‘Formal Virtu’ Project Overview

Temporal VNet Embedding and Virtualized In-Network Processing

VINO Meeting February 2014

Matthias Rost

Technische Universität Berlin

February 5th, 2014
TU Berlin

Overview

Virtual Network Embedding MIP Creator (VNetEMC)

- Unified framework for VNet embedding MIPs

Virtualized In-Network Processing

- Service deployment rather than VNet embedding
- Applicable to multicast and aggregation communication

Temporal VNet Embedding

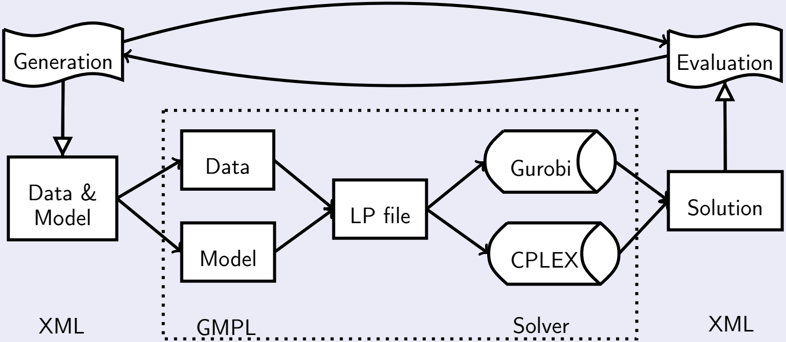
- Scheduling of VNets, given temporal flexibilities & embedding
- Applicable to other 'embeddings' as well

Virtual Network Embedding MIP Creator

Idea: Unified IO for VNet Embedding Experiments

Idea

- Framework for generation & evaluation of VNet embedding MIPs
- Unified XML input and output
- Persistent storage of models and solutions



Supported Models

VNet

- directed / undirected
- single resource for links and nodes

Substrate

- directed / undirected
- capacitated / uncapacitated links and nodes

Scenario = Objective +

Mapping

- access control
- disjoint node mappings
- node placement restrictions

+

Flow Model

- splittable
- unsplittable
- confluent

Virtualized In-Network Processing

VNet Embedding vs. Service Deployment

VNet Embedding

Customer specifies VNet *fully*, i.e.

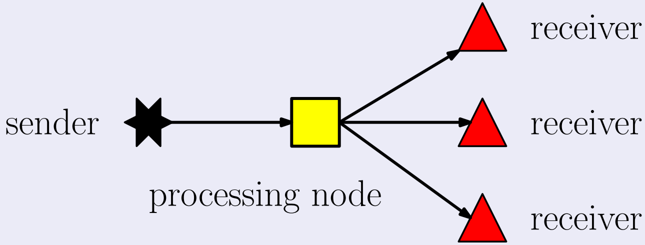
- topology, resource requirements, locations, . . .

Service Deployment

- Customer requests communication *service* between locations, **without** specifying a topology for establishing the service.
- Considered communication services: multicast & aggregation

Communication Schemes: Multicast

processing = duplication + reroute



Communication Schemes: Multicast

processing = duplication + reroute

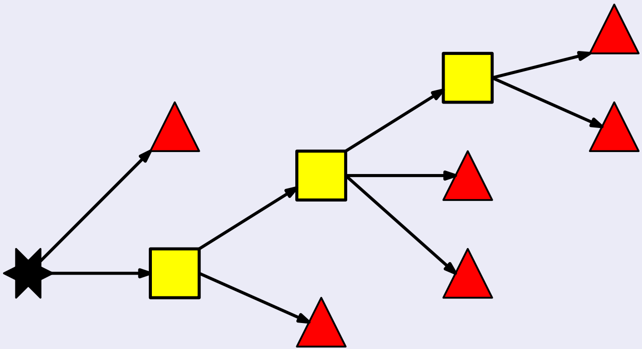
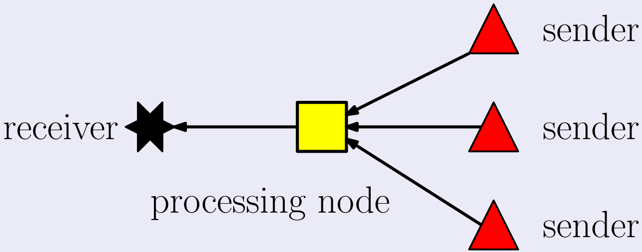


Figure: Hierarchy of processing nodes

Communication Schemes: Aggregation

processing = merge + reroute



Communication Schemes: Aggregation

processing = merge + reroute

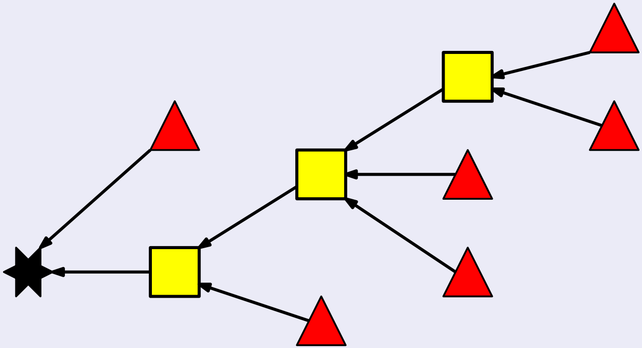


Figure: Hierarchy of processing nodes

Problem Statement

Virtualization on the rise: SDN + NFV

- How to compute virtual aggregation / multicasting trees?
- Where to place in-network processing functionality?

Our Answer

- *New Model*: Constrained Virtual Steiner Arborescence Problem
- *New Algorithm*: VirtuCast

Objective: Jointly minimize . . .

