $\phi \longmapsto$ VNEP instance $(G_{r(\phi)}, G_{S(\phi)}, \text{restrictions})$

ϕ satisfiable? $\rightleftarrows$ feasible embedding of $G_{r(\phi)}$ on $G_{S(\phi)}$ under restrictions?

Base Lemma

Formula ϕ is satisfiable **if and only if** there exists a mapping of $G_{r(\phi)}$ on $G_{S(\phi)}$, s.t.

(1) each virtual node v_i is mapped to a 'satisfying assignment node' of the i-th clause, and

(2) all virtual edges are mapped on exactly one substrate edge.

Our Reduction Framework

Base Lemma

Formula ϕ is satisfiable **if and only if** there exists a mapping of $G_{r(\phi)}$ on $G_{S(\phi)}$, s.t.

(1) each virtual node v_i is mapped to a 'satisfying assignment node' of the i-th clause, and

(2) all virtual edges are mapped on exactly one substrate edge.

Example

$\phi = (x_1 \lor x_2 \lor x_3) \land (\bar{x}_1 \lor x_2 \lor x_4) \land (x_2 \lor \bar{x}_3 \lor x_4)$

Assignment for ϕ

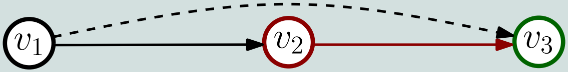
$x_1 \mapsto \mathrm{T}$

$x_2 \mapsto \mathrm{T}$

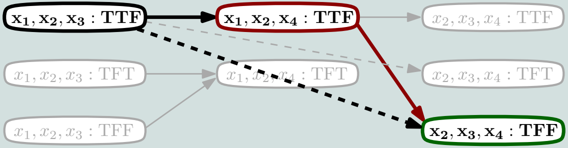
$x_3 \mapsto \mathrm{F}$

$x_4 \mapsto \mathrm{F}$

Request



Embedding satisfying conditions (1) and (2)



Our Reduction Framework

Base Lemma

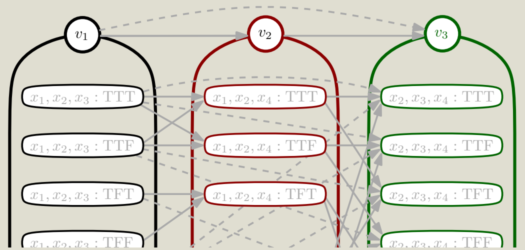
Formula ϕ is satisfiable **if and only if** there exists a mapping of $G_{r(\phi)}$ on $G_{S(\phi)}$, s.t.

(1) each virtual node v_i is mapped to a 'satisfying assignment node' of the i-th clause, and

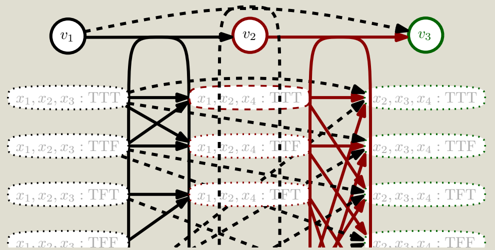
(2) all virtual edges are mapped on exactly one substrate edge.

Decision VNEP is $\mathcal{NP}$-complete under *mapping restrictions*

Node placement restrictions enforce (1)



Routing restrictions enforce (2)



Our Reduction Framework

Base Lemma

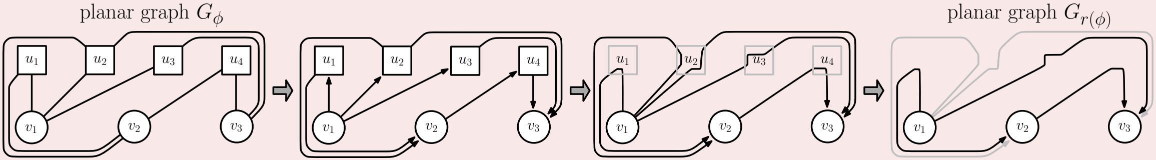
Formula ϕ is satisfiable **if and only if** there exists a mapping of $G_{r(\phi)}$ on $G_{S(\phi)}$, s.t.

(1) each virtual node v_i is mapped to a 'satisfying assignment node' of the i-th clause, and

(2) all virtual edges are mapped on exactly one substrate edge.

Decision VNEP is $\mathcal{NP}$-complete for degree-bounded, planar request graphs

- Reduction from **planar** 3-SAT variant using literals max. **4 times** (see Kratochvíl [1994]):
  - each planar formula ϕ leads to a planar request graph $G_{r(\phi)}$
  - each node of $G_{r(\phi)}$ has degree at most 12



(FPT-)Linear Programs for Computing Convex Combinations of Valid Mappings[2,3]

[2] Matthias Rost and Stefan Schmid. Virtual Network Embedding Approximations: Leveraging Randomized Rounding. In *Proc. IFIP Networking*, 2018d

[3] Matthias Rost and Stefan Schmid. (FPT-)Approximation Algorithms for the Virtual Network Embedding Problem. Technical report, March 2018a. URL `http://arxiv.org/abs/1803.04452`

Linear Programming: Classic MCF Formulation and its Limits

Classic LP Formulation

Formulation 1: Classic MCF Formulation for the VNEP

$$\sum_{u \in V_S^i} y^u_{r,i} = x_r \quad \forall r \in \mathcal{R}, i \in V_r \quad (1)$$

$$\sum_{u \in V_S \setminus V_S^i} y^u_{r,i} = 0 \quad \forall r \in \mathcal{R}, i \in V_r \quad (2)$$

$$\begin{bmatrix} \sum_{(u,v) \in \delta^+(u)} z^{u,v}_{r,i,j} \\ - \sum_{(v,u) \in \delta^-(u)} z^{v,u}_{r,i,j} \end{bmatrix} = \begin{bmatrix} y^u_{r,i} \\ -y^u_{r,j} \end{bmatrix} \forall \begin{bmatrix} r \in \mathcal{R}, (i,j) \in E_r, \\ u \in V_S \end{bmatrix} \quad (3)$$

$$z^{u,v}_{r,i,j} = 0 \quad \forall \begin{bmatrix} r \in \mathcal{R}, (i,j) \in E_r, \\ (u,v) \in E_S \setminus E_S^{i,j} \end{bmatrix} \quad (4)$$

$$\sum_{i \in V_r, \tau_r(i) = \tau} d_r(i) \cdot y^u_{r,i} = a^{\tau,u}_r \quad \forall r \in \mathcal{R}, (\tau, u) \in R^V_S \quad (5)$$

$$\sum_{(i,j) \in E_r} d_r(i,j) \cdot z^{u,v}_{r,i,j} = a^{u,v}_r \quad \forall r \in \mathcal{R}, (u,v) \in E_S \quad (6)$$

$$\sum_{r \in \mathcal{R}} a^{x,y}_r \leq c_S(x,y) \, \forall (x,y) \in R_S \quad (7)$$

Main Building Block: Multi-Commodity Flows

$\mathbf{y^u_{r,i}} \in [0,1]$: maps node $i \in V_r$ on V_S

$\mathbf{z^{u,v}_{r,i,j}} \in [0,1]$: maps $(i,j) \in E_r$ on $(u,v) \in E_S$

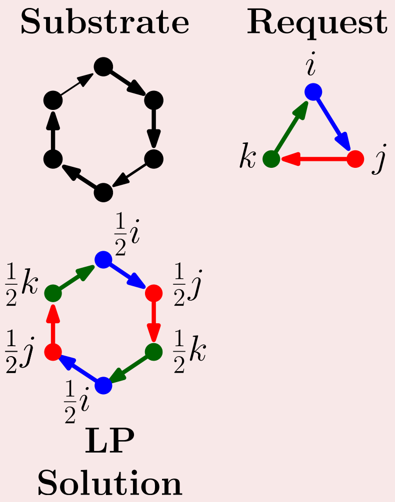
$$\sum_{(\mathbf{u},\mathbf{v}) \in \delta^+(\mathbf{u})} \mathbf{z^{u,v}_{r,i,j}} - \sum_{(\mathbf{v},\mathbf{u}) \in \delta^-(\mathbf{u})} \mathbf{z^{v,u}_{r,i,j}} = \mathbf{y^u_{r,i}} - \mathbf{y^u_{r,j}} \quad (3)$$

Local Connectivity Property

Given a (fractional) mapping of $i \in V_r$ to $u \in V_S$, a 'valid' mapping can be recovered for edges incident to i and their respective endpoints.

