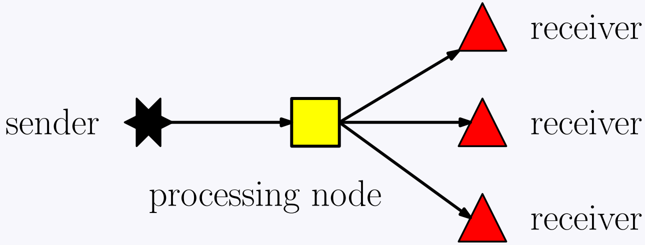
A Compact MIP for Aggregation and Multicast Trees under Flexible Routing and Function Placement

Matthias Rost, Technische Universität Berlin
joint work with Stefan Schmid, T-Labs & TU Berlin

July 17th, 2015
ISMP 2015, Pittsburgh

Communication Schemes: Multicast (same old! same old?)

processing = duplication + reroute



Communication Schemes: Multicast (same old! same old?)

processing = duplication + reroute

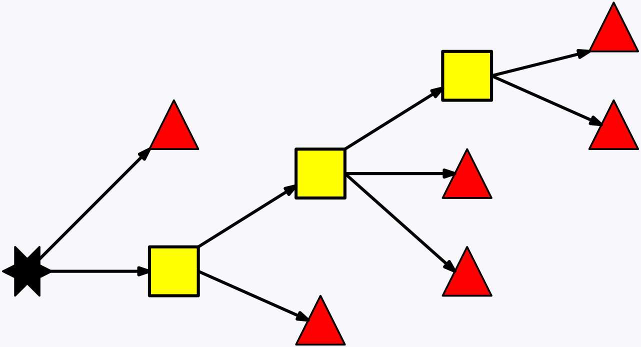
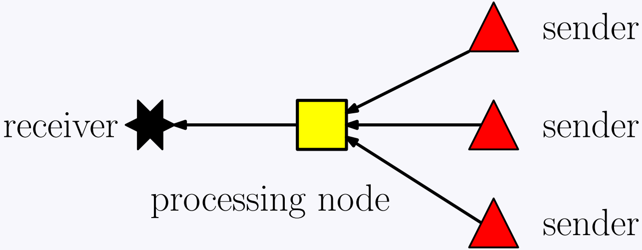


Figure: Hierarchy of processing nodes

Communication Schemes: Aggregation

processing = merge + reroute



Communication Schemes: Aggregation

processing = merge + reroute

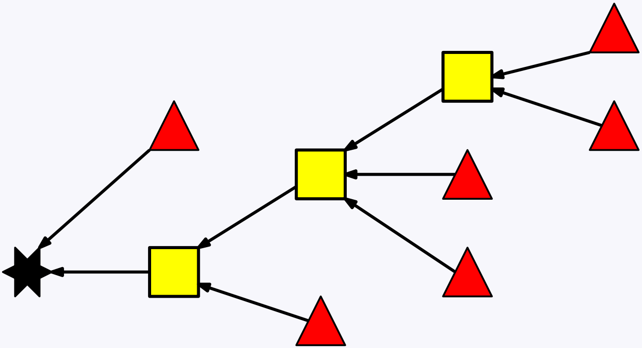


Figure: Hierarchy of processing nodes

Problem Statement

Setting: Network Virtualization

- (Unsplittable) routes can be established arbitrarily (e.g. using Software-Defined Networks)
- Processing functionality can be placed on specific nodes (e.g. using Network Functions Virtualization)

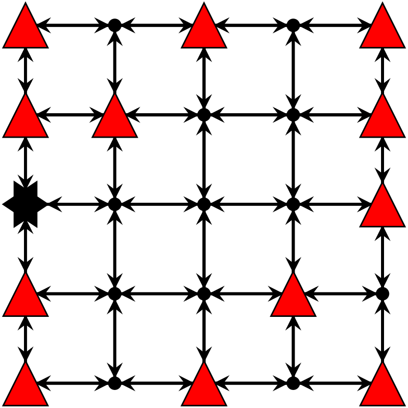
Main Questions

- How to compute *virtual* aggregation / multicasting trees?
  - Where to place in-network processing functionality?
  - How to trade-off between traffic and processing?

Introductory Example

Aggregation scenario

- bi-directed grid graph



Without in-network processing: Unicast

Solution Method

- minimal cost flow

Solution uses

- 41 edges
- 0 processing nodes

✱ receiver △ sender

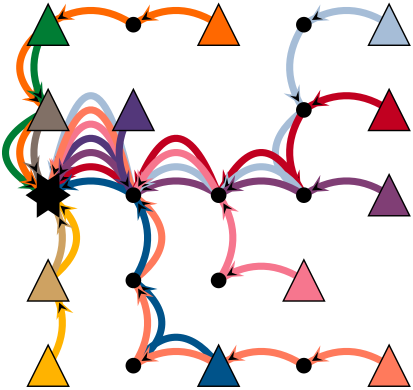


Figure: Unicast solution

With in-network processing at all nodes

Solution Method

- Steiner arborescence

Solution uses

- 16 edges
- 9 processing nodes

receiver | sender
processing node | sender with processing

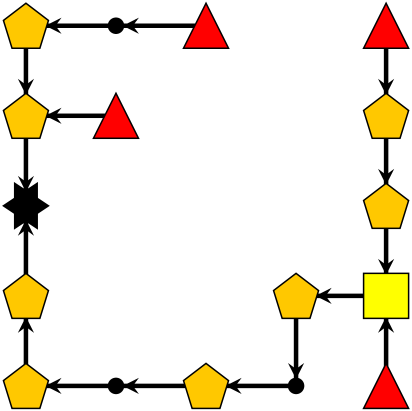
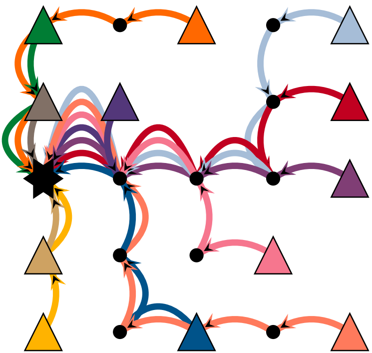
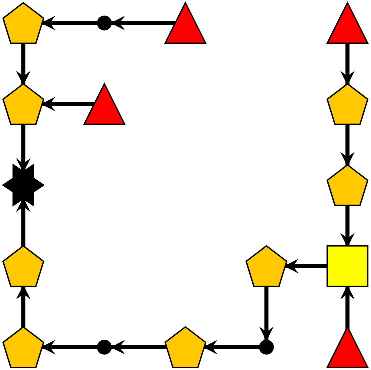


Figure: Aggregation tree

How to Trade-off?



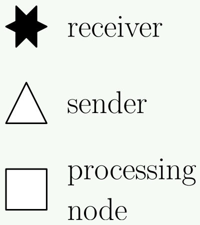
vs.

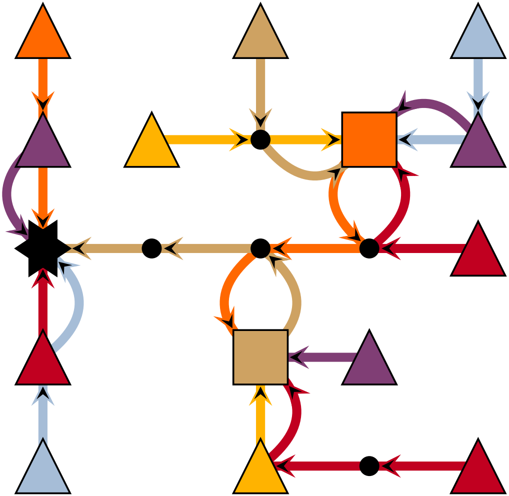


What we aim for

Solution uses

- 26 edges
- 2 processing nodes





Solution Structure

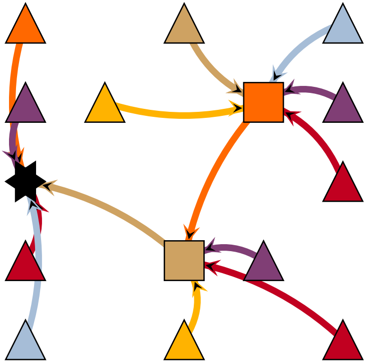


Figure: Virtual Arborescence

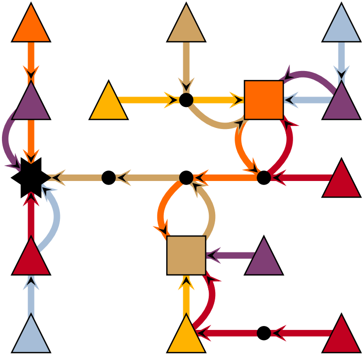


Figure: underlying routes

Input

Definition (Network $G = (V_G, E_G, c_E, u_E)$)

- integral capacities on the edges $u_E : E_G \to \mathbb{N}$
- positive edge costs $c_E : E_G \to \mathbb{R}^+$

Definition (Abstract Communication Request)

An abstract communication request on a graph G is defined as a 5-tuple $R_G = (r, S, T, u_r, c_S, u_S)$, where

- $T \subseteq V_G$ is the set of terminals,
- $r \in V_G \setminus T$ denotes the root with integral capacity $u_r \in \mathbb{N}$ and
- $S \subseteq V_G \setminus (\{r\} \cup T)$ denotes the set of possible *Steiner sites* with associated activation costs $c_S : S \to \mathbb{R}^+$ and integral capacities $u_S : S \to \mathbb{N}$.

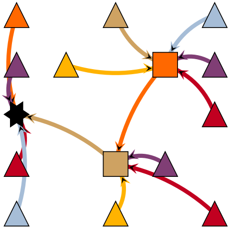
Virtual Arborescence

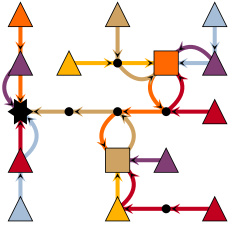
Definition (Virtual Arborescence on G: $\mathcal{T}_G = (V_\mathcal{T}, E_\mathcal{T}, r, \pi)$)

- $\{r\} \subseteq V_\mathcal{T} \subseteq V_G$ and $E_\mathcal{T} \subseteq V_\mathcal{T} \times V_\mathcal{T}$
- $\pi : E_\mathcal{T} \to \mathcal{P}_G$ maps each edge of $E_\mathcal{T}$ on a (simple) path $P \in \mathcal{P}_G$, s.t.

  (VA-1) $(V_\mathcal{T}, E_\mathcal{T}, r)$ is an rooted arborescence with edges either directed towards or away from r,

  (VA-2) for all $(u, v) \in E_\mathcal{T}$ the directed path $\pi(u, v)$ connects u to v in G.





Definition (Constrained Virtual Steiner Arborescence Problem)

Input: network $G = (V_G, E_G, c_E, u_E)$, request $R_G = (r, S, T, u_r, c_S, u_S)$.
Task: Find a minimal cost Virtual Arborescence $\mathcal{T}_G = (V_\mathcal{T}, E_\mathcal{T}, r, \pi)$ satisfying:

(CVSAP-1) $\{r\} \cup T \subseteq V_\mathcal{T}$ and $V_\mathcal{T} \subseteq \{r\} \cup S \cup T$,

(CVSAP-2) for all $t \in T$ holds $\delta^+_{E_\mathcal{T}}(t) + \delta^-_{E_\mathcal{T}}(t) = 1$,

(CVSAP-3) for the root $\delta^+_{E_\mathcal{T}}(r) + \delta^-_{E_\mathcal{T}}(r) \leq u_r$ holds,

(CVSAP-4) for all $s \in S \cap V_\mathcal{T}$ holds $\delta^-_{E_\mathcal{T}}(s) + \delta^+_{E_\mathcal{T}}(s) \leq u_S(s) + 1$ and

(CVSAP-5) for all $e \in E_G$ holds $|\left(\pi(E_\mathcal{T})\right)[e]| \leq u_E(e)$.

The cost of a Virtual Arborescence is defined to be

$$C_{\text{CVSAP}}(\mathcal{T}_G) = \sum_{e \in E_G} c_E(e) \cdot |\left(\pi(E_\mathcal{T})\right)[e]| + \sum_{s \in S \cap V_\mathcal{T}} c_S(s) \ ,$$

where $|\left(\pi(E_\mathcal{T})\right)[e]|$ denotes the number of times an edge is used.