- bandwidth
- number of processing nodes

Introductory Example

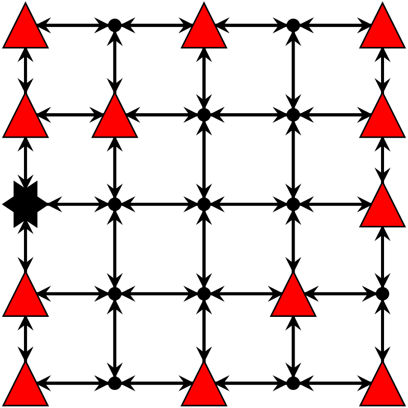
Aggregation scenario

grid graph with 14 senders and one receiver

Virtualized links

Flow can be routed along arbitrary paths

receiver sender



Without in-network processing: Unicast

Solution Method

- minimal cost flow

Solution uses

- 41 edges
- 0 processing nodes

receiver sender

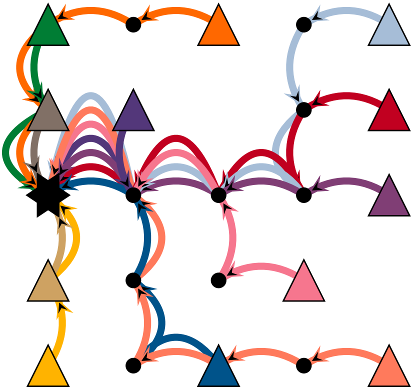


Figure: Unicast solution

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

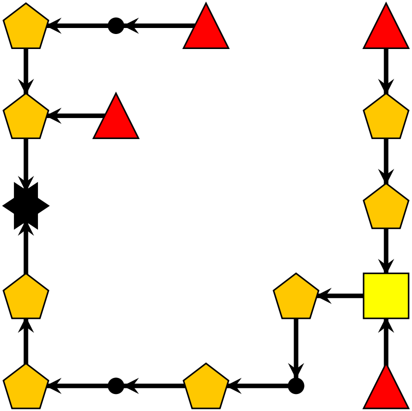
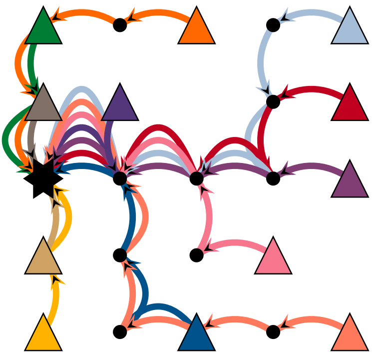
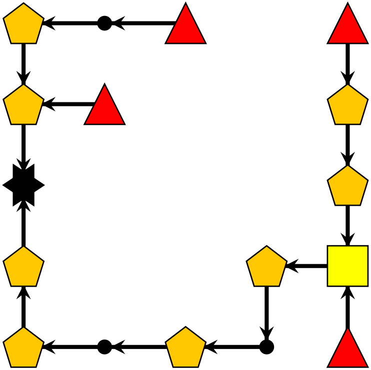


Figure: Aggregation tree

How to Trade-off?



vs.



Our Solution: CVSAP & VirtuCast

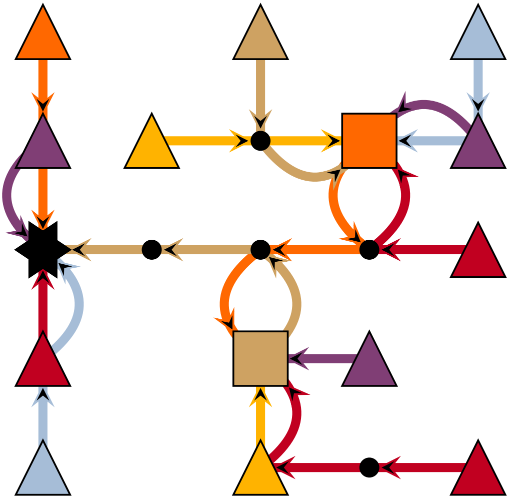
Solution uses

- 26 edges
- 2 processing nodes

receiver

sender

processing node



Solution Structure

New Model

Constrained Virtual Steiner Arboresence Problem (CVSAP)

Virtual Arborescence

- directed tree towards receiver
- sender are leaves
- inner nodes represent processing nodes
- edges represent paths in underlying network

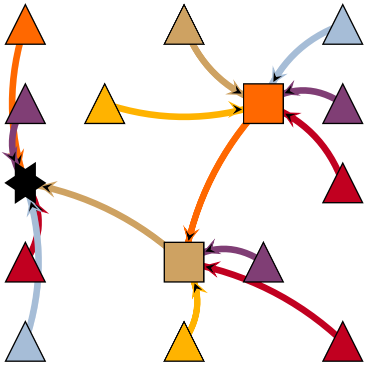


Figure: Virtual Arborescence

Definition of the
Constrained Virtual Steiner Arborescence Problem

Multicast $\triangleq$ Aggregation

Multicasting scenario can be reduced onto the aggregation scenario

We only consider the aggregation scenario.

Input to the Constained Virtual Steiner Arborescence Problem

Graph

- Directed Graph $G = (V_G, E_G)$
- Root $r \in V_G$, i.e. single receiver
- Terminals $T \subset V_G$, i.e. sender
- Steiner sites $S \subset V_G$, i.e. potential processing locations

Input to the Constained Virtual Steiner Arborescence Problem

Graph

- Directed Graph $G = (V_G, E_G)$
- Root $r \in V_G$, i.e. single receiver
- Terminals $T \subset V_G$, i.e. sender
- Steiner sites $S \subset V_G$, i.e. potential processing locations

Important

No processing functionality can be placed on non-Steiner nodes.

Input to the Constained Virtual Steiner Arborescence Problem

Graph

- Directed Graph $G = (V_G, E_G)$
- Root $r \in V_G$, i.e. single receiver
- Terminals $T \subset V_G$, i.e. sender
- Steiner sites $S \subset V_G$, i.e. potential processing locations

Important

No processing functionality can be placed on non-Steiner nodes.

Costs

- for edges
- for opening Steiner sites

Capacities

- for edges
- for Steiner sites & the root

Constrained Virtual Steiner Arborescence Problem

Definition

Find a Virtual Arborescence such that

- terminals (and only terminals) are leaves,
- non-Steiner sites are not contained $\Leftrightarrow$ all inner nodes are *activated* Steiner nodes, i.e. processing nodes, and
- node and edge capacities hold,

minimizing

sum of edge costs + sum of installation costs

Applications

Applications

	Network	Application	Technology, e.g.
multicast	ISP	service replication / cache placement [10, 11]	middleboxes / NFV + SDN
	backbone	optical multicast [6]	ROADM[1] + SDH
	all	application-level multicast [16]	different
aggregation	sensor network	value & message aggregation [4, 7]	source routing
	ISP	network analytics: Gigascope [3]	middleboxes / NFV + SDN
	data center	big data / map-reduce: Camdoop [2]	SDN

[1] reconfigurable optical add/drop multiplexer

Contributions

Contributions

- Computational Complexity
  - CVSAP is inapproximable, unless $P = NP$
  - for weaker variants, approximation algorithms exist
- Algorithms
  - VirtuCast: single-commodity flow IP with novel decomposition scheme
  - VirtuCast based heuristics
  - Multi-commodity flow IP
  - combinatorial heuristic
- Computational Evaluation
  - three topologies with five graph sizes (225 instances overall)
  - objective gap after one hour: worst case: 6%; average: 2%

Publications

- Master Thesis [12]
- Joint work with Stefan Schmid: OPODIS 2013 & arXiv [14, 13]

Computational Complexity

Computational Complexity I

Inapproximability of CVSAP

Finding a feasible solution is NP-complete.