Linear Programming: Classic MCF Formulation and its Limits

Example



Main Building Block: Multi-Commodity Flows

$\mathbf{y^u_{r,i}} \in [0,1]$: maps node $i \in V_r$ on V_S

$\mathbf{z^{u,v}_{r,i,j}} \in [0,1]$: maps $(i,j) \in E_r$ on $(u,v) \in E_S$

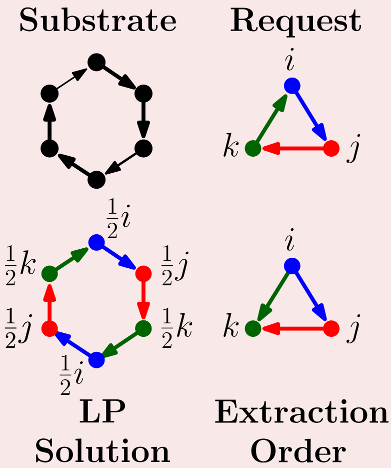
$$\sum_{(\mathbf{u},\mathbf{v}) \in \delta^+(\mathbf{u})} \mathbf{z^{u,v}_{r,i,j}} - \sum_{(\mathbf{v},\mathbf{u}) \in \delta^-(\mathbf{u})} \mathbf{z^{v,u}_{r,i,j}} = \mathbf{y^u_{r,i}} - \mathbf{y^u_{r,j}} \quad (3)$$

Local Connectivity Property

Given a (fractional) mapping of $i \in V_r$ to $u \in V_S$, a 'valid' mapping can be recovered for edges incident to i and their respective endpoints.

Linear Programming: Classic MCF Formulation and its Limits

Example



Main Building Block: Multi-Commodity Flows

$\mathbf{y}^{\mathbf{u}}_{\mathbf{r,i}} \in [0,1]$: maps node $i \in V_r$ on V_S

$\mathbf{z}^{\mathbf{u,v}}_{\mathbf{r,i,j}} \in [0,1]$: maps $(i,j) \in E_r$ on $(u,v) \in E_S$

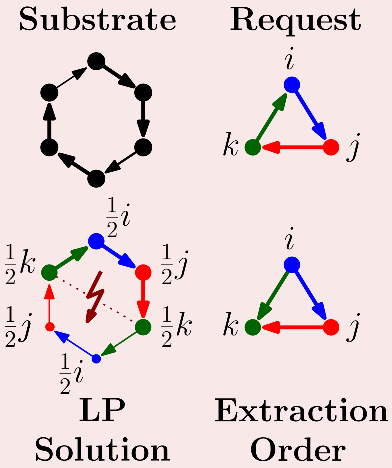
$$\sum_{(\mathbf{u},\mathbf{v}) \in \delta^+(\mathbf{u})} \mathbf{z}^{\mathbf{u,v}}_{\mathbf{r,i,j}} - \sum_{(\mathbf{v},\mathbf{u}) \in \delta^-(\mathbf{u})} \mathbf{z}^{\mathbf{v,u}}_{\mathbf{r,i,j}} = \mathbf{y}^{\mathbf{u}}_{\mathbf{r,i}} - \mathbf{y}^{\mathbf{u}}_{\mathbf{r,j}} \quad (3)$$

Local Connectivity Property

Given a (fractional) mapping of $i \in V_r$ to $u \in V_S$, a 'valid' mapping can be recovered for edges incident to i and their respective endpoints.

Linear Programming: Classic MCF Formulation and its Limits

Example



Local Connectivity Property

Given a (fractional) mapping of $i \in V_r$ to $u \in V_S$, a 'valid' mapping can be recovered for edges incident to i and their respective endpoints.

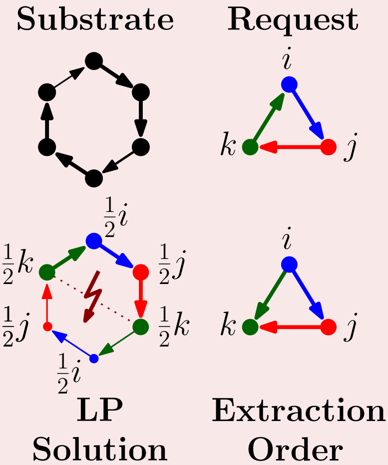
Main Issue

Targets of confluences pose problems!

In the example: target k of confluence $\langle (i,k) \rangle, \langle (i,j),(j,k) \rangle$.

Linear Programming: Classic MCF Formulation and its Limits

Example



Main Issue

Targets of confluences pose problems!

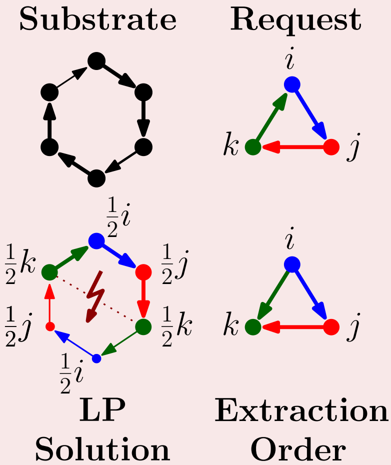
In the example: target k of confluence $\langle (i,k) \rangle, \langle (i,j),(j,k) \rangle$.

Theorem

Decomposing solutions to the MCF LP **is not possible in general**.

Linear Programming: Classic MCF Formulation and its Limits

Example



Main Issue

Targets of confluences pose problems!

In the example: target k of confluence $\langle (i,k) \rangle, \langle (i,j),(j,k) \rangle$.

Theorem

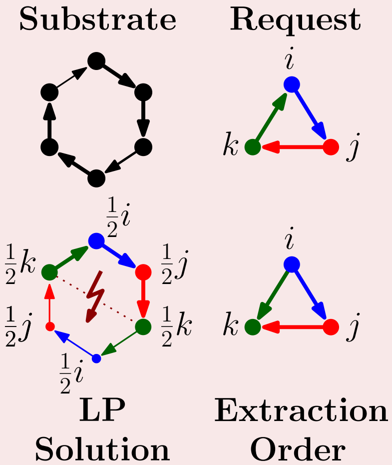
Decomposing solutions to the MCF LP **is not possible in general**.

Theorem

MCF LP Formulation has **infinite integrality gap**.

Linear Programming: Classic MCF Formulation and its Limits

Example



Main Issue

Targets of confluences pose problems!

In the example: target k of confluence $\langle (i,k) \rangle, \langle (i,j),(j,k) \rangle$.

Key Insight

If we fix confluence target nodes valid mappings can always be extracted, when following the extraction order.

In the example:
Consider one **sub-LP** formulation per potential mapping location of k.

Outline of Novel Decomposable LP Formulations

Extraction Order $G_r^{\mathcal{X}}$

Rooted acyclic reorientation of the original request graph G_r. $G_r^{\mathcal{X}}$ is **not unique**!

Confluence $C_{i,j}^{\mathcal{X}}$

A confluence $C_{i,j}^{\mathcal{X}}$ from i to j is a pair of (node-)disjoint paths connecting i to j in $G_r^{\mathcal{X}}$.

Outline of Novel Decomposable LP Formulations

Extraction Order $G_r^{\mathcal{X}}$

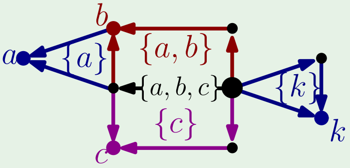
Rooted acyclic reorientation of the original request graph G_r. $G_r^{\mathcal{X}}$ is **not unique**!