Applications

Applications

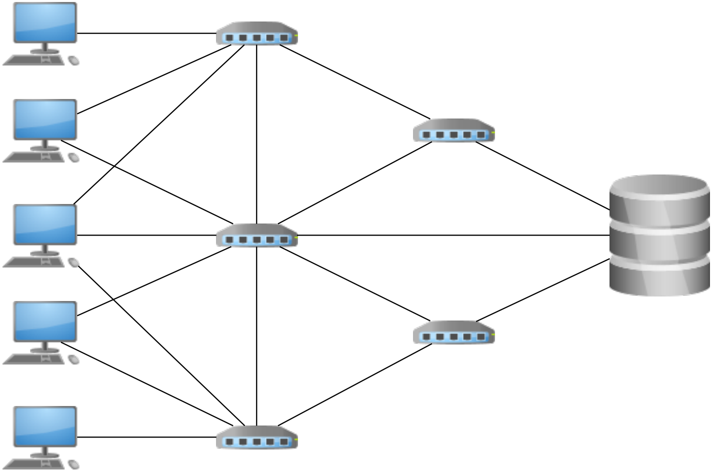
	Network	Application	Technology, e.g.
multicast	ISP	service replication / cache placement [10, 11]	middleboxes / NFV + SDN
	backbone	optical multicast [6]	ROADM + SDH
	all	application-level multicast [16]	different
aggregation	sensor network	value & message aggregation [5, 8]	source routing
	ISP	network analytics: Gigascope [3]	middleboxes / NFV + SDN
	data center	big data / map-reduce: Camdoop [2]	SDN

edge capacities

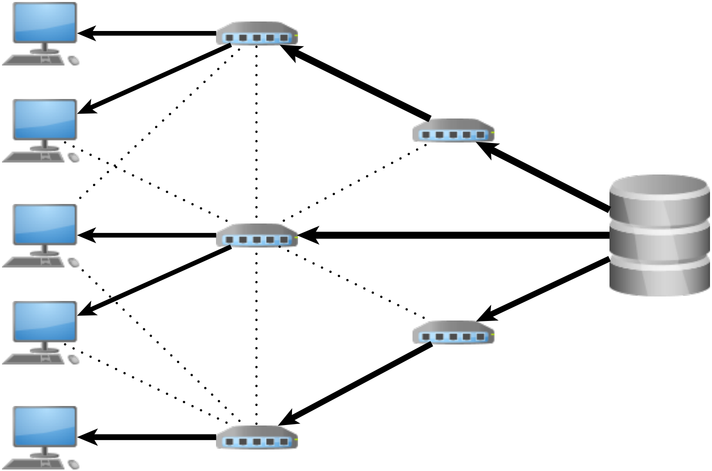
processing node locations

processing node capacities

Service Replication

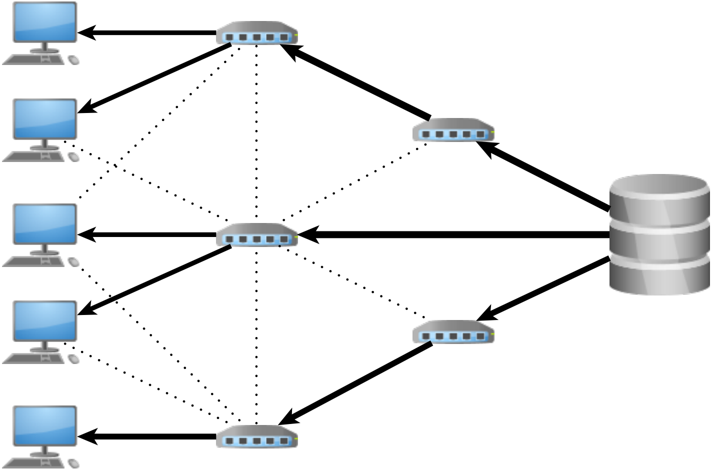


Service Replication

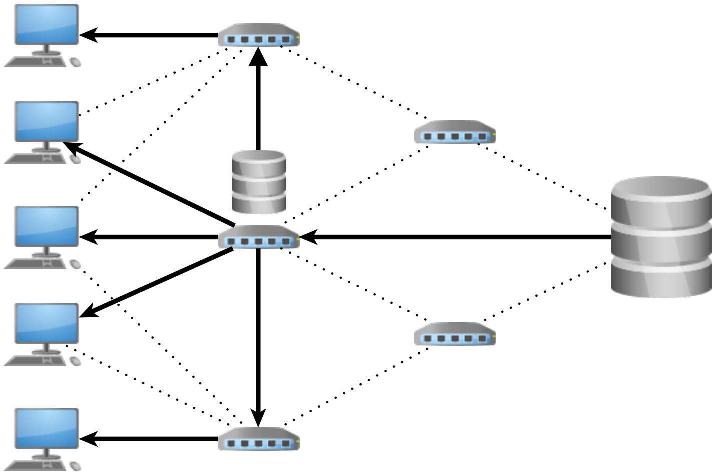


Service Replication

What if backend links are congested?

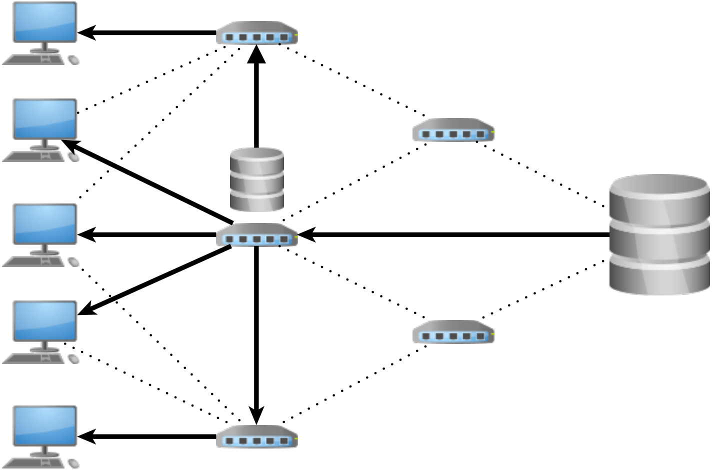


Service Replication

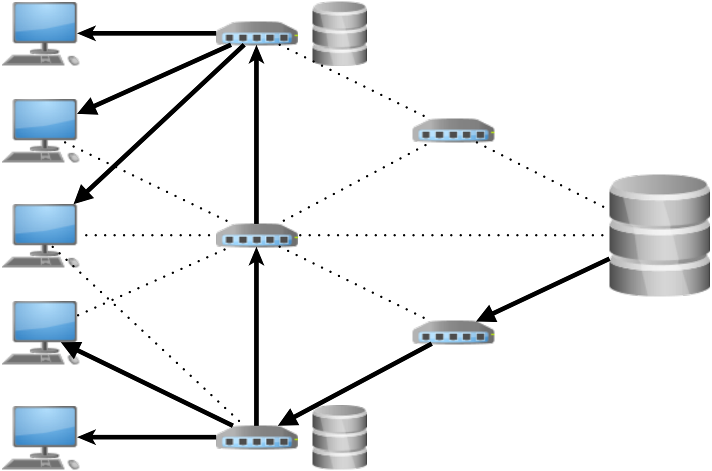


Service Replication

What if only ‘3’ users can be handled?



Service Replication



Solution Approaches

Comprehensive algorithmic study

Computational Complexity (Inapproximability)

Approximation Algorithms

Exact Algorithms (MIPs)

LP-based Heuristics

M.Sc. Thesis [13] *Matthias Rost (Advisors: Schmid, Bley, Feldmann)*
Optimal Virtualized In-Network Processing with Applications to Aggregation and Multicast, TU Berlin '14

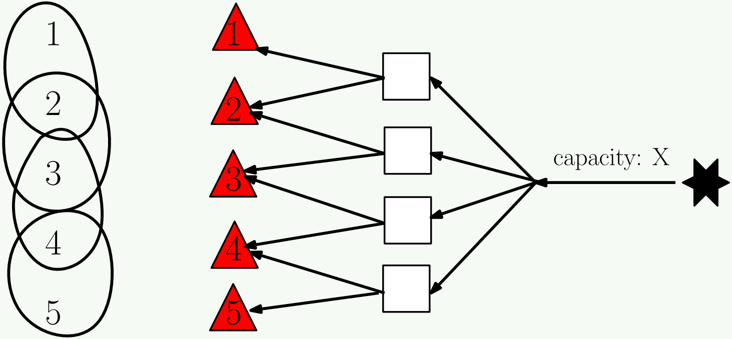
Conference [15] *Matthias Rost and Stefan Schmid*
VirtuCast, Multicast and Aggregation with In-Network Processing, OPODIS '13

Tech. Report [14] *Matthias Rost and Stefan Schmid*
The Constrained Virtual Steiner Arborescence Problem: Formal Definition, Single-Commodity Integer Programming Formulation and Computational Evaluation, arXiv '13

Inapproximability

Inapproximability

Reduction via Set Cover: Does a set cover of size X exist?



Theorem

Finding *a feasible* solution is already NP-complete.

Approximation Algorithms for Variants

Variants

$$\begin{tikzcd}
 & \textbf{Directed} & \textbf{Undirected} \\
\textbf{edge and node capacities} & \mathrm{CVSAP} & \mathrm{CVSTP} \arrow[l] \\
\textbf{node capacities} & \mathrm{NVSAP} \arrow[u] & \mathrm{NVSTP} \arrow[l] \arrow[u] \\
\textbf{no capacities} & \mathrm{VSAP} \arrow[u] & \mathrm{VSTP} \arrow[l] \arrow[u]
\end{tikzcd}$$

Approximation via related problems

Results

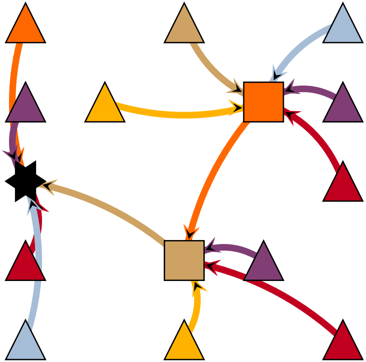
$$
\begin{tikzcd}
 & & \textbf{Directed} & \textbf{Undirected} & \\
\textbf{both capacities} & & \mathrm{CVSAP} & \mathrm{CVSTP} \arrow[l] & \\
\textbf{node capacities} & & \mathrm{NVSAP} \arrow[u] & \mathrm{NVSTP} \arrow[l] \arrow[u] \arrow[r, dashed, "\mathcal{O}(\log, \log)"] & \mathrm{DNSTP} \\
\textbf{no capacities} & \mathrm{SAP} \arrow[r, leftrightarrow, "\mathcal{O}(\log)"] & \mathrm{VSAP} \arrow[u] & \mathrm{VSTP} \arrow[l] \arrow[u] \arrow[r, dashed, "8"] & \mathrm{CFLP}
\end{tikzcd}
$$

Exact Algorithms for CVSAP

Multi-Commodity Flow (MCF) Integer Program

First approach: MCF IP

- explicitly represent virtual arborescence
- necessitates independent construction of paths for all processing nodes



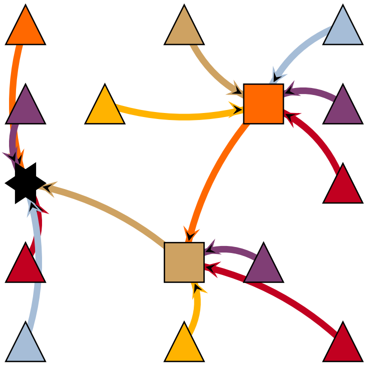
Multi-Commodity Flow (MCF) Integer Program

First approach: MCF IP

- explicitly represent virtual arborescence
- necessitates independent construction of paths for all processing nodes