Variants

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

Computational Complexity II

Approximation via related problems

Directed **Undirected**

$$\begin{tikzcd}
 & \mathrm{CVSAP} & \mathrm{CVSTP} \arrow[l] & \\
 & \mathrm{NVSAP} \arrow[u] & \mathrm{NVSTP} \arrow[l] \arrow[u] \arrow[r, dashed, "\mathcal{O}(\log|T|, \log|T|)"] & \mathrm{DNSTP} \\
\mathrm{SAP} \arrow[r, leftrightarrow, "\mathcal{O}(\log|T|)"] & \mathrm{VSAP} \arrow[u] & \mathrm{VSTP} \arrow[l] \arrow[u] \arrow[r, dashed, "\mathcal{O}(8)"] & \mathrm{CFLP}
\end{tikzcd}$$

Algorithms for CVSAP

Overview

VirtuCast: single-commodity flow IP formulation

- solves CVSAP to optimality in non-polynomial runtime
- allows trading-off runtime with solution quality
- baseline for heuristics

VirtuCast based heuristics

- yield high-quality solutions in polynomial time
- high efficiency in finding solutions

Multi-commodity flow & combinatorial heuristic

- generally way worse, not applicable 'out of the box'

Single- vs. Multi-Commodity Flows

Single-commodity flow formulation

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

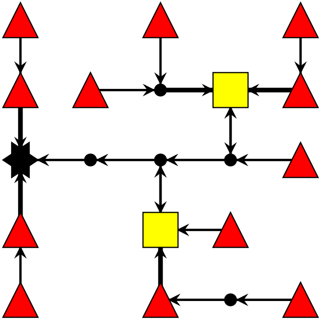


Figure: Single-commodity

Single- vs. Multi-Commodity Flows

Example: 6000 edges and 200 Steiner sites

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

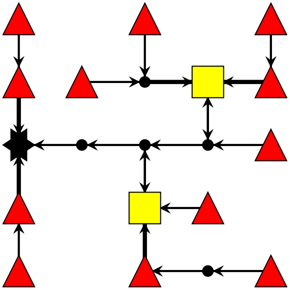


Figure: Single-commodity

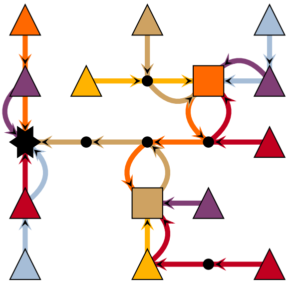


Figure: Multi-commodity

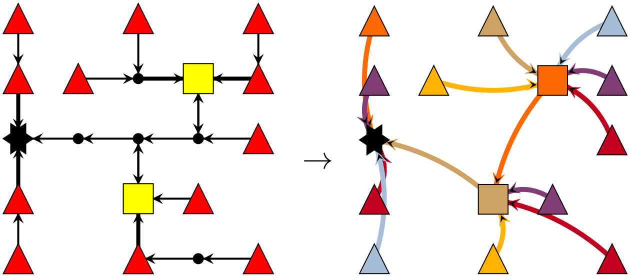
VirtuCast

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

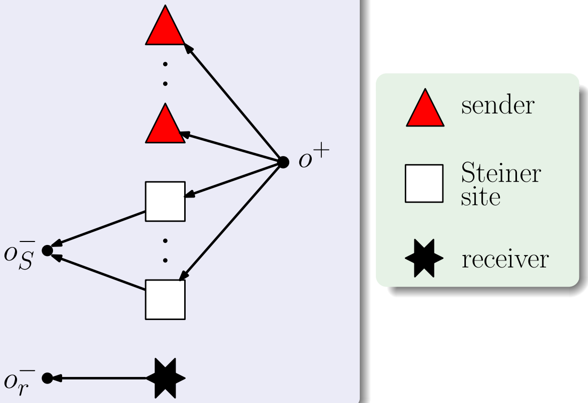
IP Formulation

Extended Graph

Additional nodes

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

Additional edges



IP Formulation I

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

Complete Formulation

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

Connectivity Inequalities

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

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

Exponentially many constraints, but . . .

can be separated in polynomial time.

Decomposition Algorithm

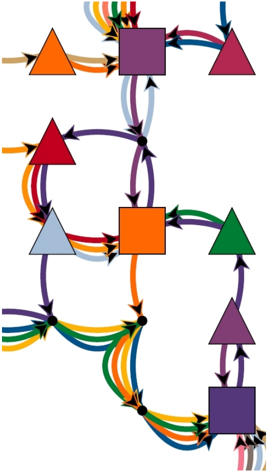
Decomposing flow is non-trivial.

Flow solution is . . .

- not a tree and
- not a DAG [9].

Flow solution . . .

- contains cycles and
- represents *arbitrary* hierarchies.



Outline of Decomposition Algorithm

Iterate

1. select a terminal t
2. construct path P from t towards o_r^- or o_S^-
3. remove one unit of flow along P
4. connect t to the second last node of P and remove t

After each iteration

Problem size reduced by one.

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 remain feasible

  redirect: choose another path from the current node

Theorem

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

Implementation

Overview over Implementation

- VirtuCast is implemented in C++ using SCIP [1].
- Separation of connectivity inequalities is implemented using the Edmonds-Karp algorithm.

VirtuCast Based Heuristics

Overview VirtuCast Based Heuristics

FlowDecoRound: $\sim$ 50 % off optimum, runtime $\sim$ 20 seconds

- based on (simple) flow decomposition and rounding

MultipleShot: $\sim$ 1-7 % off optimum, runtime up to $\sim$ 250 seconds

- treats Steiner site opening variables as probabilities
- iteratively tries to obtain a solution, recomputes LP if unsuccessful
  1. connecting Steiner nodes using 'Virtual Capacitated Prim Algorithm'
  2. min-cost assignment of terminals to Steiner nodes

Greedy Diving: $\sim$ 0.5-3 % off optimum, runtime up to $\sim$ 1500 seconds

- opens single *best* Steiner site until all Steiner sites' variables are fixed
- fixes edge variables afterwards
- recomputes LP including separation procedures
- complex fallback mechanisms

Overview VirtuCast Based Heuristics

FlowDecoRound: $\sim$ 50 % off optimum, runtime $\sim$ 20 seconds

- based on (simple) flow decomposition and rounding

MultipleShot: $\sim$ 1-7 % off optimum, runtime up to $\sim$ 250 seconds

- treats Steiner site opening variables as probabilities
- iteratively tries to obtain a solution, recomputes LP if unsuccessful
  1. connecting Steiner nodes using 'Virtual Capacitated Prim Algorithm'
  2. min-cost assignment of terminals to Steiner nodes

Greedy Diving: $\sim$ 0.5-3 % off optimum, runtime up to $\sim$ 1500 seconds

- opens single *best* Steiner site until all Steiner sites' variables are fixed
- fixes edge variables afterwards
- recomputes LP including separation procedures
- complex fallback mechanisms

Overview VirtuCast Based Heuristics

FlowDecoRound: $\sim$ 50 % off optimum, runtime $\sim$ 20 seconds

- based on (simple) flow decomposition and rounding

MultipleShot: $\sim$ 1-7 % off optimum, runtime up to $\sim$ 250 seconds

- treats Steiner site opening variables as probabilities
- iteratively tries to obtain a solution, recomputes LP if unsuccessful
  1. connecting Steiner nodes using 'Virtual Capacitated Prim Algorithm'
  2. min-cost assignment of terminals to Steiner nodes

Greedy Diving: $\sim$ 0.5-3 % off optimum, runtime up to $\sim$ 1500 seconds

- opens single *best* Steiner site until all Steiner sites' variables are fixed
- fixes edge variables afterwards
- recomputes LP including separation procedures
- complex fallback mechanisms

Overview VirtuCast Based Heuristics

FlowDecoRound: $\sim$ 50 % off optimum, runtime $\sim$ 20 seconds

- based on (simple) flow decomposition and rounding

MultipleShot: $\sim$ 1-7 % off optimum, runtime up to $\sim$ 250 seconds

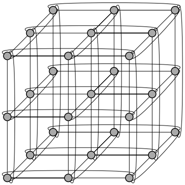
- treats Steiner site opening variables as probabilities
- iteratively tries to obtain a solution, recomputes LP if unsuccessful
  1. connecting Steiner nodes using 'Virtual Capacitated Prim Algorithm'
  2. min-cost assignment of terminals to Steiner nodes