Confluence $C_{i,j}^{\mathcal{X}}$

A confluence $C_{i,j}^{\mathcal{X}}$ from i to j is a pair of (node-)disjoint paths connecting i to j in $G_r^{\mathcal{X}}$.

Pre-Processing Extraction Orders

- If edge e lies on confluence $C_{i,j}^{\mathcal{X}}$, then it is labeled with the confluence's target j.
- Labeling can be computed in polynomial-time (by applying Menger's theorem).
- Each label has unique root node at which the mapping of the label must be fixed.
- Outgoing edges are partitioned into **edge bags** not sharing labels.



Outline of Novel Decomposable LP Formulations

Extraction Order $G_r^{\mathcal{X}}$

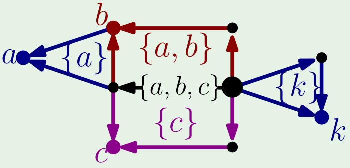
Rooted acyclic reorientation of the original request graph G_r. $G_r^{\mathcal{X}}$ is **not unique**!

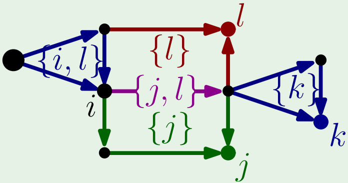
Confluence $C_{i,j}^{\mathcal{X}}$

A confluence $C_{i,j}^{\mathcal{X}}$ from i to j is a pair of (node-)disjoint paths connecting i to j in $G_r^{\mathcal{X}}$.

Pre-Processing Extraction Orders

- If edge e lies on confluence $C_{i,j}^{\mathcal{X}}$, then it is labeled with the confluence's target j.
- Labeling can be computed in polynomial-time (by applying Menger's theorem).
- Each label has unique root node at which the mapping of the label must be fixed.
- Outgoing edges are partitioned into **edge bags** not sharing labels.





Outline of Novel Decomposable LP Formulations

Extraction Order $G_r^{\mathcal{X}}$

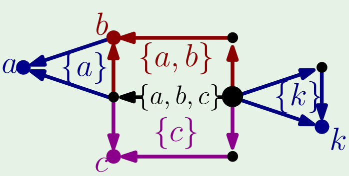
Rooted acyclic reorientation of the original request graph G_r. $G_r^{\mathcal{X}}$ is **not unique**!

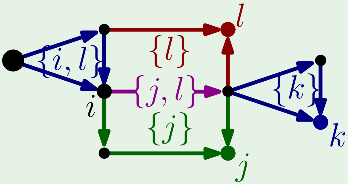
Confluence $C_{i,j}^{\mathcal{X}}$

A confluence $C_{i,j}^{\mathcal{X}}$ from i to j is a pair of (node-)disjoint paths connecting i to j in $G_r^{\mathcal{X}}$.

Pre-Processing Extraction Orders

- If edge e lies on confluence $C_{i,j}^{\mathcal{X}}$, then it is labeled with the confluence's target j.
- Labeling can be computed in polynomial-time (by applying Menger's theorem).
- Each label has unique root node at which the mapping of the label must be fixed.
- Outgoing edges are partitioned into **edge bags** not sharing labels.





Outline of Novel Decomposable LP Formulations

Extraction Order $G_r^{\mathcal{X}}$

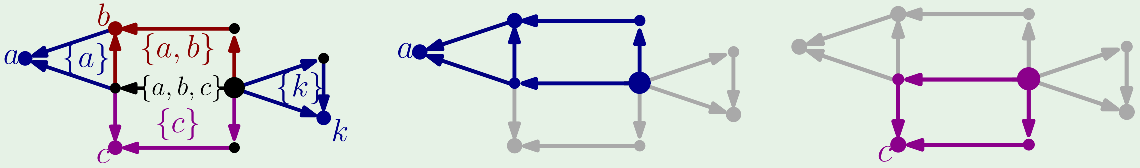
Rooted acyclic reorientation of the original request graph G_r. $G_r^{\mathcal{X}}$ is **not unique**!

Confluence $C_{i,j}^{\mathcal{X}}$

A confluence $C_{i,j}^{\mathcal{X}}$ from i to j is a pair of (node-)disjoint paths connecting i to j in $G_r^{\mathcal{X}}$.

Pre-Processing Extraction Orders

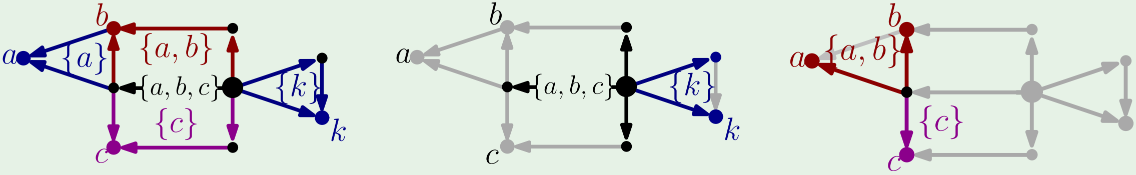
- If edge e lies on confluence $C_{i,j}^{\mathcal{X}}$, then it is labeled with the confluence's target j.
- Labeling can be computed in polynomial-time (by applying Menger's theorem).
- Each label has unique root node at which the mapping of the label must be fixed.
- Outgoing edges are partitioned into **edge bags** not sharing labels.



Outline of Novel Decomposable LP Formulations

Pre-Processing Extraction Orders

- If edge e lies on confluence $C^{\mathcal{X}}_{i,j}$, then it is labeled with the confluence's target j.
- Labeling can be computed in polynomial-time (by applying Menger's theorem).
- Each label has unique root node at which the mapping of the label must be fixed.
- Outgoing edges are partitioned into **edge bags** not sharing labels.



Outline of Novel Decomposable LP Formulation

Extraction Order $G_r^{\mathcal{X}}$

Rooted acyclic reorientation of the original request graph G_r. $G_r^{\mathcal{X}}$ is **not unique**!

Confluence $C_{i,j}^{\mathcal{X}}$

A confluence $C_{i,j}^{\mathcal{X}}$ from i to j is a pair of (node-)disjoint paths connecting i to j in $G_r^{\mathcal{X}}$.

Pre-Processing Extraction Orders

- If edge $e \in E_r^{\mathcal{X}}$ lies on confluence $C_{i,j}^{\mathcal{X}}$, then it is labeled with the confluence's target j.
- Outgoing edges are partitioned into **edge bags** not sharing labels.
- Each label has unique root node at which the mapping of the label must be fixed.

Generation of Linear Program

- If $e \in E_r$ is labeled with $\mathcal{L}_{r,e}^{\mathcal{X}}$, then $|V_S|^{|\mathcal{L}_{r,e}^{\mathcal{X}}|}$ many commodities are considered for e.
- For each edge bag variables are introduced to enumerate all potential label mappings.
- Root nodes of labels 'decide' on the confluence's mapping.

Outline of Novel Decomposable LP Formulation

Extraction Order $G_r^{\mathcal{X}}$

Rooted acyclic reorientation of the original request graph G_r. $G_r^{\mathcal{X}}$ is **not unique**!

Confluence $C_{i,j}^{\mathcal{X}}$

A confluence $C_{i,j}^{\mathcal{X}}$ from i to j is a pair of (node-)disjoint paths connecting i to j in $G_r^{\mathcal{X}}$.

Pre-Processing Extraction Orders

- If edge $e \in E_r^{\mathcal{X}}$ lies on confluence $C_{i,j}^{\mathcal{X}}$, then it is labeled with the confluence's target j.
- Outgoing edges are partitioned into **edge bags** not sharing labels.
- Each label has unique root node at which the mapping of the label must be fixed.

Generation of Linear Program

- If $e \in E_r$ is labeled with $\mathcal{L}_{r,e}^{\mathcal{X}}$, then $|V_S|^{|\mathcal{L}_{r,e}^{\mathcal{X}}|}$ many commodities are considered for e.
- For each edge bag variables are introduced to enumerate all potential label mappings.
- Root nodes of labels 'decide' on the confluence's mapping.