Intuition: does not scale well

- number of binary variables: #Steiner sites · #edges



Integer Program 1: A-CVSAP-MCF

$$
\begin{aligned}
\text{minimize} \quad & C_{\mathsf{MCF}} = \sum_{e \in E_G} \mathbf{c}_e (f_e + \sum_{s \in S} f_{s,e}) & \text{(MCF-OBJ)} \\
& \quad + \sum_{s \in S} \mathbf{c}_s \cdot x_s & \\
\text{subject to} \quad & f^T(\delta^+_{E_{\mathsf{MCF}}}(v)) = f^T(\delta^-_{E_{\mathsf{MCF}}}(v)) + |\{v\} \cap T| \qquad \forall\, v \in V_G & \text{(MCF-1)} \\
& f^s(\delta^+_{E^s_{\mathsf{MCF}}}(v)) = f^s(\delta^-_{E^s_{\mathsf{MCF}}}(v)) + \delta_{s,v} \cdot x_s \qquad \forall\, s \in S, v \in V_G & \text{(MCF-2)} \\
& f^T_e + \sum_{s \in S} f^s_e \leq \begin{cases} \mathbf{u}_s x_s, & e = (s, \mathsf{o}^-), s \in S \\ \mathbf{u}_r \quad , & e = (r, \mathsf{o}^-) \\ \mathbf{u}_e \quad , & e \in E_G \end{cases} \qquad \forall e \in E_{\mathsf{MCF}} & \text{(MCF-3)} \\
& -|S|(1 - f^s_{\bar{s},\mathsf{o}^-}) \leq p_s - p_{\bar{s}} - 1 \qquad \forall\, s, \bar{s} \in S & \text{(MCF-4)} \\
& f^s_{(\bar{s},\mathsf{o}^-)} \leq x_{\bar{s}} \qquad \forall\, s \in S, \bar{s} \in S - s & \text{(MCF-5}^\star\text{)} \\
& f^s_{s,\mathsf{o}^-} = 0 \qquad \forall\, s \in S & \text{(MCF-6}^\star\text{)} \\
& f^s_{\bar{s},\mathsf{o}^-} + f^{\bar{s}}_{s,\mathsf{o}^-} \leq 1 \qquad \forall\, s, \bar{s} \in S & \text{(MCF-7}^\star\text{)} \\
& x_s \in \{0,1\} \qquad \forall\, s \in S & \text{(MCF-8)} \\
& f^T_e \in \mathbb{Z}_{\geq 0} \qquad \forall\, e \in E_{\mathsf{MCF}} & \text{(MCF-9)} \\
& f^s_e \in \{0,1\} \qquad \forall\, s \in S, e \in E_{\mathsf{MCF}} & \text{(MCF-10)} \\
& p \in [0, |S| - 1] \qquad \forall\, s \in S & \text{(MCF-11)}
\end{aligned}
$$

Integer Program 2: A-CVSAP-MCF

$$\text{minimize} \quad C_{\mathrm{MCF}} = \sum_{e \in E_G} \mathbf{c}_e \Big(f_e + \sum_{s \in S} f_{s,e}\Big) + \sum_{s \in S} \mathbf{c}_s \cdot x_s \tag{MCF-OBJ}$$

subject to

$$f^T(\delta^+_{E_{\mathrm{MCF}}}(v)) = f^T(\delta^-_{E_{\mathrm{MCF}}}(v)) + |\{v\} \cap T| \qquad \forall\, v \in V_G \tag{MCF-1}$$

$$f^s(\delta^+_{E^S_{\mathrm{MCF}}}(v)) = f^s(\delta^-_{E^S_{\mathrm{MCF}}}(v)) + \delta_{s,v} \cdot x_s \qquad \forall\, s \in S, v \in V_G \tag{MCF-2}$$

$$f^T_e + \sum_{s \in S} f^s_e \leq \begin{cases} \mathbf{u}_s x_s, & e = (s, \mathrm{o}^-), s \in S \\ \mathbf{u}_r &, e = (r, \mathrm{o}^-) \\ \mathbf{u}_e &, e \in E_G \end{cases} \qquad \forall e \in E_{\mathrm{MCF}} \tag{MCF-3}$$

$$-|S|(1 - f^s_{\bar{s},\mathrm{o}^-}) \leq p_s - p_{\bar{s}} - 1 \qquad \forall\, s, \bar{s} \in S \tag{MCF-4}$$

$$f^s_{(\bar{s},\mathrm{o}^-)} \leq x_{\bar{s}} \qquad \forall\, s \in S, \bar{s} \in S - s \tag{MCF-5*}$$

$$f^s_{s,\mathrm{o}^-} = 0 \qquad \forall\, s \in S \tag{MCF-6*}$$

$$f^s_{\bar{s},\mathrm{o}^-} + f^{\bar{s}}_{s,\mathrm{o}^-} \leq 1 \qquad \forall\, s, \bar{s} \in S \tag{MCF-7*}$$

$$x_s \in \{0,1\} \qquad \forall\, s \in S \tag{MCF-8}$$

$$f^T_e \in \mathbb{Z}_{\geq 0} \qquad \forall\, e \in E_{\mathrm{MCF}} \tag{MCF-9}$$

$$f^s_e \in \{0,1\} \qquad \forall\, s \in S, e \in E_{\mathrm{MCF}} \tag{MCF-10}$$

$$p \in [0, |S| - 1] \qquad \forall\, s \in S \tag{MCF-11}$$

Integer Program 3: A-CVSAP-MCF

$$\text{minimize} \quad C_{\text{MCF}} = \sum_{e \in E_G} \mathbf{c}_e (f_e + \sum_{s \in S} f_{s,e}) + \sum_{s \in S} \mathbf{c}_s \cdot x_s \qquad \text{(MCF-OBJ)}$$

$$\text{subject to} \quad f^T(\delta^+_{E_{\text{MCF}}}(v)) = f^T(\delta^-_{E_{\text{MCF}}}(v)) + |\{v\} \cap T| \qquad \forall\, v \in V_G \qquad \text{(MCF-1)}$$

$$f^s(\delta^+_{E^s_{\text{MCF}}}(v)) = f^s(\delta^-_{E^s_{\text{MCF}}}(v)) + \delta_{s,v} \cdot x_s \qquad \forall\, s \in S, v \in V_G \qquad \text{(MCF-2)}$$

$$f^T_e + \sum_{s \in S} f^s_e \leq \begin{cases} \mathbf{u}_s x_s, & e = (s, \text{o}^-), s \in S \\ \mathbf{u}_r, & e = (r, \text{o}^-) \\ \mathbf{u}_e, & e \in E_G \end{cases} \qquad \forall e \in E_{\text{MCF}} \qquad \text{(MCF-3)}$$

$$-|S|(1 - f^s_{\bar{s},\text{o}^-}) \leq p_s - p_{\bar{s}} - 1 \qquad \forall\, s, \bar{s} \in S \qquad \text{(MCF-4)}$$

$$f^s_{(\bar{s},\text{o}^-)} \leq x_{\bar{s}} \qquad \forall\, s \in S, \bar{s} \in S - s \qquad \text{(MCF-5}^\star\text{)}$$

$$f^s_{s,\text{o}^-} = 0 \qquad \forall\, s \in S \qquad \text{(MCF-6}^\star\text{)}$$

$$f^s_{\bar{s},\text{o}^-} + f^{\bar{s}}_{s,\text{o}^-} \leq 1 \qquad \forall\, s, \bar{s} \in S \qquad \text{(MCF-7}^\star\text{)}$$

$$x_s \in \{0, 1\} \qquad \forall\, s \in S \qquad \text{(MCF-8)}$$

$$f^T_e \in \mathbb{Z}_{\geq 0} \qquad \forall\, e \in E_{\text{MCF}} \qquad \text{(MCF-9)}$$

$$f^s_e \in \{0, 1\} \qquad \forall\, s \in S, e \in E_{\text{MCF}} \qquad \text{(MCF-10)}$$

$$p \in [0, |S| - 1] \qquad \forall\, s \in S \qquad \text{(MCF-11)}$$

Integer Program 4: A-CVSAP-MCF

$$
\begin{aligned}
\text{minimize} \quad & C_{\text{MCF}} = \sum_{e \in E_G} \mathbf{c}_e (f_e + \sum_{s \in S} f_{s,e}) && \text{(MCF-OBJ)} \\
& \quad + \sum_{s \in S} \mathbf{c}_s \cdot x_s && \\
\text{subject to} \quad & f^T(\delta^+_{E_{\text{MCF}}}(v)) = f^T(\delta^-_{E_{\text{MCF}}}(v)) + |\{v\} \cap T| && \forall\, v \in V_G \quad \text{(MCF-1)} \\
& f^s(\delta^+_{E^s_{\text{MCF}}}(v)) = f^s(\delta^-_{E^s_{\text{MCF}}}(v)) + \delta_{s,v} \cdot x_s && \forall\, s \in S, v \in V_G \quad \text{(MCF-2)} \\
& f^T_e + \sum_{s \in S} f^s_e \leq \begin{cases} \mathbf{u}_s x_s, & e = (s, o^-), s \in S \\ \mathbf{u}_r \quad , & e = (r, o^-) \\ \mathbf{u}_e \quad , & e \in E_G \end{cases} && \forall e \in E_{\text{MCF}} \quad \text{(MCF-3)} \\
& -|S|(1 - f^s_{\bar{s},o^-}) \leq p_s - p_{\bar{s}} - 1 && \forall\, s, \bar{s} \in S \quad \text{(MCF-4)} \\
& f^s_{(\bar{s},o^-)} \leq x_{\bar{s}} && \forall\, s \in S, \bar{s} \in S - s \quad \text{(MCF-5}^\star) \\
& f^s_{s,o^-} = 0 && \forall\, s \in S \quad \text{(MCF-6}^\star) \\
& f^s_{\bar{s},o^-} + f^{\bar{s}}_{s,o^-} \leq 1 && \forall\, s, \bar{s} \in S \quad \text{(MCF-7}^\star) \\
& x_s \in \{0, 1\} && \forall\, s \in S \quad \text{(MCF-8)} \\
& f^T_e \in \mathbb{Z}_{\geq 0} && \forall\, e \in E_{\text{MCF}} \quad \text{(MCF-9)} \\
& f^s_e \in \{0, 1\} && \forall\, s \in S, e \in E_{\text{MCF}} \quad \text{(MCF-10)} \\
& p \in [0, |S| - 1] && \forall\, s \in S \quad \text{(MCF-11)}
\end{aligned}
$$

Single-Commodity Flow IP

Single-commodity flow formulation

- computes *aggregated* flow on edges independently of the origin
- does not represent virtual arborescence

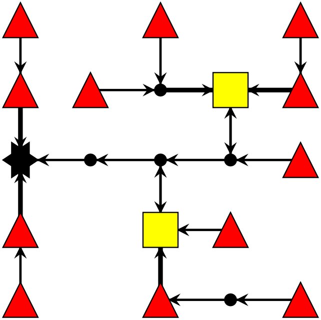


Figure: Single-commodity

Multi- vs Single-Commodity

Example: 6000 edges and 200 Steiner sites

- Single-commodity: 6000 integer variables
- Multi-commodity: 1,200,000 binary variables

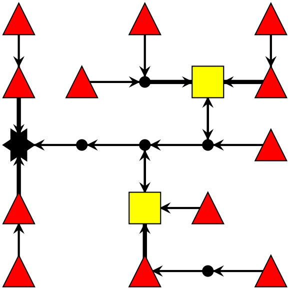


Figure: Single-commodity

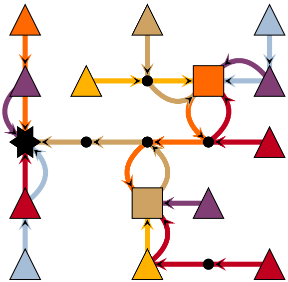


Figure: Multi-commodity

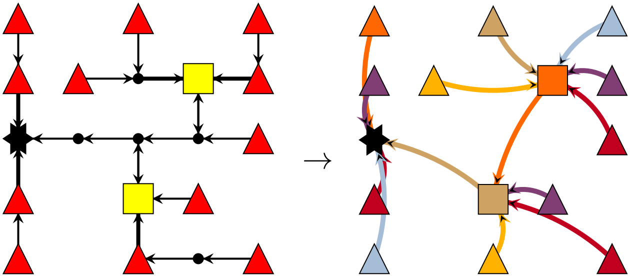
VirtuCast Algorithm

VirtuCast Algorithm

Outline of VirtuCast

1. Solve single-commodity flow IP formulation.
2. Decompose IP solution into Virtual Arborescence.

How to decompose?