Greedy Diving: $\sim$ 0.5-3 % off optimum, runtime up to $\sim$ 1500 seconds

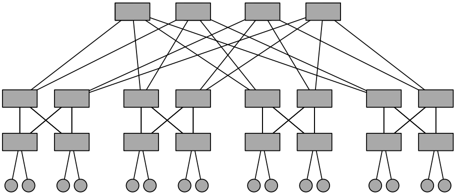
- opens single *best* Steiner site until all Steiner sites' variables are fixed
- fixes edge variables afterwards
- recomputes LP including separation procedures
- complex fallback mechanisms

Computational Evaluation

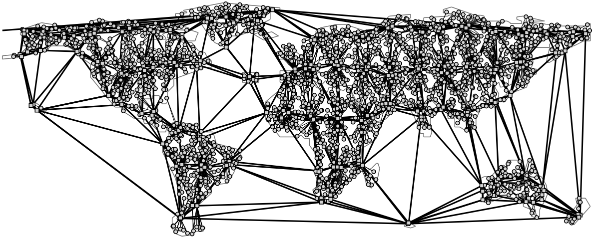
Topologies



3D torus



Fat tree



An ISP topology generated by IGen with 2400 nodes.

Instances

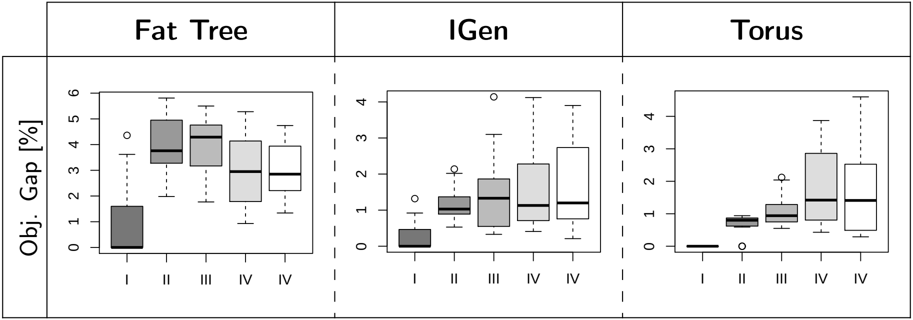
Generation Parameters

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

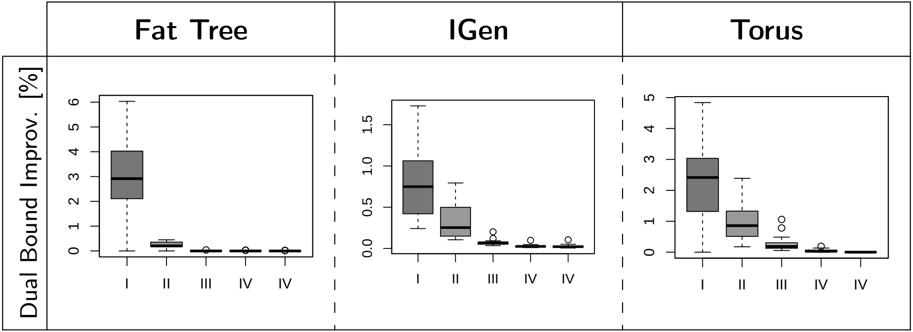
	Nodes	Edges	Steiner Sites	Terminals
Fat tree	1584	14680	720	864
3D torus	1728	10368	432	864
IGen	4000	16924	401	800

Table: Largest graph sizes

Objective Gap



Dual Bound Improvement



Other approaches

Multi-commodity flow formulation

- solved using CPLEX
- fails to compute *root relaxation* for even medium sized instances
- dual bound gaps: 5-20% (fat tree), 3-10% (IGen), 0.1-1% (3D torus)

Combinatorial Greedy Heuristic

- only 'reliable' on fat tree and IGen instances, maximal 1 solved instance on 3D torus instances
- runtime $\sim$ 500 seconds on largest instances
- gap $\sim$ 5-40%

Conclusion

Future Work

Model Extensions

- Prize-collecting variants
- Generalize CVSAP for multiple concurrent multicast / aggregation sessions.
- Try to incorporate service-chaining (EU project UNIFY).

IP formulation

- Try to derive further cuts or even facets for e.g. fat tree instances to improve dual bound.

IP formulation

- Try to derive further cuts or even facets for e.g. fat tree instances to improve dual bound.

Conclusion

Motivation

- Network virtualization enables virtual multicasting / aggregation trees.
- NFV enables placement of processing functionality.
- Goals: Improve scalability or reduce costs.

Summary

- Concise graph theoretic definition of CVSAP.
- Algorithm to solve CVSAP: VirtuCast.
- Computational Evaluation:
  - Feasible to solve realistically sized instances using VirtuCast.
  - Significant Improvement over naive multi-commodity flow IP.

Discussion

Restriction of single-commodity flow model: no path semantics

- iterative aggregation of flows
- no control over path length / latency

Advantages

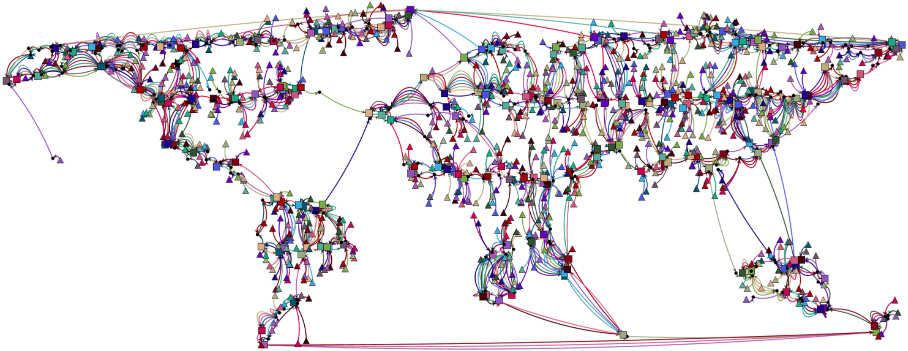
- yields good solutions quickly
- models multicast scenarios accurately
- aggregation compression is limited (at each node)

Applications to BigFoot?

- Can CVSAP be used to model workloads in private clouds?
- If not, which model extensions are necessary?

Thanks for your attention (so far) :).

Project Homepage   OPODIS '13   Technical Report



Temporal Virtual Network Embedding Problem

Problem Statement

Temporal Virtual Network Embedding Problem (TVNEP)

- VNet Requests have additional temporal specification $(\mathbf{t}^{\mathbf{s}}_{R}, \mathbf{t}^{\mathbf{e}}_{R}, \mathbf{d}_{R})$
  - Request R must be embedded in the interval $[\mathbf{t}^{\mathbf{s}}_{R}, \mathbf{t}^{\mathbf{e}}_{R}]$
  - with duration $\mathbf{d}_{R}$
- Temporal flexibilities, if $\mathbf{t}^{\mathbf{e}}_{R} - \mathbf{t}^{\mathbf{s}}_{R} > \mathbf{d}_{R}$, allow scheduling by provider

Objectives

Find embedding of requests *and a schedule* to . . .