Outline of Novel Decomposable LP Formulation

Extraction Order $G_r^{\mathcal{X}}$

Rooted acyclic reorientation of the original request graph G_r. $G_r^{\mathcal{X}}$ is **not unique**!

Confluence $C_{i,j}^{\mathcal{X}}$

A confluence $C_{i,j}^{\mathcal{X}}$ from i to j is a pair of (node-)disjoint paths connecting i to j in $G_r^{\mathcal{X}}$.

Pre-Processing Extraction Orders

- If edge $e \in E_r^{\mathcal{X}}$ lies on confluence $C_{i,j}^{\mathcal{X}}$, then it is labeled with the confluence's target j.
- Outgoing edges are partitioned into **edge bags** not sharing labels.
- Each label has unique root node at which the mapping of the label must be fixed.

Generation of Linear Program

- If $e \in E_r$ is labeled with $\mathcal{L}_{r,e}^{\mathcal{X}}$, then $|V_S|^{|\mathcal{L}_{r,e}^{\mathcal{X}}|}$ many commodities are considered for e.
- For each edge bag variables are introduced to enumerate all potential label mappings.
- Root nodes of labels 'decide' on the confluence's mapping.

Outline of Novel Decomposable LP Formulation

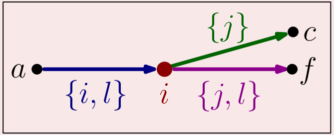
Pre-Processing Extraction Orders

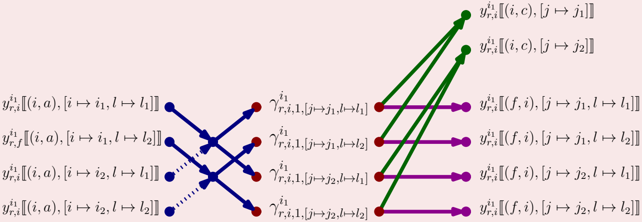
- If edge $e \in E_r^{\mathcal{X}}$ lies on confluence $C_{i,j}^{\mathcal{X}}$, then it is labeled with the confluence's target j.
- Outgoing edges are partitioned into **edge bags** not sharing labels.
- Each label has unique root node at which the mapping of the label must be fixed.

Generation of Linear Program

- If $e \in E_r$ is labeled with $\mathcal{L}_{r,e}^{\mathcal{X}}$, then $|V_S|^{|\mathcal{L}_{r,e}^{\mathcal{X}}|}$ many commodities are considered for e.
- For each edge bag variables are introduced to enumerate all potential label mappings.
- Root nodes of labels 'decide' on the confluence's mapping.

Stitching Flow Variables via Node Mapping Variables





Outline of Novel Decomposable LP Formulation

Pre-Processing Extraction Orders

- If edge $e \in E_r^{\mathcal{X}}$ lies on confluence $C_{i,j}^{\mathcal{X}}$, then it is labeled with the confluence's target j.
- Outgoing edges are partitioned into **edge bags** not sharing labels.
- Each label has unique root node at which the mapping of the label must be fixed.

Generation of Linear Program

- If $e \in E_r$ is labeled with $\mathcal{L}_{r,e}^{\mathcal{X}}$, then $|V_S|^{|\mathcal{L}_{r,e}^{\mathcal{X}}|}$ many commodities are considered for e.
- For each edge bag variables are introduced to enumerate all potential label mappings.
- Root nodes of labels 'decide' on the confluence's mapping.

Def. Extraction Width $\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})$

...is the size of the largest edge bag plus one of the extraction order $\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})$.

Proof of Decomposability

...via decomposition algorithm. Overall runtime $\mathcal{O}(\mathrm{poly}(|G_S|^{\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})} \cdot |G_r|))$.

Novel Decomposable LP Formulation: Takeaways

Overview of Construction

Request Graph G_r

$\Downarrow$

Extraction Order $G_r^{\mathcal{X}}$

$\Downarrow$

Labeling / Extraction Width $\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})$

$\Downarrow$

LP of size $\mathcal{O}(\mathrm{poly}(|G_S|^{\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})} \cdot |G_r|))$

$\Downarrow$

Decomposition Algorithm with runtime $\mathcal{O}(\mathrm{poly}(|G_S|^{\mathrm{ew}_{\mathcal{X}}(G_r^{\mathcal{X}})} \cdot |G_r|))$

$\Downarrow$

Convex Combinations of valid mappings: $\mathcal{D}_r = \{(f_r^k, m_r^k) | f_r^k > 0, m_r^k \in \mathcal{M}_r\}$

Extraction Width – Overview of Results

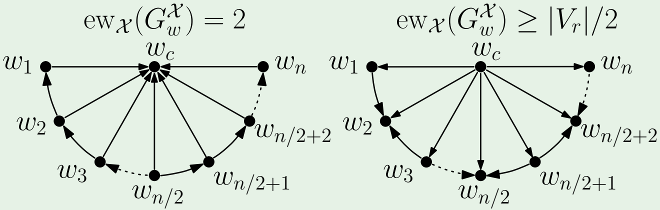
- Which graphs have bounded extraction width?
- How to find extraction orders of small width?

Extraction Width – Overview of Results

- Which graphs have bounded extraction width?
- How to find extraction orders of small width?

Extraction Width: Overview of Results

- Extraction width may vary by factor $\Omega(|V_r|)$
- Minimizing extraction width is $\mathcal{NP}$-hard (via reduction from Vertex-Cover)
- Cactus graphs (cycles intersect in at most a single node) have bounded extraction width

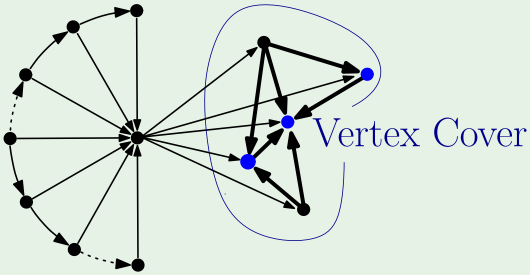


Extraction Width – Overview of Results

- Which graphs have bounded extraction width?
- How to find extraction orders of small width?

Extraction Width: Overview of Results

- Extraction width may vary by factor $\Omega(|V_r|)$
- Minimizing extraction width is $\mathcal{NP}$-hard (via reduction from Vertex-Cover)
- Cactus graphs (cycles intersect in at most a single node) have bounded extraction width



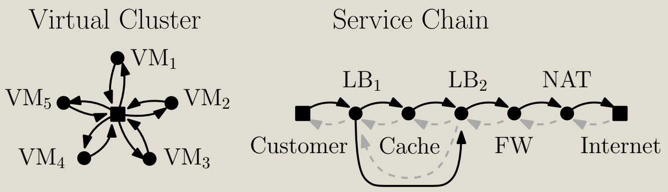
Extraction Width – Overview of Results

- Which graphs have bounded extraction width?
- How to find extraction orders of small width?

Extraction Width: Overview of Results

- Extraction width may vary by factor $\Omega(|V_r|)$
- Minimizing extraction width is $\mathcal{NP}$-hard (via reduction from Vertex-Cover)
- Cactus graphs (cycles intersect in at most a single node) have bounded extraction width

Real World Cactus Graph Examples



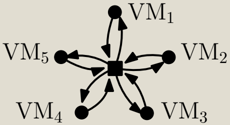
Extraction Width – Overview of Results

Extraction Width: Overview of Results

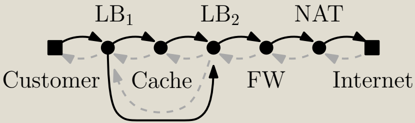
- Extraction width may vary by factor $\Omega(|V_r|)$
- Minimizing extraction width is $\mathcal{NP}$-hard (via reduction from Vertex-Cover)
- Cactus graphs (cycles intersect in at most a single node) have bounded extraction width

Real World Cactus Graph Examples

Virtual Cluster



Service Chain



Can we do substantially better? No!

Computing valid mappings for planar graphs is $\mathcal{NP}$-complete $\Rightarrow$ FPT algorithms are necessary.