(a) IP solution

(b) Virtual Arborescence

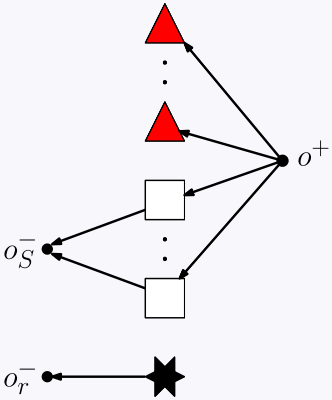
IP Formulation

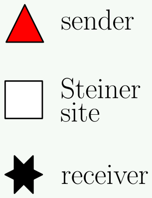
Extended Graph

Additional nodes

- source o^+
- sinks o_r^- and o_S^-

Additional edges





IP Formulation I

$$
\begin{aligned}
\text{minimize} \quad & C_{\mathrm{IP}}(x,f) = \sum_{e \in E_G} \mathbf{c}_e f_e + \sum_{s \in S} \mathbf{c}_s x_s & \\
\text{subject to} \quad & f(\delta^+_{E_{\mathrm{ext}}}(v)) = f(\delta^-_{E_{\mathrm{ext}}}(v)) & \forall\ v \in V_G \\
& f(\delta^+_{E^R_{\mathrm{ext}}}(W)) \geq x_s & \forall\ W \subseteq V_G, s \in W \cap S \neq \emptyset \\
& f_e = 1 & \forall\ e = (\mathrm{o}^+, t) \in E^{T^+}_{\mathrm{ext}} \\
& f_e = x_s & \forall\ e = (\mathrm{o}^+, s) \in E^{S^+}_{\mathrm{ext}} \\
& x_s \in \{0,1\} & \forall\ s \in S \\
& f_e \in \mathbb{Z}_{\geq 0} & \forall\ e \in E_{\mathrm{ext}}
\end{aligned}
$$

Connectivity Inequalities

STP Excursion [7]

$$
\text{(uSP)}\quad
\begin{array}{rll}
\min & c^T x & \\
(i) & x(\delta(W)) \geq 1, & \text{for all } W \subset V, W \cap T \neq \emptyset, \\
 & & (V \setminus W) \cap T \neq \emptyset, \\
(ii) & 0 \leq x_e \leq 1, & \text{for all } e \in E, \\
(iii) & x \text{ integer}, &
\end{array}
$$

Connectivity Inequalities

STP Excursion [7]

$$
(\text{uSP}) \quad
\begin{array}{rll}
\min & c^T x & \\
(i) & x(\delta(W)) \geq 1, & \text{for all } W \subset V, W \cap T \neq \emptyset, \\
 & & (V \setminus W) \cap T \neq \emptyset, \\
(ii) & 0 \leq x_e \leq 1, & \text{for all } e \in E, \\
(iii) & x \text{ integer}, &
\end{array}
$$

$\forall\ W \subseteq V_G, s \in W \cap S \neq \emptyset.\ \ f(\delta^+_{E^R_{\text{ext}}}(W)) \geq x_s$

'*From each activated Steiner site, there exists a path towards* o_r^-.'

Exponentially many constraints, but ...

... can be separated in polynomial time.

Complete Formulation

$$
\begin{array}{lrll}
\text{minimize} & C_{\mathrm{IP}}(x,f) & = \sum_{e \in E_G} \mathbf{c}_e f_e + \sum_{s \in S} \mathbf{c}_s x_s & \\
\text{subject to} & f(\delta^+_{E_{\text{ext}}}(v)) & = f(\delta^-_{E_{\text{ext}}}(v)) & \forall\, v \in V_G \\
& f(\delta^+_{E^R_{\text{ext}}}(W)) & \geq x_s & \forall\, W \subseteq V_G, s \in W \cap S \neq \emptyset \\
& f_e & \leq \mathbf{u}_s x_s & \forall\, e = (s, \mathrm{o}^-_S) \in E^{S^-}_{\text{ext}} \\
& f_{(r,\mathrm{o}^-_r)} & \leq \mathbf{u}_r & \\
& f_e & \leq \mathbf{u}_e & \forall\, e \in E_G \\
& f_e & = 1 & \forall\, e \in E^{T^+}_{\text{ext}} \\
& f_e & = x_s & \forall\, e = (\mathrm{o}^+, s) \in E^{S^+}_{\text{ext}} \\
& x_s & \in \{0,1\} & \forall\, s \in S \\
& f_e & \in \mathbb{Z}_{\geq 0} & \forall\, e \in E_{\text{ext}}
\end{array}
$$

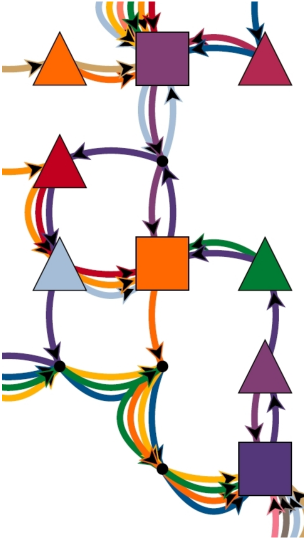
Decomposing flow is non-trivial!

Flow solution . . .

- contains cycles and
- represents *arbitrary* hierarchies.

However, . . .

- decomposition is *always* feasible
- constructive proof:
  polynomial time algorithm



Outline of Decomposition Algorithm

Decomposition Approach

1. select a terminal $t \in T$
2. construct path P from t towards o_r^-
3. reduce flow along edges in P, ***s.t.*** *connectivity inequalities are valid*
4. connect t to the second last node of P and remove t

Outline of Decomposition Algorithm

Reduced problem must be feasible

Removing flow must not invalidate any connectivity inequalities.

Principle: Repair & Redirect

- decrease flow on path edge by edge
- if connectivity inequalities are violated

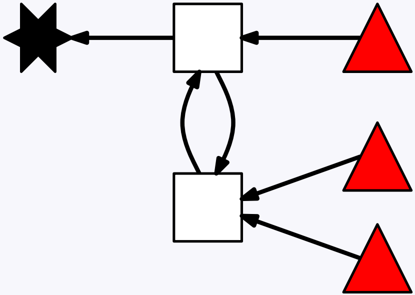
  repair: increment flow on edge to regain feasibility
  redirect: choose a different path from current node

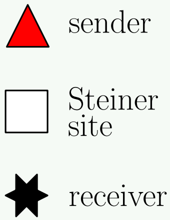
Theorem

Given an optimal solution, the Decomposition Algorithm computes a Virtual Arborescence in time $\mathcal{O}\left(|V_G|^2 \cdot |E_G| \cdot (|V_G| + |E_G|)\right)$.

Example

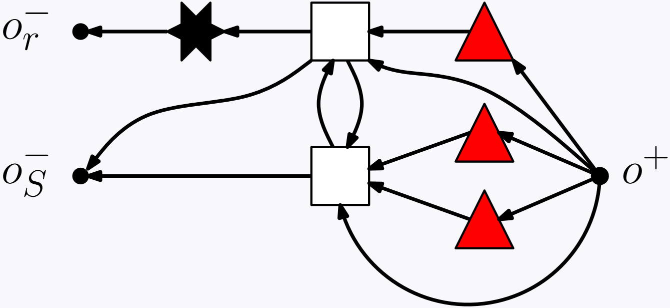
Scenario

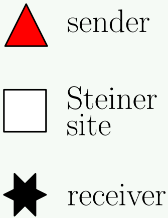




Example

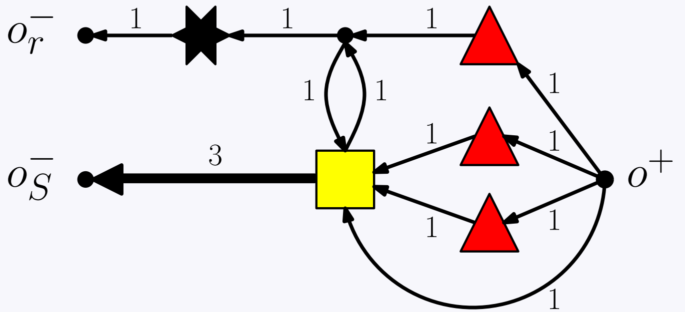
Extended Graph

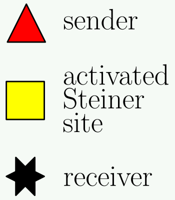




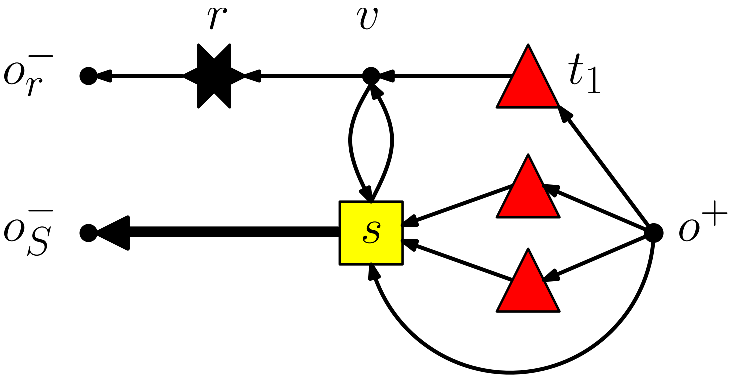
Example

Solution



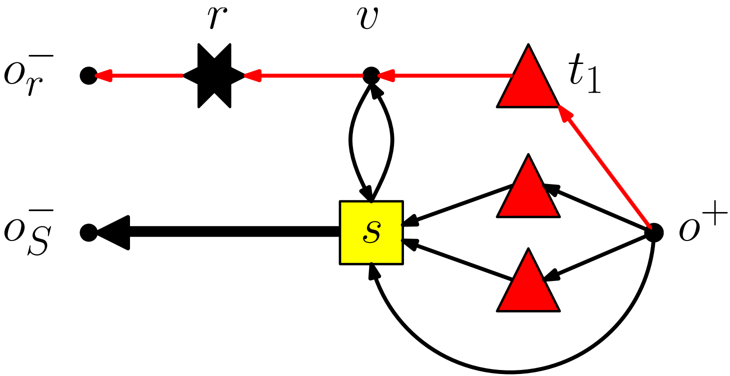


Decomposition Example I



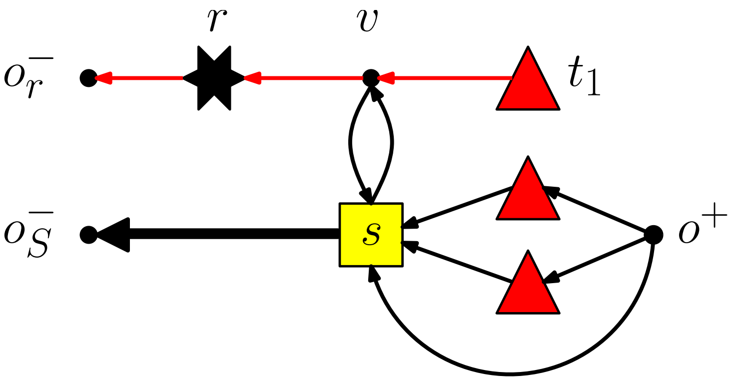
Decomposition Example I

$$P = \langle o^+, t_1, v, r, o_r^- \rangle$$



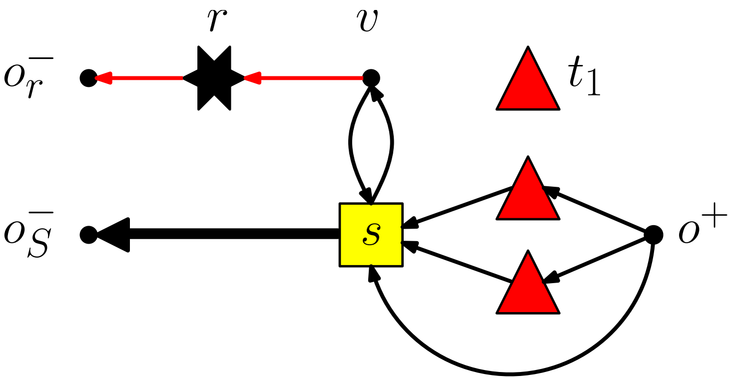
Decomposition Example I

$$P = \langle \mathrm{o}^+, t_1, v, r, \mathrm{o}_r^- \rangle$$



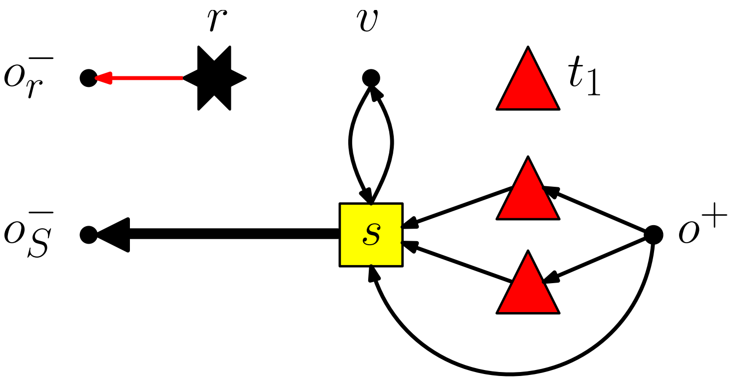
Decomposition Example I

$$P = \langle o^+, t_1, v, r, o_r^- \rangle$$

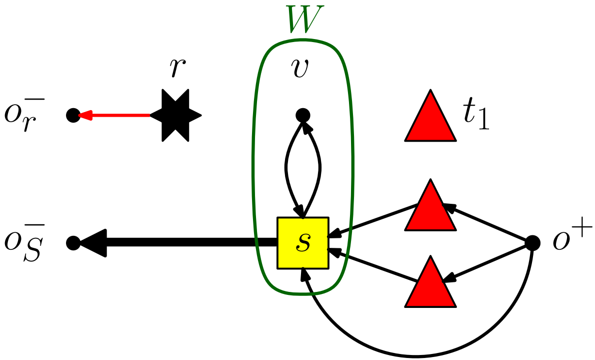


Decomposition Example I

$$P = \langle o^+, t_1, v, r, o_r^- \rangle$$



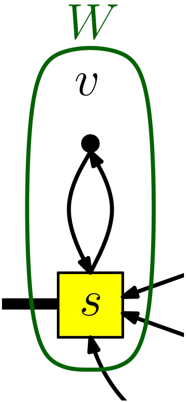
Redirecting Flow



Violation of Connectivity Inequality

$$f(\delta^+_{E^R_{\text{ext}}}(W)) \geq x_s \qquad \forall\, W \subseteq V_G, s \in W \cap S \neq \emptyset$$

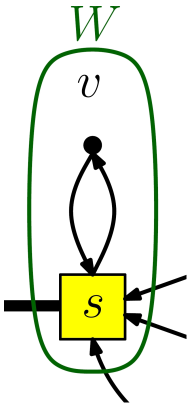
Redirecting Flow



Redirection towards o_S^- is possible!