- maximize number of embedded requests
- maximize earliness
- maximize energy savings by disabling links / nodes
- . . .

Contribution

Continuous-Time

- Requests may be scheduled at arbitrary points in time
- avoids discretization (errors)

MIP formulations

- Δ: represents state changes only (bad idea)
- Σ: represent state changes explicitly (better idea)
- $c\Sigma$: Σ-model using symmetry & state-space reductions (best idea)

Greedy Heuristic

- based on $c\Sigma$-model

Publication

IPDPS 2014 [15], joint work with Stefan Schmid and Anja Feldmann

Applications

Applications

Data center

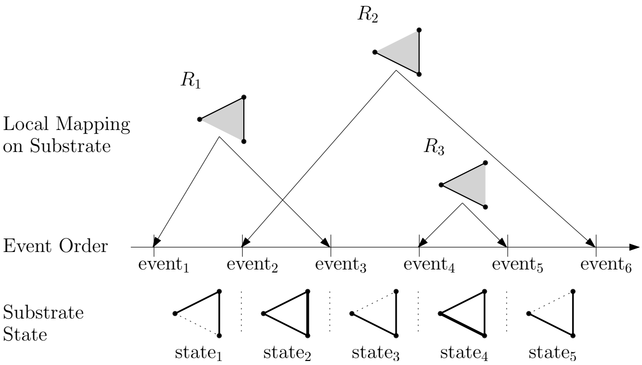
- e.g. MapReduce cycles through different phases, traffic only during 30-60% of execution [17]
- price incentives for customers and providers to allow for / harness temporal flexibility [8]

Wide area networks

- Google uses SDN in the WAN to connect data centers [5]
- scheduling of bandwidth-intensive synchronizations
  - is necessary to achieve good utilization and resource isolation
  - is enabled by central SDN control

Overview of Models

Event Point Abstraction



General approach

- compute *local* mapping of Requests onto substrate
- linearly order starts and ends of requests via mapping on event points
- compute states and check feasibility

Δ-Model

Idea

- only compute state changes via *conditional assignment*

$$\Delta_{\mathbf{e}_i}(N_s) = \begin{cases} +alloc_V(R_1, N_s) & \text{, if start of } R_1 \text{ is mapped on } \mathbf{e}_i \\ -alloc_V(R_1, N_s) & \text{, if end of } R_1 \text{ is mapped on } \mathbf{e}_i \\ \vdots & \\ +alloc_V(R_k, N_s) & \text{, if start of } R_k \text{ is mapped on } \mathbf{e}_i \\ -alloc_V(R_k, N_s) & \text{, if end of } R_k \text{ is mapped on } \mathbf{e}_i \end{cases}$$

- substrate state s_i is computed inductively via

$$\sum_{j=1}^{i} \Delta_{\mathbf{e}_i} \ .$$

Δ-Model: LP-Smearings!

MIP implementation of conditional assignment: big-M

$$\Delta_{\mathbf{e}_i}(N_s) \leq + \mathit{alloc}_V(R, N_s) + \mathbf{c_S}(N_s)(1 - \chi^{+}_{\mathrm{R}_1}(\mathbf{e}_i)) \tag{1}$$

$$\Delta_{\mathbf{e}_i}(N_s) \geq + \mathit{alloc}_V(R, N_s) - \mathbf{c_S}(N_s)(1 - \chi^{+}_{\mathrm{R}_1}(\mathbf{e}_i)) \cdot 2 \tag{2}$$

$$\Delta_{\mathbf{e}_i}(N_s) \leq - \mathit{alloc}_V(R, N_s) + \mathbf{c_S}(N_s)(1 - \chi^{-}_{\mathrm{R}_1}(\mathbf{e}_i)) \cdot 2 \tag{3}$$

$$\Delta_{\mathbf{e}_i}(N_s) \geq - \mathit{alloc}_V(R, N_s) - \mathbf{c_S}(N_s)(1 - \chi^{-}_{\mathrm{R}_1}(\mathbf{e}_i)) \tag{4}$$

LP-Smearings

- assuming assignments to be $1/2$, state changes can be set to ≤ 0 for all resources and all events!
- very bad relaxations in practice

Σ-Model

Insight

- State allocations must be modeled more explicitly.

$$\mathrm{a}_{\mathsf{R}} : \mathcal{S} \times (\mathbf{V_S} \cup \mathbf{E_S}) \to \mathbb{R}_{\geq 0}$$

$$\Sigma(R, \mathbf{e}_i) = \sum_{j=1,\ldots,i} \chi^{+}_{\mathsf{R}}(\mathbf{e}_j, \mathsf{R}) - \sum_{j=i,\ldots,|\mathcal{E}|} \chi^{-}_{\mathsf{R}}(\mathbf{e}_j, \mathsf{R})$$

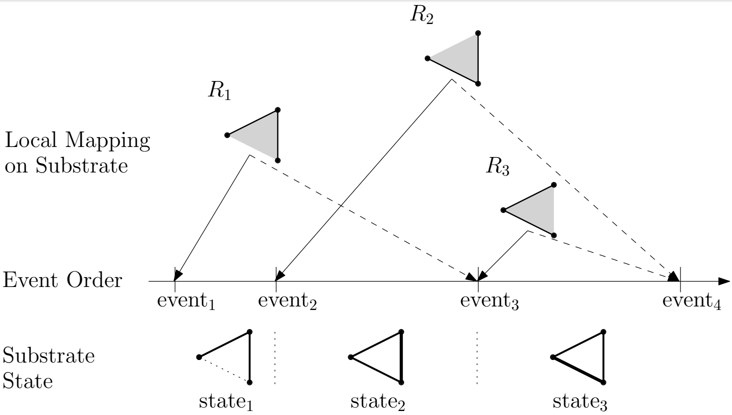
$$\mathrm{a}_{\mathbf{s}_i}(\mathsf{R}, N_s) \geq alloc_V(\mathsf{R}, N_s) - \mathbf{c_V}(N_s) \cdot (1 - \Sigma(R, \mathbf{e}_i))$$

$$\mathbf{c_S}(r) \geq \sum_{\mathsf{R} \in \mathcal{R}} \mathrm{a}_{\mathsf{R}}(\mathbf{s}_i, r)$$

$c\Sigma$-Model

Compactification

- Partial order on the end of requests suffices.
- Yields *symmetry* reduction and *state-space* reduction.



Optimizations: Dependency Graph User Cuts

Dependency Graph User Cuts

Dependency Graph

- Nodes are *abstract* start and end events of requests.
- Edge (u, v) exists, if u must take place *before* v.
- Dependency graph is a DAG.

State-space reduction

- If u is preceeded by n abstract events, u cannot be mapped on the first n event points.
- Analogously for trailing event points.

User Cuts

- If u is mapped on event $\mathbf{e}_i$, then all trailing abstract events must happen *after* $\mathbf{e}_i$.
- Improves relaxation.