(FPT-)Approximations for offline VNEP based on Randomized Rounding[2,3]

[2] Matthias Rost and Stefan Schmid. Virtual Network Embedding Approximations: Leveraging Randomized Rounding. In *Proc. IFIP Networking*, 2018d

[3] Matthias Rost and Stefan Schmid. (FPT-)Approximation Algorithms for the Virtual Network Embedding Problem. Technical report, March 2018a. URL `http://arxiv.org/abs/1803.04452`

Approximating the Offline VNEP

Profit Variant

- A **set** of request $\mathcal{R} = \{r_1, r_2, \dots\}$ is given.
- Profit for request $p_r > 0$.
- **Task:** Embed subset of requests *feasibly* maximizing the attained profit.

Approximating the Offline VNEP

Profit Variant

- A **set** of request $\mathcal{R} = \{r_1, r_2, \ldots\}$ is given.
- Profit for request $p_r > 0$.
- **Task:** Embed subset of requests *feasibly* maximizing the attained profit.

Cost Variant

- A **set** of request $\mathcal{R} = \{r_1, r_2, \ldots\}$ is given.
- Substrate resource costs $k_S : G_S \to \mathbb{R}_{\geq 0}$.
- **Task:** Find *feasible* embeddings for all requests minimizing cost.

Approximating the Offline VNEP

Focus: Profit Variant

- A **set** of request $\mathcal{R} = \{r_1, r_2, \dots\}$ is given.
- Profit for request $p_r > 0$.
- **Task:** Embed subset of requests *feasibly* maximizing the attained profit.

Cost Variant

- A **set** of request $\mathcal{R} = \{r_1, r_2, \dots\}$ is given.
- Substrate resource costs $k_S : G_S \to \mathbb{R}_{\geq 0}$.
- **Task:** Find *feasible* embeddings for all requests minimizing cost.

Approximating the Offline VNEP

Focus: Profit Variant

- A **set** of request $\mathcal{R} = \{r_1, r_2, \ldots\}$ is given.
- Profit for request $p_r > 0$.
- **Task:** Embed subset of requests *feasibly* maximizing the attained profit.

Cost Variant

- A **set** of request $\mathcal{R} = \{r_1, r_2, \ldots\}$ is given.
- Substrate resource costs $k_S : G_S \to \mathbb{R}_{\geq 0}$.
- **Task:** Find *feasible* embeddings for all requests minimizing cost.

Combine Single Decomposable LP Formulations while ...

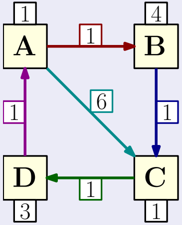
...enforcing capacity constraints and maximizing the profit. $\Rightarrow$ LP for offline VNEP (profit).

Approximating the Offline VNEP

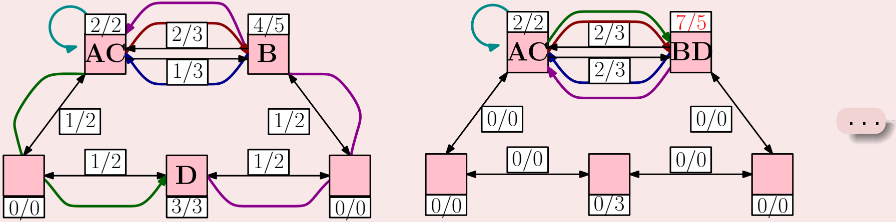
Combine Single Decomposable LP Formulations while . . .

. . . enforcing capacity constraints and maximizing the profit. ⇒ LP for offline VNEP (profit).

Request



Valid Mappings $\mathcal{M}_r = \{m_r^1, m_r^2, m_r^3, \ldots\}$



Valid mappings do not necessarily respect capacity constraints!

Approximating the Offline VNEP

Combine Single Decomposable LP Formulations while ...

... enforcing capacity constraints and maximizing the profit. $\Rightarrow$ LP for offline VNEP (profit).

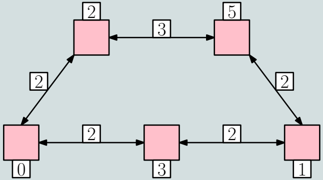
Decomposable LP Formulation allows us to solve *Fractional VNEP*

- Is k-th mapping of request r chosen? $$f_r^k \in \{0,1\} \qquad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \tag{8}$$
- Select at most one mapping: $$\sum_{m_r^k \in \mathcal{M}_r} f_r^k \leq 1 \qquad \forall r \in \mathcal{R} \tag{9}$$
- Enforce capacity for each resource x: $$\sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} A(m_r^k, x) \cdot f_r^k \leq c_S(x) \qquad \forall x \in R_S \tag{10}$$
- Maximize the profit: $$\max \sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} p_r f_r^k \tag{11}$$

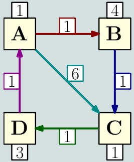
Randomized Rounding Revisited

Example

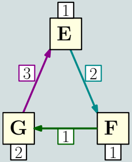
Substrate Network



Request r_1: profit 100$



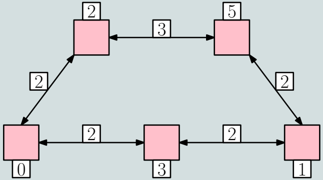
Request r_2: profit 50$



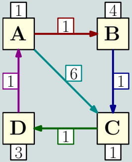
Randomized Rounding Revisited

Example

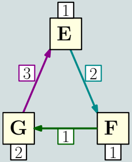
Substrate Network



Request r_1: profit 100$



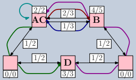
Request r_2: profit 50$



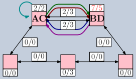
Example Solution to Linear Program: Profit 133$

Variables of r_1 (profit: 100$)

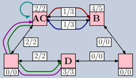
$f_1^1 = 0.5$



$f_1^2 = 0.3$



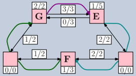
$f_1^3 = 0.2$



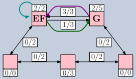
...

Variables of r_2 (profit: 50$)

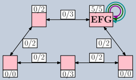
$f_2^1 = 0.5$



$f_2^2 = 0.16$



$f_2^3 = 0$



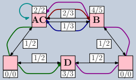
...

Randomized Rounding Revisited

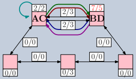
Example Solution to Linear Program: Profit 133$

Variables of r_1 (profit: 100$)

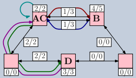
$f_1^1 = 0.5$



$f_1^2 = 0.3$



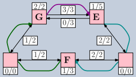
$f_1^3 = 0.2$



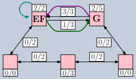
...

Variables of r_2 (profit: 50$)

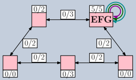
$f_2^1 = 0.5$



$f_2^2 = 0.16$



$f_2^3 = 0$



...

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$

foreach $r \in \mathcal{R}$ **do**

  choose m_r^k **with probability** f_r^k

end

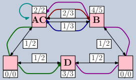
return *solution*

Randomized Rounding Revisited

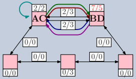
Example Solution to Linear Program: Profit 133$

Variables of r_1 (profit: 100$)

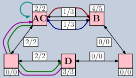
$f_1^1 = 0.5$



$f_1^2 = 0.3$



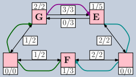
$f_1^3 = 0.2$



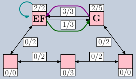
...

Variables of r_2 (profit: 50$)

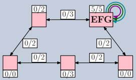
$f_2^1 = 0.5$



$f_2^2 = 0.16$



$f_2^3 = 0$



...

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$
foreach $r \in \mathcal{R}$ **do**
  choose m_r^k **with probability** f_r^k
end
return *solution*

Rounding Outcomes

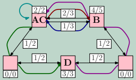
Iter.	Req. 1	Req. 2	Profit	max Load

Randomized Rounding Revisited

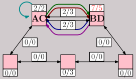
Example Solution to Linear Program: Profit 133$

Variables of r_2 (profit: 50$)

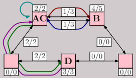
$f_1^1 = 0.5$



$f_1^2 = 0.3$



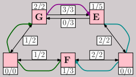
$f_1^3 = 0.2$



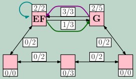
...