There exists a path from v towards o_S^- in W.

Redirecting Flow



Redirection towards o_S^- is possible!

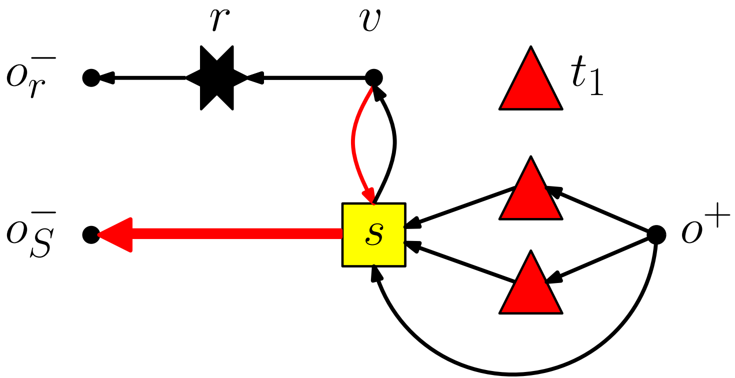
There exists a path from v towards o_S^- in W.

Reasoning

1. Flow preservation holds within W.
2. s could reach o_r^- via v before the reduction of flow.
3. v receives at least one unit of flow.
4. Flow leaving v must eventually terminate at o_S^-.

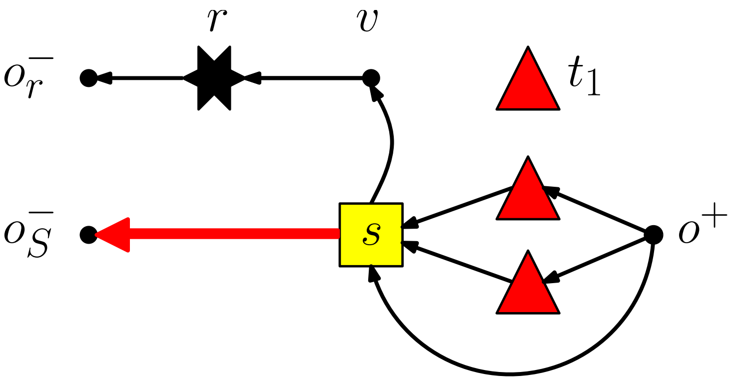
Decomposition Example II

$P = \langle o^+, t_1, v, s, o_S^- \rangle$

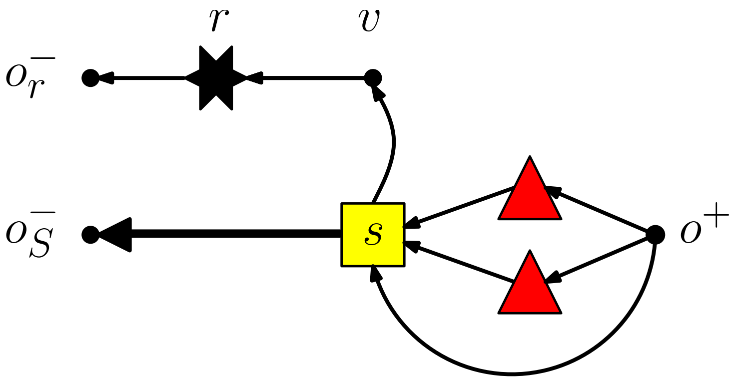


Decomposition Example II

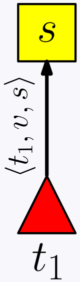
$$P = \langle o^+, t_1, v, s, o^-_S \rangle$$



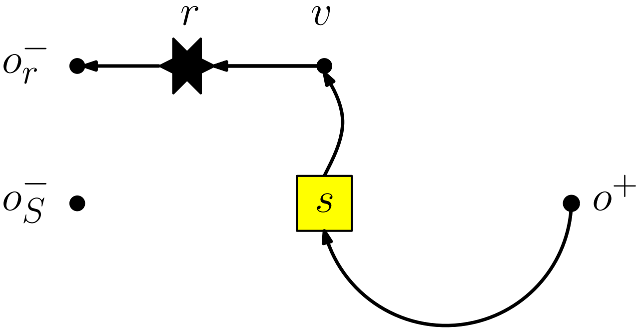
Decomposition Example II



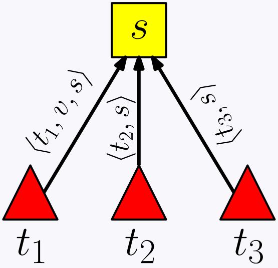
Solution



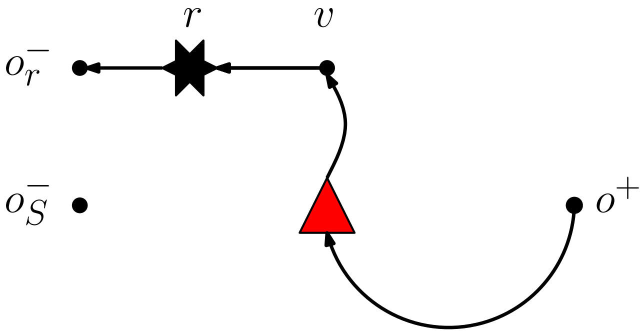
Decomposition Example II



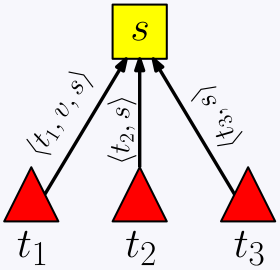
Solution



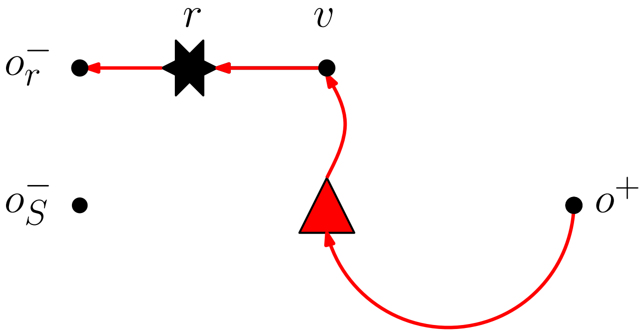
Decomposition Example II

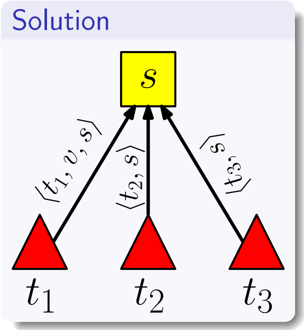


Solution



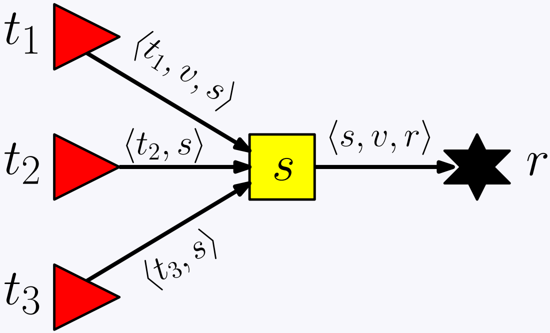
Decomposition Example II





Decomposition Example II

Final Solution



Linear Heuristics

Overview

Linear Relaxations

- The linear relaxation of an IP is obtained by relaxing the integrality constraints of the variables, thereby obtaining a Linear Program (LP).
- Solutions to linear relaxations are readily availabe when using branch-and-bound to solve an IP.
- May provide useful information to guide the construction of a solution.

Usage

- LP-based heuristics are employed within the VirtuCast *solver* to improve the bounding process.
- Yield polynomial time heuristics when used together with the root relaxation.

FlowDecoRound Heuristic

- computes a *flow* decomposition and connects nodes randomly according to the decomposition
- processing nodes are activated if another node node connects to it, must be connected itself
- failsafe: shortest paths

Algorithm 1: FlowDecoRound

Input : Network $G = (V_G, E_G, c_E, u_E)$, Request $R_G = (r, S, T, u_r, c_S, u_S)$, LP relaxation solution $(\hat{x}, \hat{f}) \in \mathcal{F}_{\mathrm{LP}}$ to **??**

Output: A Feasible Virtual Arborescence $\hat{\mathcal{T}}_G$ or `null`

set $\hat{S} \triangleq \emptyset$ **and** $\hat{T} \triangleq \emptyset$ **and** $U = T$

set $\hat{V}_{\mathcal{T}} \triangleq \{r\}$, $\hat{E}_{\mathcal{T}} \triangleq \emptyset$ **and** $\hat{\pi} : \hat{E}_{\mathcal{T}} \to \mathcal{P}_G$

set $u(e) \triangleq \begin{cases} u_E(e) & \text{, if } e \in E_G \\ u_r(r) & \text{, if } e = (r, o_r^-) \\ u_S(s) & \text{, if } e = (s, o_S^-) \in E_{\mathrm{ext}}^{S-} \\ 1 & \text{, else} \end{cases}$ **for all** $e \in E_{\mathrm{ext}}$

while $U \neq \emptyset$ **do**

- **choose** $t \in U$ uniformly at random **and set** $U \leftarrow U - t$
- **set** $\Gamma_t \triangleq$ `MinCostFlow` $\left(G_{\mathrm{ext}}, \hat{f}, \hat{f}(o^+, t), t, \{o_S^-, o_r^-\}\right)$
- **set** $\hat{f} \leftarrow \hat{f} - \sum_{(P,f) \in \Gamma_t, e \in P} f$
- **set** $\Gamma_t \leftarrow \Gamma_t \setminus \{(P, f) \in \Gamma_t | \exists e \in P. u(e) = 0\}$
- **set** $\Gamma_t \leftarrow \Gamma_t \setminus \{(P, f) \in \Gamma_t | (\hat{V}_{\mathcal{T}} + t, \hat{E}_{\mathcal{T}} + (t, P_{|P|-1}))$ is not acyclic $\}$
- **if** $\Gamma_t \neq \emptyset$ **then**
  - **choose** $(P, f) \in \Gamma_t$ with probability $f / \left(\sum_{(P_j, f_j) \in \Gamma_t} f_j\right)$
  - **if** $P_{|P|-1} \notin \hat{V}_{\mathcal{T}}$ **then**
    - **set** $U \leftarrow U + P_{|P|-1}$ **and** $\hat{V}_{\mathcal{T}} \leftarrow \hat{V}_{\mathcal{T}} + P_{|P|-1}$
  - **set** $\hat{V}_{\mathcal{T}} \leftarrow \hat{V}_{\mathcal{T}} + t$ **and** $\hat{E}_{\mathcal{T}} \leftarrow \hat{E}_{\mathcal{T}} + (t, P_{|P|-1})$ **and** $\hat{\pi}(t, P_{|P|-1}) \triangleq P$
  - **set** $u(e) \leftarrow u(e) - 1$ **for all** $e \in P$