Computational Evaluation

Computational Evaluation: One-day workload

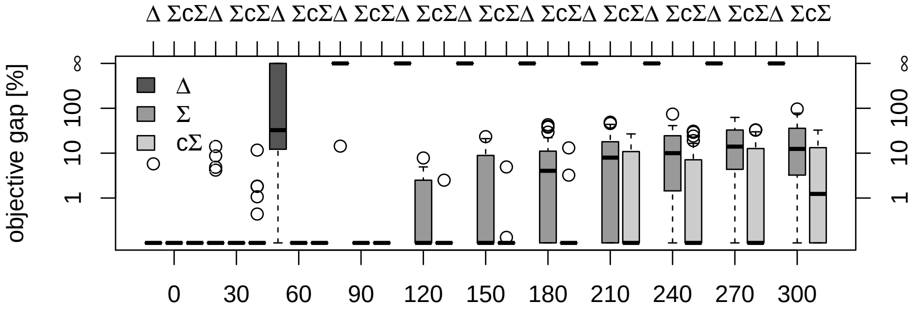
Scenario

- consider scenarios with 20 requests over time
- poisson inter-arrival time
- weibull duration (heavy tailed)
- node-mappings are fixed
- link-mappings are not fixed
- 0, 30, 60, 90, 120, . . . , 300 minutes of flexibility

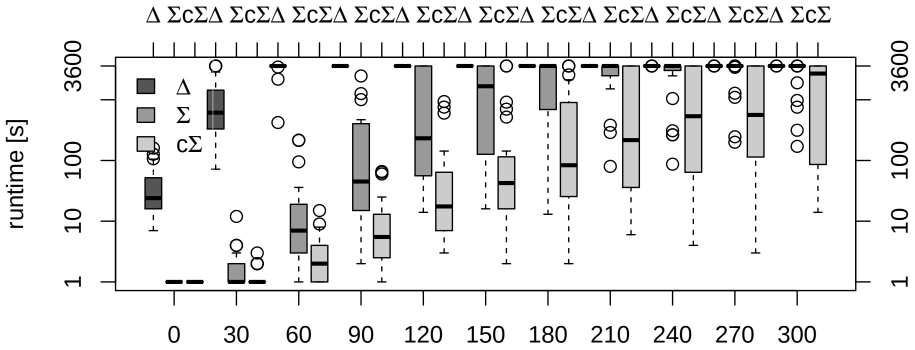
Task

- Decide which requests to embed,
- when to embed and
- how to route the flow.

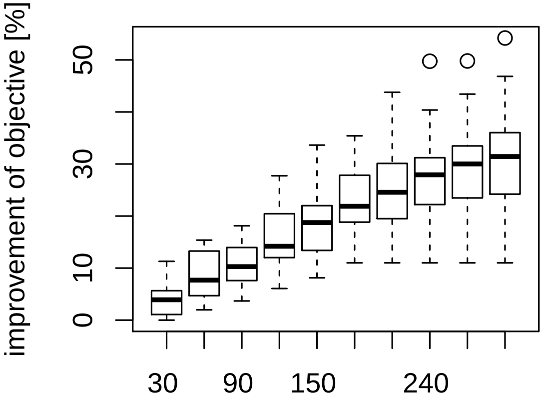
Objective Gap



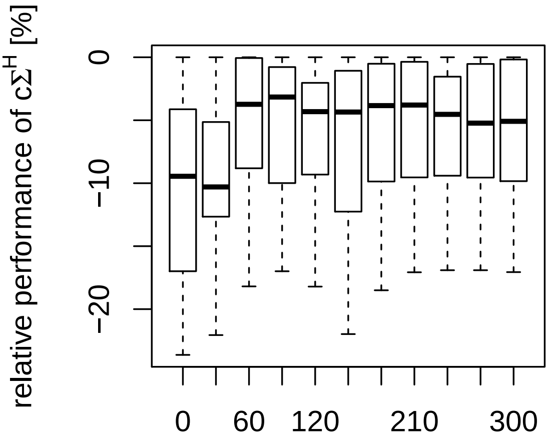
Runtime


Δ ΣcΣΔ ΣcΣΔ ΣcΣΔ ΣcΣΔ ΣcΣΔ ΣcΣΔ ΣcΣΔ ΣcΣΔ ΣcΣΔ ΣcΣΔ ΣcΣ
Δ
Σ
cΣ
runtime [s]
3600
100
10
1
0
30
60
90
120
150
180
210
240
270
300

Benefit of Flexibility



Performance of Greedy Heuristic



Discussion & Future Work

Future Work

- Incorporate flexible duration of requests.
- Allow for more complex scenarios: requests consider of request groups and dependencies between them.
- Develop heuristics for other objectives as well.
- Evaluate our approach in conjunction with embedding heuristics.

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] 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.

[5] S. J. et al.
B4: experience with a globally-deployed software defined wan.
In *Proc. ACM SIGCOMM*, pages 3–14, New York, NY, USA, 2013. ACM.

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

[8] L. Mai, E. Kalyvianaki, and P. Costa.
Exploiting time-malleability in cloud-based batch processing systems.
In *Proc. 7th LADIS*, 2013.

[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] M. Rost.
Optimal Virtualized In-Network Processing with Applications to Aggregation and Multicast.
Master's thesis, Technische Universität Berlin, Germany, 2014.

[13] 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.

[14] 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

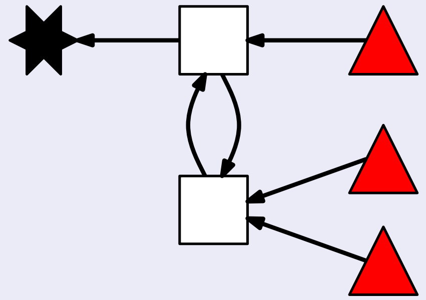
[15] M. Rost, S. Schmid, and A. Feldmann.
It's about time: On optimal virtual network embeddings under temporal flexibilities.
In *Parallel & Distributed Processing (IPDPS), 2014 IEEE 28th International Symposium on*. IEEE, 2014.

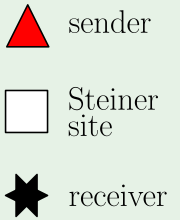
[16] S. Shi.
A Proposal for A Scalable Internet Multicast Architecture.
Technical Report WUCS-01-03, Washington University, 2001.

[17] A. Singla, A. Singh, K. Ramachandran, L. Xu, and Y. Zhang.
Proteus: a topology malleable data center network.
In *Proc. 9th ACM HotNets*, 2010.