Variables of r_2 (profit: 50$)

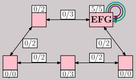
$f_2^1 = 0.5$



$f_2^2 = 0.16$



$f_2^3 = 0$



...

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$

foreach $r \in \mathcal{R}$ **do**

  choose m_r^k **with probability** f_r^k

end

return *solution*

Rounding Outcomes

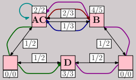
Iter.	Req. 1	Req. 2	Profit	max Load
1	m_1^1	m_2^2	150$	200%

Randomized Rounding Revisited

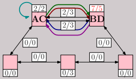
Example Solution to Linear Program: Profit 133$

Variables of request 1

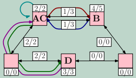
$f_1^1 = 0.5$



$f_1^2 = 0.3$



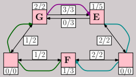
$f_1^3 = 0.2$



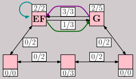
...

Variables of r_2 (profit: 50$)

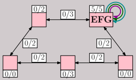
$f_2^1 = 0.5$



$f_2^2 = 0.16$



$f_2^3 = 0$



...

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$

foreach $r \in \mathcal{R}$ **do**

  choose m_r^k **with probability** f_r^k

end

return *solution*

Rounding Outcomes

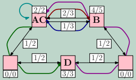
Iter.	Req. 1	Req. 2	Profit	max Load
1	m_1^1	m_2^2	150$	200%
2	m_1^3	$\emptyset$	100$	100%

Randomized Rounding Revisited

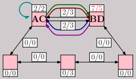
Example Solution to Linear Program: Profit 133$

Variables of request 1

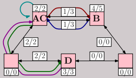
$f_1^1 = 0.5$



$f_1^2 = 0.3$



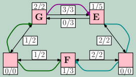
$f_1^3 = 0.2$



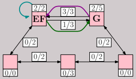
...

Variables of r_2 (profit: 50$)

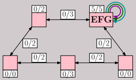
$f_2^1 = 0.5$



$f_2^2 = 0.16$



$f_2^3 = 0$



...

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$

foreach $r \in \mathcal{R}$ **do**

  choose m_r^k **with probability** f_r^k

end

return *solution*

Rounding Outcomes

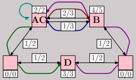
Iter.	Req. 1	Req. 2	Profit	max Load
1	m_1^1	m_2^2	150$	200%
2	m_1^3	$\emptyset$	100$	100%
3	m_1^1	m_2^1	150$	200%

Randomized Rounding Revisited

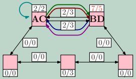
Example Solution to Linear Program: Profit 133$

Variables of request 1

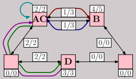
$f_1^1 = 0.5$



$f_1^2 = 0.3$



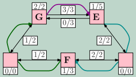
$f_1^3 = 0.2$



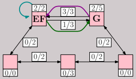
...

Variables of r_2 (profit: 50$)

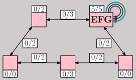
$f_2^1 = 0.5$



$f_2^2 = 0.16$



$f_2^3 = 0$



...

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$

foreach $r \in \mathcal{R}$ **do**

  choose m_r^k **with probability** f_r^k

end

return *solution*

Rounding Outcomes

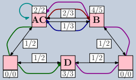
Iter.	Req. 1	Req. 2	Profit	max Load
1	m_1^1	m_2^2	150$	200%
2	m_1^3	$\emptyset$	100$	100%
3	m_1^1	m_2^1	150$	200%
4	m_1^2	m_2^1	150$	200%

Randomized Rounding Revisited

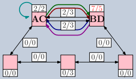
Example Solution to Linear Program: Profit 133$

Variables of request 1

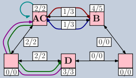
$f_1^1 = 0.5$



$f_1^2 = 0.3$



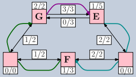
$f_1^3 = 0.2$



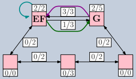
...

Variables of r_2 (profit: 50$)

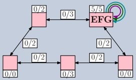
$f_2^1 = 0.5$



$f_2^2 = 0.16$



$f_2^3 = 0$



...

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$

foreach $r \in \mathcal{R}$ **do**

  choose m_r^k **with probability** f_r^k

end

return *solution*

Rounding Outcomes

Iter.	Req. 1	Req. 2	Profit	max Load
1	m_1^1	m_2^2	150$	200%
2	m_1^3	$\emptyset$	100$	100%
3	m_1^1	m_2^1	150$	200%
4	m_1^2	m_2^1	150$	200%
⋮	⋮	⋮	⋮	⋮

First (FPT-)Approximation Algorithm for VNEP

Randomized Rounding Approximation

Algorithm: VNEP Approximation (Profit)

```
// perform preprocessing
```

compute *optimal* LP solution
compute $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$ from LP solution
do
  solution $\leftarrow$ RoundingProcedure($\{\mathcal{D}_r\}_{r \in \mathcal{R}}$)
while (solution *not* (α, β, γ)-approximate and rounding tries not exceeded)

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$
foreach $r \in \mathcal{R}$ **do**
  choose m_r^k **with probability** f_r^k
end
return *solution*

First (FPT-)Approximation Algorithm for VNEP

Randomized Rounding Approximation

Algorithm: VNEP Approximation (Profit)

```
// perform preprocessing
```

compute *optimal* LP solution
compute $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$ from LP solution
do
  solution $\leftarrow$ RoundingProcedure($\{\mathcal{D}_r\}_{r \in \mathcal{R}}$)
while (solution *not* (α, β, γ)-approximate and rounding tries not exceeded)

Main Theorem: (FPT-)Approximation for the Virtual Network Embedding Problem

The Algorithm returns (α, β, γ)-approximate solutions for the of at least an α fraction of the optimal profit, and allocations on nodes and edges within factors of β and γ of the original capacities, respectively, *with high probability*.

First (FPT-)Approximation Algorithm for VNEP

Randomized Rounding Approximation

Algorithm: VNEP Approximation (Profit)

```
// perform preprocessing
```
compute *optimal* LP solution
compute $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$ from LP solution
do
  solution ← RoundingProcedure($\{\mathcal{D}_r\}_{r \in \mathcal{R}}$)
while (solution *not* (α, β, γ)-approximate and rounding tries not exceeded)

Definition of Parameters

$\alpha = 1/3$ (relative achieved profit)

$\beta = (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(V_S) \cdot \log(|V_S|)})$ (max node load)

$\gamma = (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(E_S) \cdot \log(|E_S|)})$ (max edge load)

$\varepsilon = \max_{r \in \mathcal{R}, x \in R_S} d_{\max}(r, x)/c_S(x) \leq 1$ (max demand/capacity)

$\Delta(X) = \max_{x \in X} \sum_{r \in \mathcal{R}} (A_{\max}(r, x)/d_{\max}(r, x))^2$ (sum over $\mathcal{R}$ of squared max (total / single) alloc)

Main Theorem: (FPT-)Approximation for the Virtual Network Embedding Problem

The Algorithm returns (α, β, γ)-approximate solutions for the of at least an α fraction of the optimal profit, and allocations on nodes and edges within factors of β and γ of the original capacities, respectively, *with high probability*.