set $u(e) \leftarrow 0$ **for all** $e = (s, o_S^-) \in E_{\mathrm{ext}}^{S-}$ **with** $s \in S \wedge s \notin \hat{V}_{\mathcal{T}}$

set $\bar{T} \triangleq (T \setminus \hat{V}_{\mathcal{T}}) \cup (\{s \in S \cap \hat{V}_{\mathcal{T}} | \delta^+_{\hat{E}_{\mathcal{T}}}(s) = 0\})$

for $t \in \bar{T}$ **do**

- **choose** $P \leftarrow$ `ShortestPath` $(G^u_{\mathrm{ext}}, c_E, t, \{o_S^-, o_r^-\})$ **such that** $(\hat{V}_{\mathcal{T}} + t, \hat{E}_{\mathcal{T}} + (t, P_{|P|-1}))$ is acyclic
- **if** $P = \emptyset$ **then**
  - **return** `null`
- **set** $\hat{V}_{\mathcal{T}} \leftarrow \hat{V}_{\mathcal{T}} + t$ **and** $\hat{E}_{\mathcal{T}} \leftarrow \hat{E}_{\mathcal{T}} + (t, P_{|P|-1})$ **and** $\hat{\pi}(t, P_{|P|-1}) \triangleq P$
- **set** $u(e) \leftarrow u(e) - 1$ **for all** $e \in P$

for $e \in \hat{E}_{\mathcal{T}}$ **do**

- **set** $P \triangleq \hat{\pi}(e)$
- **set** $\hat{\pi}(e) \leftarrow \langle P_1, \ldots, P_{|P|-1} \rangle$

set $\hat{\mathcal{T}}_G \triangleq$ Virtual Arborescence $(\hat{V}_{\mathcal{T}}, \hat{E}_{\mathcal{T}}, r, \hat{\pi})$

return `PruneSteinerNodes` $(\hat{\mathcal{T}}_G)$

Intermezzo: VCPrimConnect

Important Observation

If all placed processing nodes are already connected, all senders can be assigned *optimally* using a minimum cost flow.

Outline

1. connect all opened processing nodes in tree via a adaption of Prim's MST algorithm
2. assign all sending nodes using min-cost flow

Algorithm 2: VCPrimConnect

Input : Network $G = (V_G, E_G, c_E, u_E)$, Request $R_G = (r, S, T, u_r, c_S, u_S)$, *Partial* Virtual Arborescence $\mathcal{T}_G^P = (V_{\mathcal{T}}^P, E_{\mathcal{T}}^P, r, \pi^P)$

Output: Feasible Virtual Arborescence $\mathcal{T}_G = (V_{\mathcal{T}}, E_{\mathcal{T}}, r, \pi)$ or `null`

set $U \triangleq \{v | v \in V_{\mathcal{T}}^P \setminus \{r\}, \delta^+_{E_{\mathcal{T}}^P}(v) = 0\}$
set $\bar{S} \triangleq U \cap S$
set $V_{\mathcal{T}} \triangleq V_{\mathcal{T}}^P$, $E_{\mathcal{T}} \triangleq E_{\mathcal{T}}^P$ **and** $\pi(u,v) = \pi^P(u,v)$ **for all** $(u,v) \in E_{\mathcal{T}}$
set $u(e) \triangleq u_E(e) - |\pi(E_{\mathcal{T}})[e]|$ **for all** $e \in E_G$
while $\bar{S} \neq \emptyset$ **do**
    compute $R \leftarrow \{r' | r \in \{r\} \cup (V_{\mathcal{T}} \cap S), r'$ reaches r in $\mathcal{T}_G, \delta^-_{E_{\mathcal{T}}}(r') < u_{r,S}(r')\}$
    compute $P = \mathtt{MinAllShortestPath}(G^u, c_E, \bar{S}, R)$
    if $P = \mathtt{null}$ **then**
        return $\mathtt{null}$
    end
    set $\bar{S} \leftarrow \bar{S} - P_1$
    set $E_{\mathcal{T}} \leftarrow E_{\mathcal{T}} + (P_1, P_{|P|})$ **and** $\pi(P_1, P_{|P|}) \triangleq P$
    set $u(e) \leftarrow u(e) - 1$ **for all** $e \in P$
end
set $\bar{T} \triangleq U \cap T$
set $u_V(r') \triangleq u_{r,S}(r') - \delta^-_{E_{\mathcal{T}}}(r')$ **for all** $r' \in \{r\} \cup (V_{\mathcal{T}} \cap S)$
compute $\Gamma = \{P^{\bar{t}}\} \leftarrow \mathtt{MinCostAssignment}(G, c_E, u, u_V, \bar{T}, \{r\} \cup V_{\mathcal{T}} \cap S)$
if $\Gamma = \emptyset$ **then**
    return $\mathtt{null}$
end
set $E_{\mathcal{T}} \leftarrow E_{\mathcal{T}} + (t, P^t_{|P^t|})$ **and** $\pi(t, P^t_{|P^t|}) \triangleq P^t$ **for all** $P^t \in \Gamma$
return $\mathcal{T}_G \triangleq (V_{\mathcal{T}}, E_{\mathcal{T}}, r, \pi)$

MultipleShots

- treats node variables as probabilities and iteratively places processing functionality accordingly
- tries to generate a feasible solution in each round via VCPrimConnect

Algorithm 3: MultipleShots

Input : Network $G = (V_G, E_G, c_E, u_E)$, Request $R_G = (r, S, T, u_r, c_S, u_S)$, LP relaxation solution $(\hat{x}, \hat{f}) \in \mathcal{F}_{\text{LP}}$ to **??**

Output: A Feasible Virtual Arborescence $\hat{\mathcal{T}}_G$ or `null`

set $\lfloor S \rfloor \triangleq \{s \in S | \hat{x}_s \leq 0.01\}$ **and** $\lceil S \rceil \triangleq \{s \in S | \hat{x}_s \geq 0.99\}$
`addConstraintsLocally`$(\{x_s = 0 | s \in \lfloor S \rfloor\} \cup \{x_s = 1 | s \in \lceil S \rceil\})$
set $\dot{S}_0 \triangleq \lfloor S \rfloor \cup$ **and** $\dot{S}_1 \triangleq \lceil S \rceil$
`disableGlobalPrimalBound()`
repeat
    $(\hat{x}, \hat{f}) \leftarrow$ `solveSeparateSolve()`
    if `infeasibleLP()` **return** *null*
    set $\lfloor S \rfloor \triangleq \{s \in S | \hat{x}_s \leq 0.01\}$ **and** $\lceil S \rceil \triangleq \{s \in S | \hat{x}_s \geq 0.99\}$
    `addConstraintsLocally`$(\{x_s = 0 | s \in \lfloor S \rfloor\} \cup \{x_s = 1 | s \in \lceil S \rceil\})$
    set $\dot{S}_0 \leftarrow \dot{S}_0 \cup \lfloor S \rfloor$ **and** $\dot{S}_1 \leftarrow \dot{S}_1 \cup \lceil S \rceil$
    set $\hat{S} \triangleq S \setminus (\dot{S}_0 \cup \dot{S}_1)$
    if $\hat{S} \neq \emptyset$ **then**
        repeat
            set $S_1 \triangleq \hat{S}$
            remove s **from** S_1 **with probability** $1 - \hat{x}_s$ **for all** $s \in S_1$
            if $S_1 = \emptyset$ ***and*** $|S \setminus (\dot{S}_0 \cup \dot{S}_1)| < 10$ **then**
                set $S_1 \leftarrow S \setminus (\dot{S}_0 \cup \dot{S}_1)$
        until $S_1 \neq \emptyset$
        `addConstraintsLocally`$(\{x_s = 1 | s \in S_1\})$
        set $\dot{S}_1 \leftarrow \dot{S}_1 \cup S_1$
    $\hat{\mathcal{T}}^P_G \triangleq (\hat{V}^P_T, \hat{E}^P_T, r, \emptyset)$ **where** $\hat{V}^P_T \triangleq \{r\} \cup T \cup \dot{S}_1$ **and** $\hat{E}_T \triangleq \emptyset$
    set $\hat{\mathcal{T}}_G \triangleq$`VCPrimConnect`$(G, R_G, \hat{\mathcal{T}}^P_G)$
    if $\hat{\mathcal{T}}_G \neq$ *null* **then**
        return `PruneSteinerNodes`$(\hat{\mathcal{T}}_G)$
until $\dot{S}_0 \cup \dot{S}_1 = S$
return *null*

GreedyDiving

- aims at generating a feasible *IP* solution
- iteratively bounds at least a single variable from below, first fixing node variables
- complex failsafe: PartialDecompose + VCPrimConnect

Algorithm 4: GreedyDiving

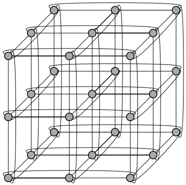
Input : Network $G = (V_G, E_G, c_E, u_E)$, Request $R_G = (r, S, T, u_r, c_S, u_S)$, LP relaxation solution $(\hat{x}, \hat{f}) \in \mathcal{F}_{\text{LP}}$ to **??**

Output: A Feasible Virtual Arborescence $\hat{\mathcal{T}}_G$ or `null`

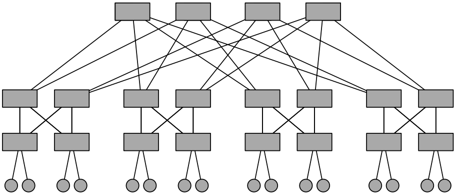
set $\lfloor S \rfloor \triangleq \{s \in S | \hat{x}_s \leq 0.01\}$ **and** $\lceil S \rceil \triangleq \{s \in S | \hat{x}_s \geq 0.99\}$
`addConstraintsLocally`$(\{x_s = 0 | s \in \lfloor S \rfloor\} \cup \{x_s = 1 | s \in \lceil S \rceil\})$
set $\dot{S} \triangleq \lfloor S \rfloor \cup \lceil S \rceil$ **and** $\dot{E} \triangleq \emptyset$
do
    $(\hat{x}', \hat{f}') \leftarrow$ `solveSeparateSolve()`
    if *infeasibleLP()* ***and*** $\dot{S} = S$ **then**
        break
    else if *infeasibleLP()* ***or*** *objectiveLimit()* **then**
        return *null*
    set $(\hat{x}, \hat{f}) \leftarrow (\hat{x}', \hat{f}')$
    if $\dot{S} \neq S$ **then**
        set $\lfloor S \rfloor \triangleq \{s \in S | \hat{x}_s \leq 0.01\}$ **and** $\lceil S \rceil \triangleq \{s \in S | \hat{x}_s \geq 0.99\}$
        `addConstraintsLocally`$(\{x_s = 0 | s \in \lfloor S \rfloor\} \cup \{x_s = 1 | s \in \lceil S \rceil\})$
        set $\dot{S} \leftarrow \dot{S} \cup \lfloor S \rfloor \cup \lceil S \rceil$
        set $\hat{S} \triangleq S \setminus \dot{S}$
        if $\hat{S} \neq \emptyset$ **then**
            choose $\hat{s} \in \hat{S}$ **with** $c_S(\hat{s})/\hat{x}_{\hat{s}}$ **minimal**
            `addConstraintsLocally`$(\{x_{\hat{s}} = 1\})$
            set $\dot{S} \leftarrow \dot{S} + \hat{s}$
    else if $\dot{E} \neq E_{\text{ext}}$ **then**
        set $\lfloor E \rfloor \triangleq \{e \in E_{\text{ext}} | \, |\hat{f}_e - \lfloor \hat{f}_e \rfloor| \leq 0.001\}$, $\lceil E \rceil \triangleq \{e \in E_{\text{ext}} | \, |\hat{f}_e - \lceil \hat{f}_e \rceil| \leq 0.001\}$
        `addConstraintsLocally`$(\{f_e = \lfloor \hat{f}_e \rfloor | e \in \lfloor E \rfloor\} \cup \{f_e = \lceil \hat{f}_e \rceil | e \in \lceil E \rceil\}$
        set $\dot{E} \leftarrow \dot{E} \cup \lfloor E \rfloor \cup \lceil E \rceil$
        set $\hat{E} \triangleq E_{\text{ext}} \setminus \dot{E}$
        if $\hat{E} \neq \emptyset$ **then**
            choose $\hat{e} \in \hat{E}$ **with** $\lceil \hat{f}_{\hat{e}} \rceil - \hat{f}_{\hat{e}}$ **minimal**
            `addConstraintsLocally`$(\{\hat{f}_{\hat{e}} \geq \lceil \hat{f}_{\hat{e}} \rceil\})$
            set $\dot{E} \leftarrow \dot{E} + \hat{e}$
    else
        break
set $\hat{f}_e \leftarrow \lfloor \hat{f}_e \rfloor$ **for all** $e \in E_{\text{ext}} \setminus \dot{E}$
set $\hat{\mathcal{T}}_G^P \leftarrow$ `PartialDecompose` $(G, R_G, (\hat{x}, \hat{f}))$
return `VCPrimConnect`$(G, R_G, \hat{\mathcal{T}}_G^P)$

Computational Evaluation

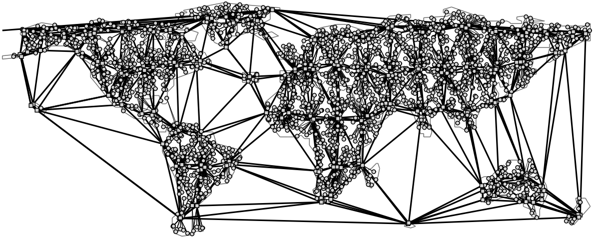
Topologies



3D torus



Fat tree



An ISP topology generated by IGen with 2400 nodes.