Example

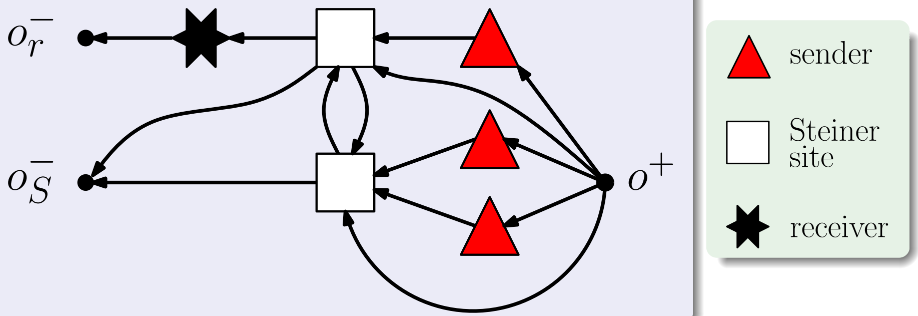
Scenario





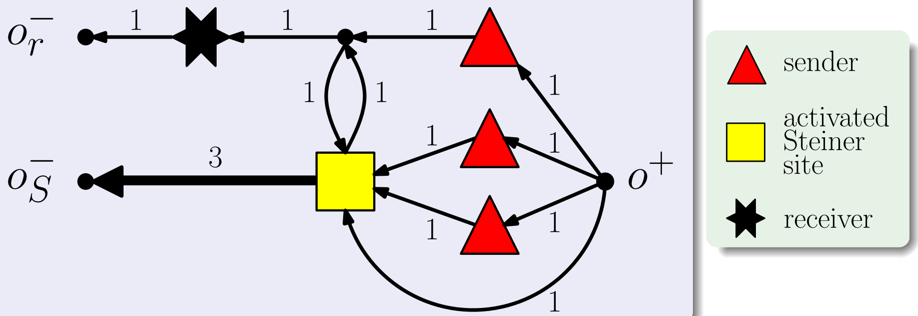
Example

Extended Graph

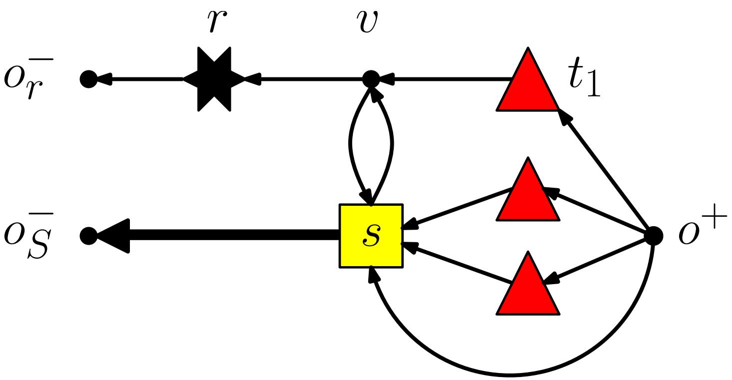


Example

Solution

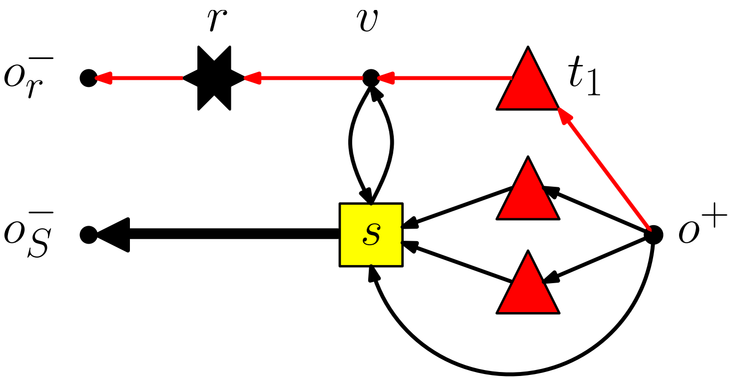


Decomposition Example I



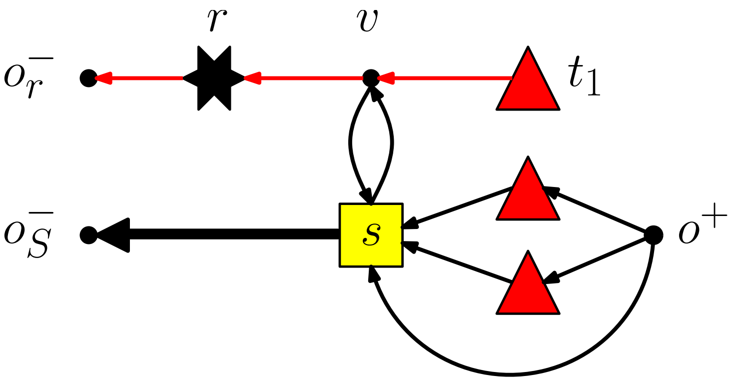
Decomposition Example I

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



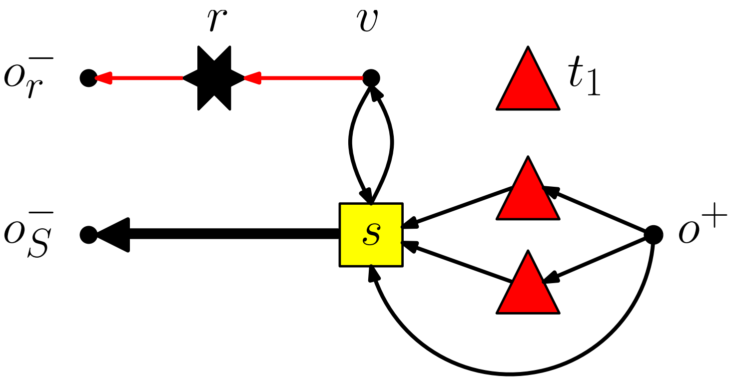
Decomposition Example I

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



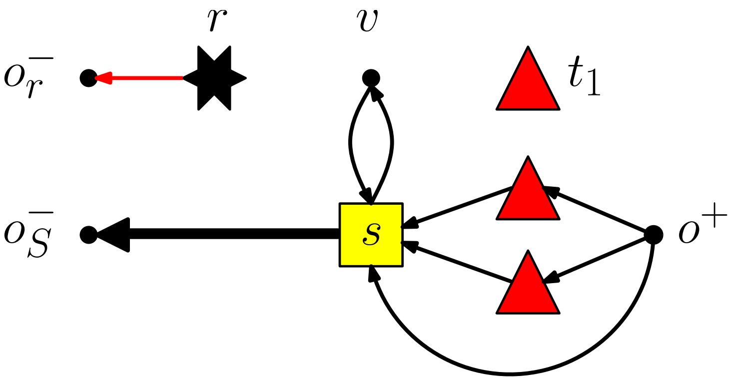
Decomposition Example I

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

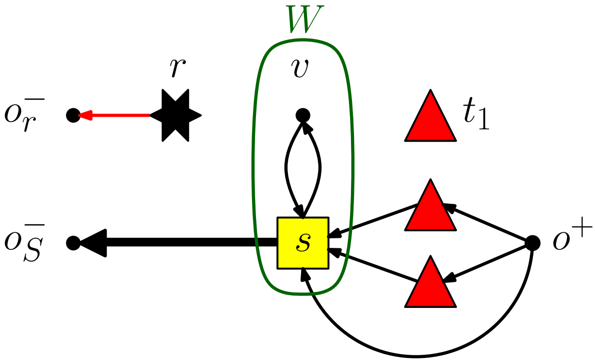


Decomposition Example I

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



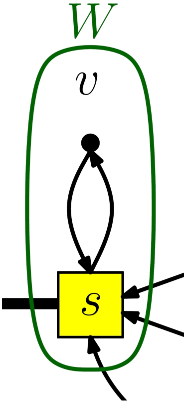
Redirecting Flow



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



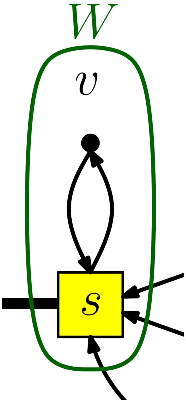
Redirection towards o_S^- is possible!