First (FPT-)Approximation Algorithm for VNEP

Randomized Rounding Approximation

Algorithm: VNEP Approximation (Profit)

```
// perform preprocessing
```
compute *optimal* LP solution
compute $\{\mathcal{D}_r\}_{r\in\mathcal{R}}$ from LP solution
do
  solution $\leftarrow$ RoundingProcedure($\{\mathcal{D}_r\}_{r\in\mathcal{R}}$)
while (solution *not* (α,β,γ)-approximate and rounding tries not exceeded)

Definition of Parameters

$\alpha = 1/3$ (relative achieved profit)

$\beta = (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(V_S) \cdot \log(|V_S|)})$ (max node load)

$\gamma = (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(E_S) \cdot \log(|E_S|)})$ (max edge load)

$\varepsilon = \max_{r\in\mathcal{R}, x\in R_S} d_{\max}(r,x)/c_S(x) \leq 1$ (max demand/capacity)

$$\Delta(X) = \max_{x\in X} \sum_{r\in\mathcal{R}} (A_{\max}(r,x)/d_{\max}(r,x))^2$$ (sum over $\mathcal{R}$ of squared max (total / single) alloc)

Applicability in Practice: Computing β and γ is hard ...

- Computing β and γ requires enumerating all valid mappings.
- $\beta \in \mathcal{O}(\varepsilon \cdot \sqrt{|\mathcal{R}| \cdot \max_{r\in\mathcal{R}} |V_r| \cdot \log(|V_S|)})$ and $\gamma \in \mathcal{O}(\varepsilon \cdot \sqrt{|\mathcal{R}| \cdot \max_{r\in\mathcal{R}} |E_r| \cdot \log(|E_S|)})$

First (FPT-)Approximation Algorithm for VNEP

Randomized Rounding Approximation

Algorithm: VNEP Approximation (Profit)

```
// perform preprocessing
```
compute *optimal* LP solution
compute $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$ from LP solution
do
  solution $\leftarrow$ RoundingProcedure($\{\mathcal{D}_r\}_{r \in \mathcal{R}}$)
while (solution *not* (α, β, γ)-approximate and rounding tries not exceeded)

Definition of Parameters

$\alpha = 1/3$ (relative achieved profit)

$\beta = (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(V_S) \cdot \log(|V_S|)})$ (max node load)

$\gamma = (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(E_S) \cdot \log(|E_S|)})$ (max edge load)

$\varepsilon = \max_{r \in \mathcal{R}, x \in R_S} d_{\max}(r,x)/c_S(x) \leq 1$ (max demand/capacity)

$$\Delta(X) = \max_{x \in X} \sum_{r \in \mathcal{R}} (A_{\max}(r,x)/d_{\max}(r,x))^2$$ (sum over $\mathcal{R}$ of squared max (total / single) alloc)

Applicability in Practice: Computing β and γ is hard ...

- Computing β and γ requires enumerating all valid mappings.
- $\beta \in \mathcal{O}(\varepsilon \cdot \sqrt{|\mathcal{R}| \cdot \max_{r \in \mathcal{R}} |V_r| \cdot \log(|V_S|)})$ and $\gamma \in \mathcal{O}(\varepsilon \cdot \sqrt{|\mathcal{R}| \cdot \max_{r \in \mathcal{R}} |E_r| \cdot \log(|E_S|)})$

Consider Heuristics

Return best solution found within X iterations.

Derived Heuristics

Randomized Rounding Approximation

Algorithm: VNEP Approximation

```
// perform preprocessing
```
compute *optimal* LP solution
compute $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$ from LP solution
do
  solution $\leftarrow$ RoundingProcedure($\{\mathcal{D}_r\}_{r \in \mathcal{R}}$)
while (solution *not* (α, β, γ)-approximate and rounding tries not exceeded)

Derived Heuristics

Heuristic Idea: Fixed #Iterations

Algorithm: Heuristic Adaptation

```
// perform preprocessing
```

compute *optimal* LP solution
compute $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$ from LP solution
do
  solution ← RoundingProcedure($\{\mathcal{D}_r\}_{r \in \mathcal{R}}$)
while *rounding tries not exceeded*
return *best* solution

Vanilla Rounding: $\mathrm{RR}_{\mathrm{MinLoad}}$

- still may exceed capacities
- return solution with least resource violations (among those: highest profit)

Derived Heuristics

Heuristic Idea: Fixed #Iterations

Algorithm: Heuristic Adaptation

```
// perform preprocessing
```

compute *optimal* LP solution
compute $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$ from LP solution
do
  solution ← RoundingProcedure($\{\mathcal{D}_r\}_{r \in \mathcal{R}}$)
while *rounding tries not exceeded*
return *best* solution

Algorithm: RoundingProcedure (Heuristic)

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$
foreach $r \in \mathcal{R}$ **do**
  choose m_r^k **with probability** f_r^k
  discard mapping if capacity violated
end
return *solution*

Vanilla Rounding: $\mathrm{RR}_{\mathrm{MinLoad}}$

- still may exceed capacities
- return solution with least resource violations (among those: highest profit)

Heuristic Rounding: $\mathrm{RR}_{\mathrm{Heuristic}}$

- RoundingProcedure:
  discard chosen mappings exceeding capacities
- always yields feasible solutions
- return solution with highest profit

Computational Evaluation[4,5]

[4] Matthias Rost and Stefan Schmid. Virtual Network Embedding Approximations: Leveraging Randomized Rounding. In *Proc. IFIP Networking*, 2018d

[5] Matthias Rost and Stefan Schmid. Virtual Network Embedding Approximations: Leveraging Randomized Rounding. Technical report, March 2018b. URL `http://arxiv.org/abs/1803.03622`

Computational Evaluation: Setup

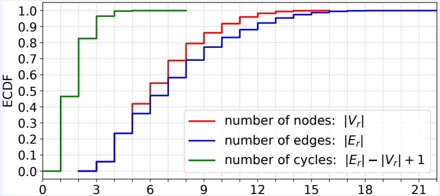
Substrate: GEANT

Code available:
https://github.com/vnep-approx/evaluation-ifip-networking-2018

Generation Parameters for 1,500 instances

Number of requests: 40, 60, 80, 100
Node-Resource Factor (NRF): 0.2, 0.4, 0.6, 0.8, 1.0
Edge-Resource Factor (ERF): 0.25, 0.5, 1.0, 2.0, 4.0
Instances per combination: 15

Requests: Synthetic Cactus Requests



Profit: minimum embedding resource costs

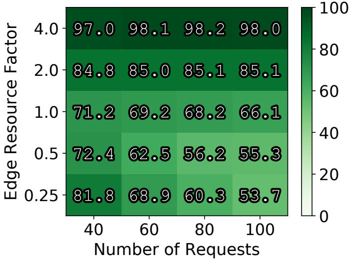
Node mapping restriction: 1/4 substrate nodes

Demands: exp. dist. according to NRF/ERF

Computational Evaluation: Setup

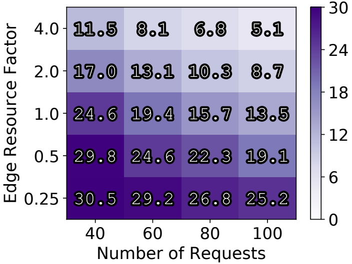
Baseline Algorithm – $\mathrm{MIP}_{\mathrm{MCF}}$: solve classic MIP Formulation for upto 3 hours

Acceptance Ratio



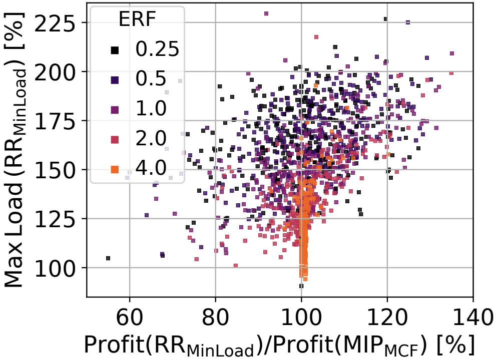
Avg. Node Load

Avg. Edge Load



Computational Evaluation: Heuristic Performance

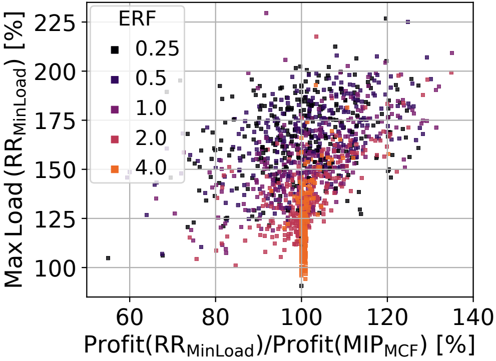
Vanilla Rounding Performance



- Relative profit $\approx$ 80 - 120%
- Resource augmentations mostly $<$ 200%

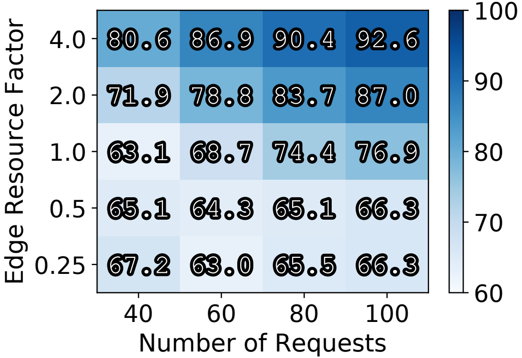
Computational Evaluation: Heuristic Performance