Instances

Generation Parameters

- five graph sizes I-V
- 15 instances per graph size: different Steiner costs, different edge capacities

	Nodes	Edges	Processing Locations	Senders
Fat tree	1584	14680	720	864
3D torus	1728	10368	432	864
IGen	4000	16924	401	800

Table: Largest graph sizes

Computational Setup

Implementation

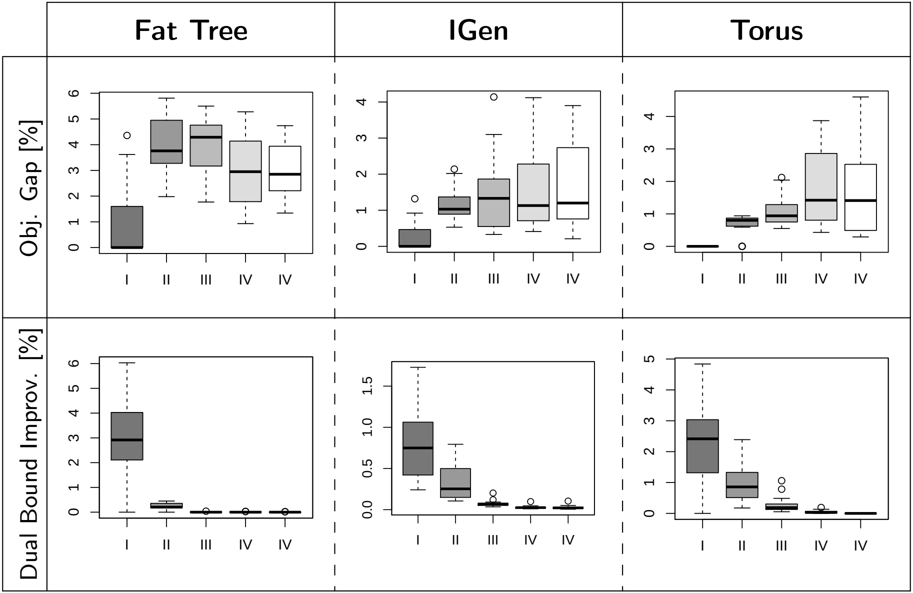
- *all* algorithms (except MCF-IP) are implemented in C/C++
- VirtuCast uses SCIP [1], many different parameters to consider
  - separation
  - branching
  - heuristics
  - separation procedure: nested cuts, creep flow, cyclic generation...
- MCF-IP is implemented using GMPL + CPLEX

Objective

Solve instances within reasonable time: 1 hour runtime limit

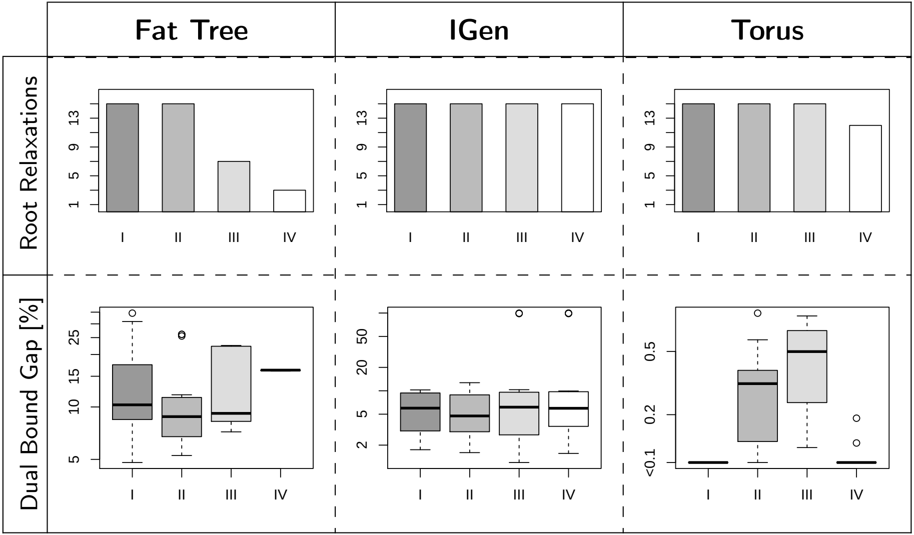
VirtuCast + LP-based Heuristics

VirtuCast + LP-based Heuristics



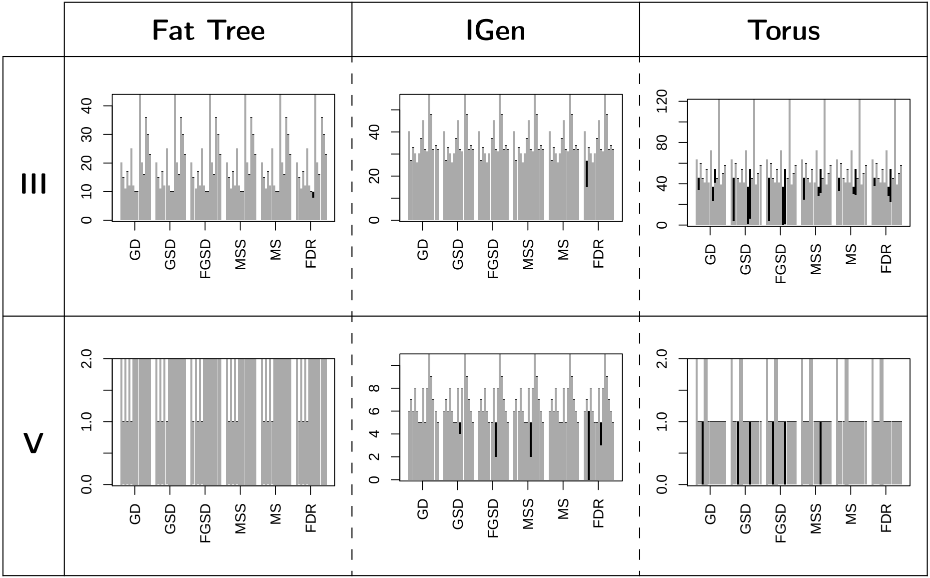
MCF-IP

MCF-IP: Performance

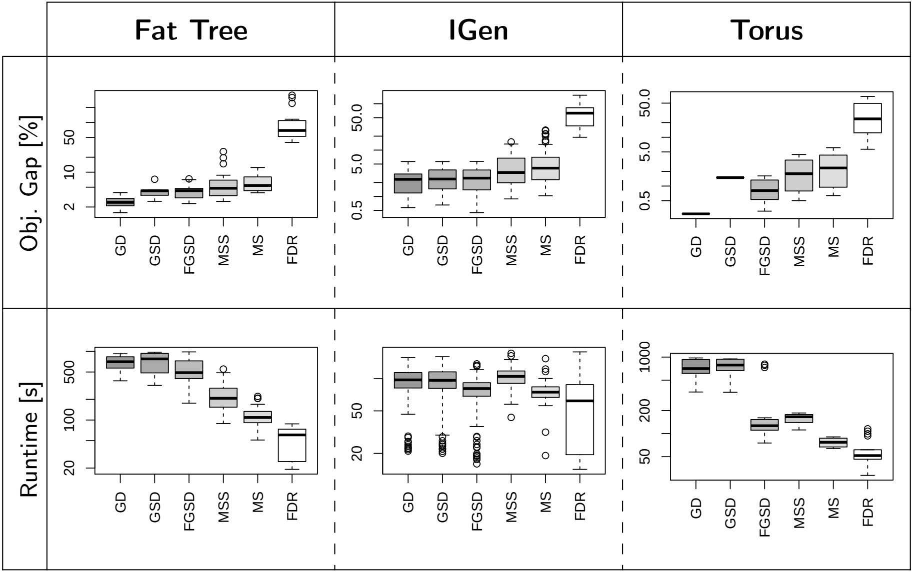


LP-based Heuristics

LP-based Heuristics: Efficacy



LP-based Heuristics: Performance on graph size V



Conclusion

Publications

Matthias Rost, Stefan Schmid: OPODIS 2013 & arXiv [15, 14]
Matthias Rost (Adv. Stefan Schmid): M.Sc. Thesis [13]

Concise definition of CVSAP

Inapproximability

Approximations

- NVSTP
- VSTP
- VSAP

Exact Algorithms

- multi-commodity flow
- single-commodity flow
  $\rightarrow$ VirtuCast

Heuristics

- FlowDecoRound
- MultipleShots
- GreedyDiving

Extensive explorative Computational Evaluation

Related Work

Molnar: Constrained Spanning Tree Problems [9]

- Shows that optimal solution is a 'spanning hierarchy' and not a DAG.

Oliveira et. al: Flow Streaming Cache Placement Problem [11]

- Consider a weaker variant of multicasting CVSAP without bandwidth
- Use a (faulty) MIP to define the problem
- Give weak approximation algorithm

Shi: Scalability in Overlay Multicasting [16]

- Provided heuristic and showed improvement in scalability.

Future Work

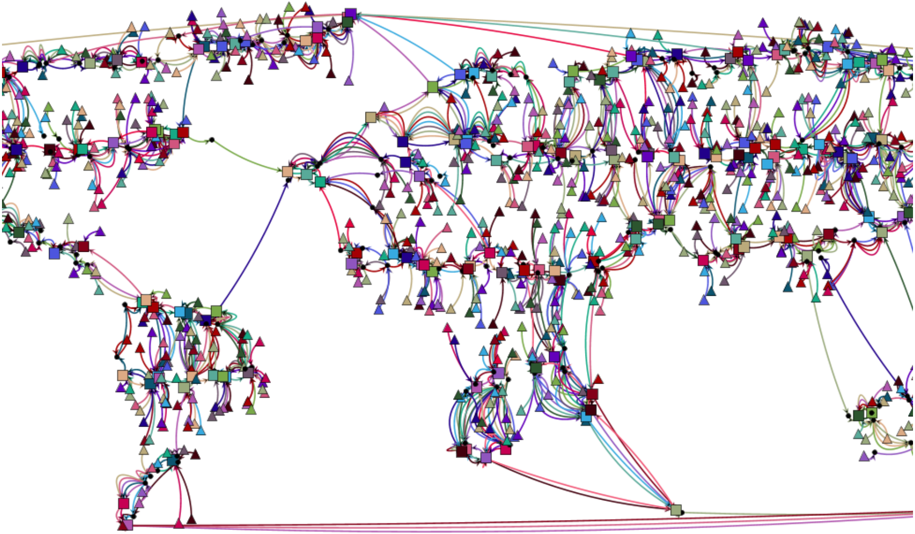
Model Extensions

- prize-collecting variants
- concurrent multicast / aggregation sessions
- 'extend' MIP formulation for weaker variants

Speeding-up Separation / Public Service Announcement

- Koch et al. [7] stated that using Hao-Orlin the computation could be sped up.
- Cronholm et al. show that this is not really the case, but derive an adaptation [4]:
  For single node, all separations can be computed in $\mathcal{O}(nm \log(n^2/m))$

Thanks



References I

[1] T. Achterberg.
SCIP: solving constraint integer programs.
Mathematical Programming Computation, 1(1):1–41, 2009.

[2] P. Costa, A. Donnelly, A. Rowstron, and G. O. Shea.
Camdoop: Exploiting In-network Aggregation for Big Data Applications.
In *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2012.

[3] C. Cranor, T. Johnson, O. Spataschek, and V. Shkapenyuk.
Gigascope: A Stream Database for Network Applications.
In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 647–651, 2003.