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

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^-.

Redirecting Flow



Redirection towards o_S^- is possible!

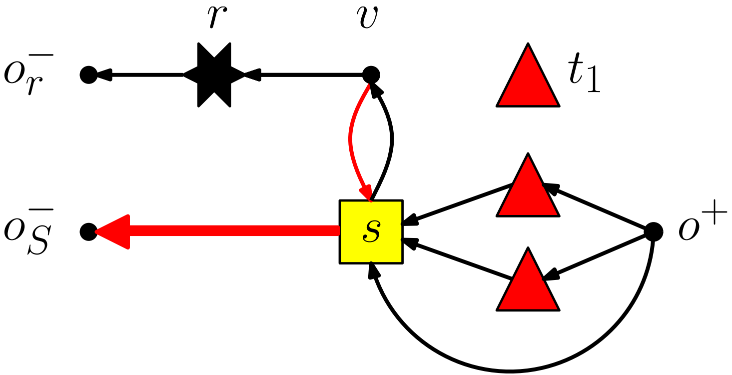
There exists a path from v towards o_S^- in W.

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^-.

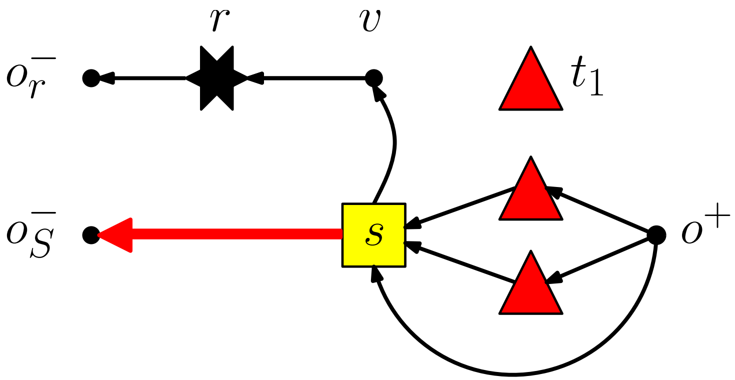
Decomposition Example II

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

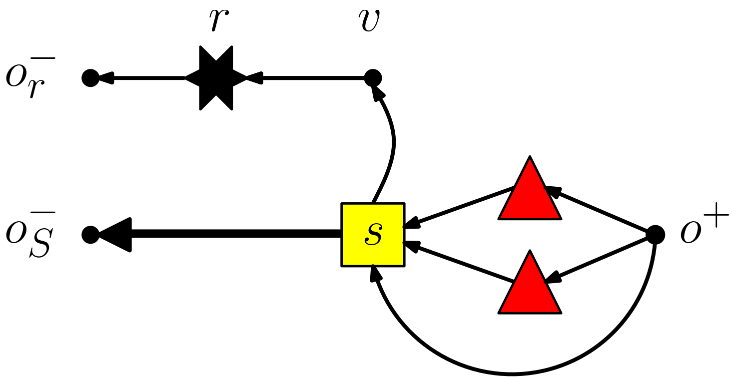


Decomposition Example II

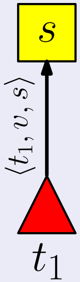
$$P = \langle o^+, t_1, v, s, o_S^- \rangle$$



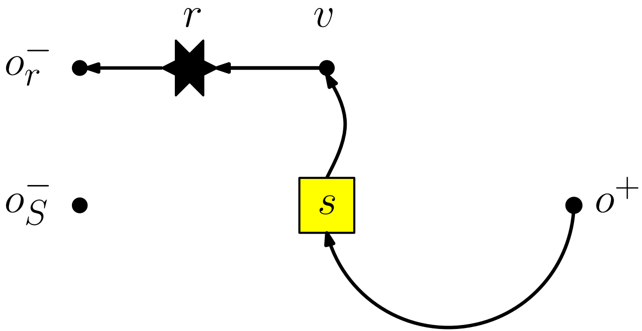
Decomposition Example II



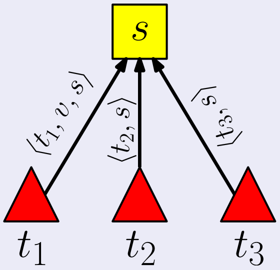
Solution



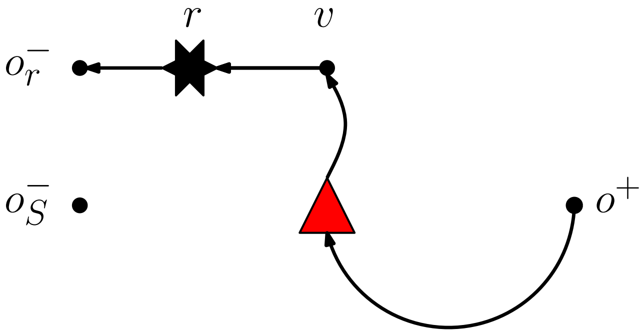
Decomposition Example II

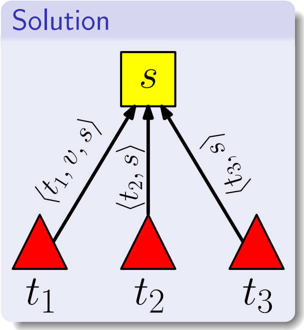


Solution

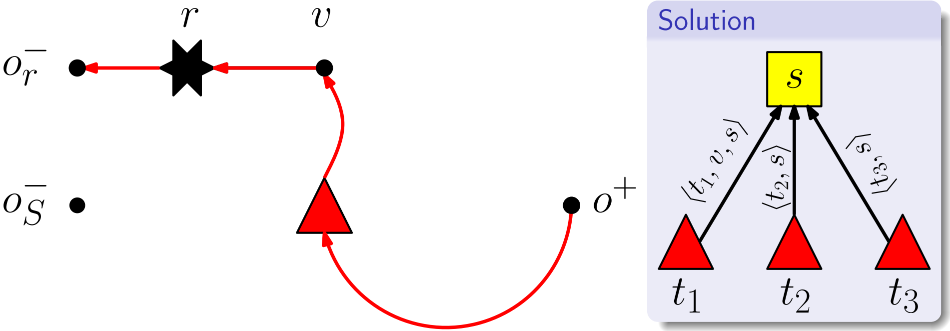


Decomposition Example II



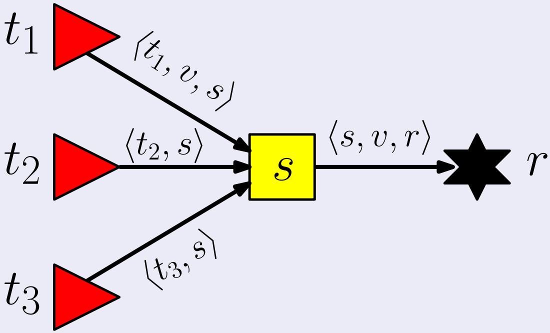


Decomposition Example II



Decomposition Example II

Final Solution



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
- Give weak approximation algorithm

Shi: Scalability in Overlay Multicasting [16]

- Provided heuristic and showed improvement in scalability.