Vanilla Rounding Performance



- Relative profit $\approx$ 80 - 120%
- Resource augmentations mostly $<$ 200%

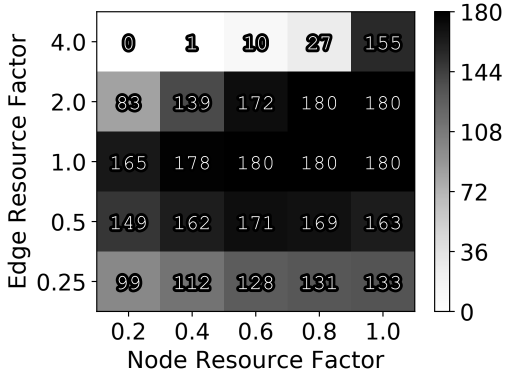
Heuristic Rounding (w/o augmentations)



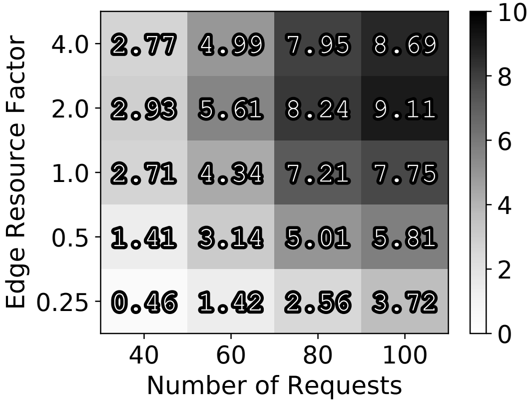
- Relative profit $\approx$ 65 - 90%
- min: 22.5% / mean: 73.8% / max: 101%

Computational Evaluation: Runtimes

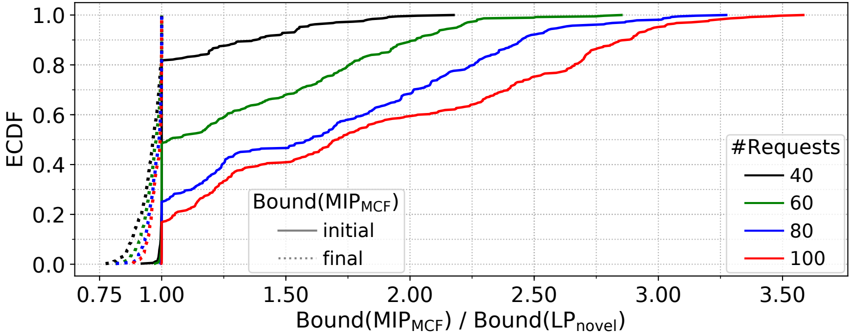
Runtime $\mathrm{MIP}_{\mathrm{MCF}}$ (Gurobi 7.5.1)



Runtime $\mathrm{LP}_{\mathrm{novel}}$ (Gurobi 7.5.1)



Computational Evaluation: Formulation Strengths



- Root relaxation values upto 3.5 times better than when using the MCF LP.
- Final MIP bounds improve novel LP bounds by at most a factor of 1.3.

Conclusion

Conclusion

Summary

Complexity: Computing valid mappings is $\mathcal{NP}$-complete for planar graphs.

(FPT-)Linear Programs: Valid mappings can be computed in FPT using novel LP.

(FPT-)Approximations: For offline VNEP (profit & cost) based on randomized rounding.

Evaluation:
- Solutions quite good even without resource augmentations.
- Novel formulation is much stronger.
- Runtime becomes an issue.

Conclusion

Summary

Complexity: Computing valid mappings is $\mathcal{NP}$-complete for planar graphs.

(FPT-)Linear Programs: Valid mappings can be computed in FPT using novel LP.

(FPT-)Approximations: For offline VNEP (profit & cost) based on randomized rounding.

Evaluation:
- Solutions quite good even without resource augmentations.
- Novel formulation is much stronger.
- Runtime becomes an issue.

Future Work

Runtime: Column generation could be readily applied, need to try it.

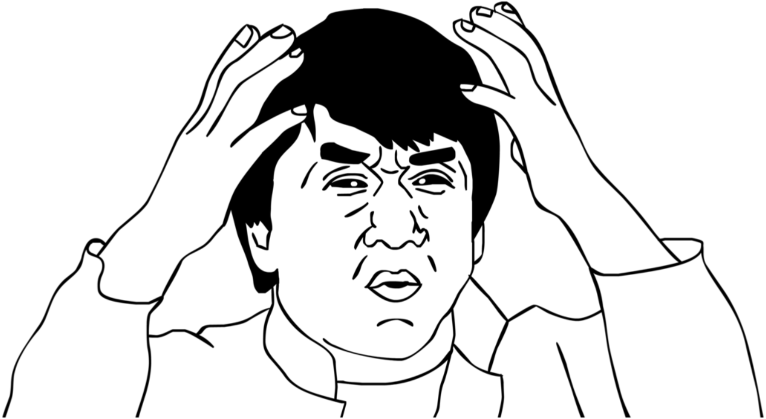
Heuristics: Many possibilities, also for online problem.

Extraction width: Can improve the formulation further ($\rightarrow$ tree-width).

Online Approximation: Need to improve rounding scheme (using e.g. Bansal et al. [2011]).

Thank You!

Questions?



References I

Edoardo Amaldi, Stefano Coniglio, Arie M.C.A. Koster, and Martin Tieves. On the computational complexity of the virtual network embedding problem. *Electronic Notes in Discrete Mathematics*, 52:213 – 220, 2016.

David G. Andersen. Theoretical approaches to node assignment. [Online]. Available: `http://repository.cmu.edu/compsci/86/`, December 2002.

Nikhil Bansal, Kang-Won Lee, Viswanath Nagarajan, and Murtaza Zafer. Minimum congestion mapping in a cloud. In *Proc. ACM PODC*, 2011.

N. Chowdhury, M.R. Rahman, and R. Boutaba. Virtual network embedding with coordinated node and link mapping. In *Proc. IEEE INFOCOM*, 2009.

Guy Even, Matthias Rost, and Stefan Schmid. An approximation algorithm for path computation and function placement in sdns. In Jukka Suomela, editor, *Structural Information and Communication Complexity*. Springer, 2016.

References II

Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

Jan Kratochvíl. A special planar satisfiability problem and a consequence of its np-completeness. *Discrete Applied Mathematics*, 52(3), 1994.

Matthias Rost and Stefan Schmid. (FPT-)Approximation Algorithms for the Virtual Network Embedding Problem. Technical report, March 2018a. URL `http://arxiv.org/abs/1803.04452`.

Matthias Rost and Stefan Schmid. Virtual Network Embedding Approximations: Leveraging Randomized Rounding. Technical report, March 2018b. URL `http://arxiv.org/abs/1803.03622`.

Matthias Rost and Stefan Schmid. Charting the Complexity Landscape of Virtual Network Embeddings. In *Proc. IFIP Networking*, 2018c.

Matthias Rost and Stefan Schmid. Virtual Network Embedding Approximations: Leveraging Randomized Rounding. In *Proc. IFIP Networking*, 2018d.