[4] W. Cronholm, F. Ajili, and S. Panagiotidi.
On the minimal steiner tree subproblem and its application in branch-and-price.
In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 125–139. Springer, 2005.

[5] M. Ding, X. Cheng, and G. Xue.
Aggregation tree construction in sensor networks.
In *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, volume 4, pages 2168–2172. IEEE, 2003.

References II

[6] C. Hermsmeyer, E. Hernandez-Valencia, D. Stoll, and O. Tamm.
Ethernet aggregation and core network models for effcient and reliable iptv services.
Bell Labs Technical Journal, 12(1):57–76, 2007.

[7] T. Koch and A. Martin.
Solving steiner tree problems in graphs to optimality.
Networks, 32(3):207–232, 1998.

[8] B. Krishnamachari, D. Estrin, and S. Wicker.
Modelling data-centric routing in wireless sensor networks.
In *IEEE infocom*, volume 2, pages 39–44, 2002.

[9] M. Molnár.
Hierarchies to Solve Constrained Connected Spanning Problems.
Technical Report lirmm-00619806, University Montpellier 2, LIRMM, 2011.

[10] S. Narayana, W. Jiang, J. Rexford, and M. Chiang.
Joint Server Selection and Routing for Geo-Replicated Services.
In *Proc. Workshop on Distributed Cloud Computing (DCC)*, 2013.

References III

[11] C. Oliveira and P. Pardalos.
Streaming cache placement.
In *Mathematical Aspects of Network Routing Optimization*, Springer Optimization and Its Applications, pages 117–133. Springer New York, 2011.

[12] R. Ravi, M. V. Marathe, S. Ravi, D. J. Rosenkrantz, and H. B. Hunt III.
Approximation algorithms for degree-constrained minimum-cost network-design problems.
Algorithmica, 31(1):58–78, 2001.

[13] M. Rost.
Optimal Virtualized In-Network Processing with Applications to Aggregation and Multicast, 2014.

[14] M. Rost and S. Schmid.
The Constrained Virtual Steiner Arborescence Problem: Formal Definition, Single-Commodity Integer Programming Formulation and Computational Evaluation.
Technical report, arXiv, 2013.

[15] M. Rost and S. Schmid.
Virtucast: Multicast and aggregation with in-network processing.
In R. Baldoni, N. Nisse, and M. Steen, editors, *Principles of Distributed Systems*, volume 8304 of *Lecture Notes in Computer Science*, pages 221–235. Springer International Publishing, 2013.

References IV

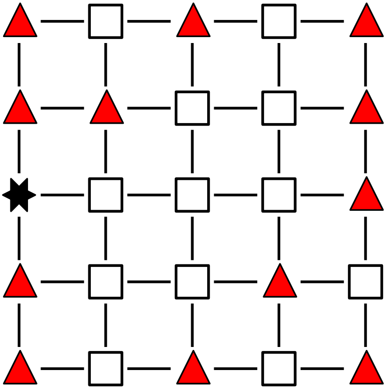
[16] S. Shi.
A proposal for a scalable internet multicast architecture.
In *Washington Universtiy*, 2001.

Approximation of NVSTP via DNSTP

Approximation of NVSTP via DNSTP

NVSTP

- undirected version
- no edge capacities
- Steiner nodes have capacities
- connect terminals using Steiner nodes to root



Approximation of NVSTP via DNSTP: [12]

Definition (Degree-Constrained Node Weighted Steiner Tree Problem [12])

Given: Undirected network $G = (V_G, E_G, c_E, c_V, u_V)$ with edge costs $c_E : E_G \to \mathbb{R}_{\geq 0}$[a], node costs $c_V : V_G \to \mathbb{R}_{\geq 0}$, and a degree bound function $u_V : V_G \to \mathbb{N}_{\geq 2}$ and set of terminals $T \subset V_G$.

Task: Find a Steiner tree $\mathcal{T} \subseteq E_G$ connecting all terminals T, such that for each node v that is contained in $\mathcal{T}$ the degree bound is not violated, i.e. that $\delta_{\mathcal{T}}(v) \leq u_V(v)$ holds, minimizing the cost $C_{\mathrm{DNSTP}}(\mathcal{T}) = \sum_{e \in \mathcal{T}} c_E(e) + \sum_{v \in \mathcal{T}} c_V(v)$.

[a] The original definition and the corresponding theorem only considers the node weighted case.

Approximation of NVSTP via DNSTP

Theorem (Logarithmic bi-criteria approximation for DNSTP [12])

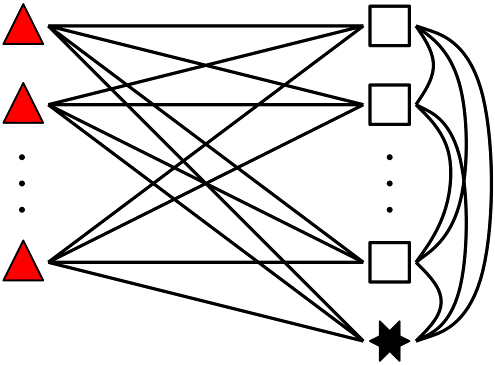
There exists a polynomial-time algorithm that returns a solution where node capacities are (individually) violated at most by a factor $\mathcal{O}(\log |T|)$ and of cost within a factor of $\mathcal{O}(\log |T|)$ the optimum solution.

Differences of NVSTP w.r.t. DNSTP

- NVSTP constructs a tree, i.e. terminals have degree $\mathbf{1}$.
- NVSTP may use arbitrary paths to connect nodes.
- Not all nodes may be used as Steiner nodes.

NVSTP via DNSTP: Construction

- bipartite mesh connecting any terminal to any Steiner node
- clique between all Steiner nodes and the root
- all edges have cost of respective shortest path

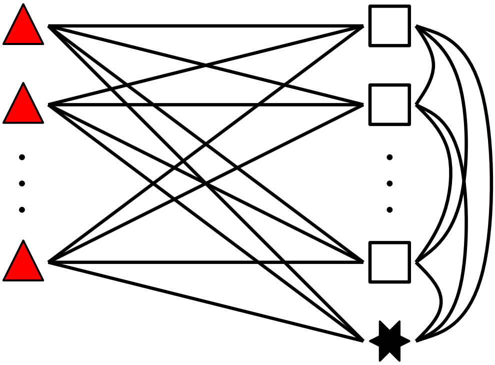


Checklist

- NVSTP constructs a tree, i.e. terminals have degree 1.
- ~~NVSTP may use arbitrary paths to connect nodes.~~
- ~~Not all nodes may be used as Steiner nodes.~~

NVSTP via DNSTP: Construction

- bipartite mesh connecting any terminal to any Steiner node
- clique between all Steiner nodes and the root
- all edges have cost of respective shortest path



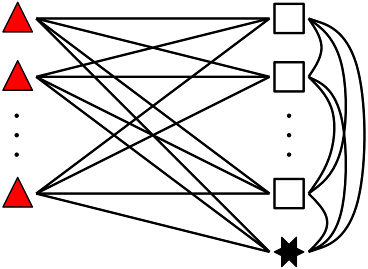
Checklist

- NVSTP constructs a tree, i.e. terminals have degree **1**.
- ~~NVSTP may use arbitrary paths to connect nodes~~.
- ~~Not all nodes may be used as Steiner nodes~~.

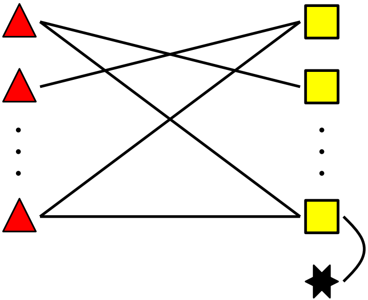
Outline of Bicriteria Approximation for NVSTP

Algorithm

1. construct graph as described above



Outline of Bicriteria Approximation for NVSTP



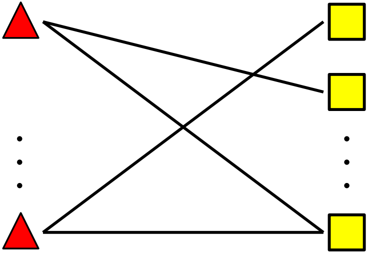
Algorithm

1. construct graph as described above
2. use Approximation by Ravi et al. to obtain DNSTP solution

Outline of Bicriteria Approximation for NVSTP

Algorithm

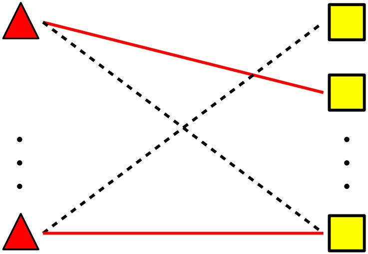
1. construct graph as described above
2. use Approximation by Ravi et al. to obtain DNSTP solution
3. consider bipartite subgraph of terminals with degree > 1



Outline of Bicriteria Approximation for NVSTP

Algorithm

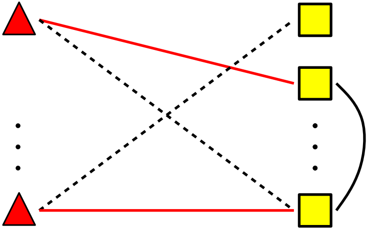
1. construct graph as described above
2. use Approximation by Ravi et al. to obtain DNSTP solution
3. consider bipartite subgraph of terminals with degree > 1
4. compute maximum matching with size $=$ number of terminals



Outline of Bicriteria Approximation for NVSTP

Algorithm

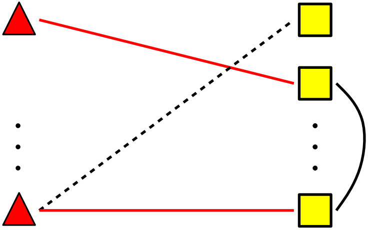
1. construct graph as described above
2. use Approximation by Ravi et al. to obtain DNSTP solution
3. consider bipartite subgraph of terminals with degree > 1
4. compute maximum matching with size $=$ number of terminals
5. perform 'leafify' operation on terminals



Outline of Bicriteria Approximation for NVSTP

Algorithm

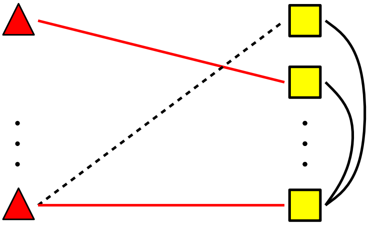
1. construct graph as described above
2. use Approximation by Ravi et al. to obtain DNSTP solution
3. consider bipartite subgraph of terminals with degree > 1
4. compute maximum matching with size $=$ number of terminals
5. perform 'leafify' operation on terminals



Outline of Bicriteria Approximation for NVSTP

Algorithm

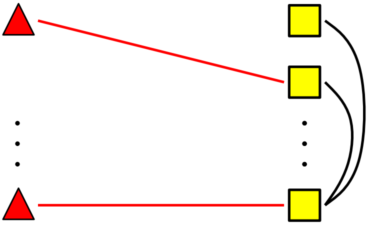
1. construct graph as described above
2. use Approximation by Ravi et al. to obtain DNSTP solution
3. consider bipartite subgraph of terminals with degree > 1
4. compute maximum matching with size $=$ number of terminals
5. perform 'leafify' operation on terminals



Outline of Bicriteria Approximation for NVSTP

Algorithm

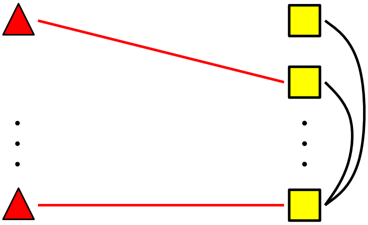
1. construct graph as described above
2. use Approximation by Ravi et al. to obtain DNSTP solution
3. consider bipartite subgraph of terminals with degree > 1
4. compute maximum matching with size $=$ number of terminals
5. perform 'leafify' operation on terminals



Outline of Bicriteria Approximation for NVSTP

Algorithm

1. construct graph as described above
2. use Approximation by Ravi et al. to obtain DNSTP solution
3. consider bipartite subgraph of terminals with degree > 1
4. compute maximum matching with size $=$ number of terminals
5. perform 'leafify' operation on terminals



Theorem

1. Cost of introduced edges is bounded by triangle equation
2. Degree of non-terminals in matching is increased by 1
3. $\mathcal{O}(\log|T|, \log|T|)$ for DNSTP $\Rightarrow$ $\mathcal{O}(\log|T|, \log|T|)$ for NVSTP