

# VirtuCast: Multicast and Aggregation with In-Network Processing

OPODIS 2013

Matthias Rost & Stefan Schmid

TU Berlin & Telekom Innovation Laboratories (T-Labs)

December 19th, 2013  
*EURECOM*

# Our Work in a Nutshell

## Virtualization on the rise: SDN + NFV

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

## Our Answer

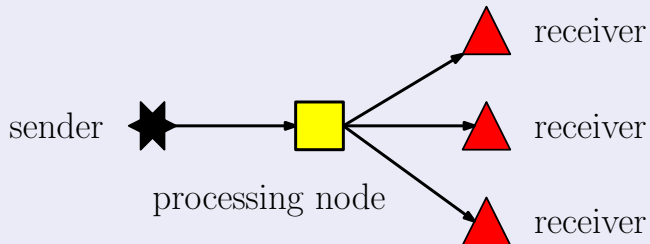
- *New Model*: Constrained Virtual Steiner Arborescence Problem
- *New Algorithm*: VirtuCast

## Objective: Jointly minimize ...

- bandwidth
- number of processing nodes

## Communication Schemes: Multicast

processing = duplication + reroute



# Communication Schemes: Multicast

processing = duplication + reroute

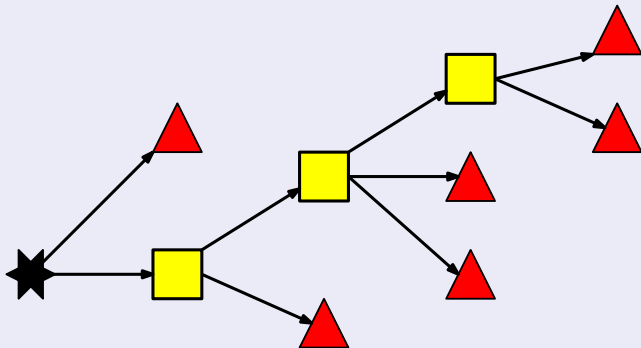
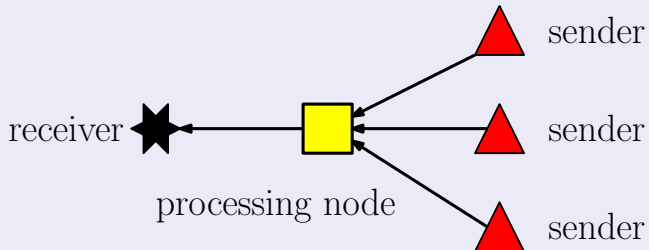


Figure: Hierarchy of processing nodes

# Communication Schemes: Aggregation

processing = merge + reroute



# Communication Schemes: Aggregation

processing = merge + reroute

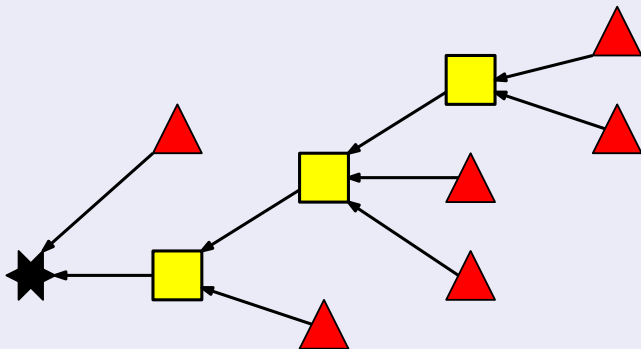


Figure: Hierarchy of processing nodes



# Introductory Example

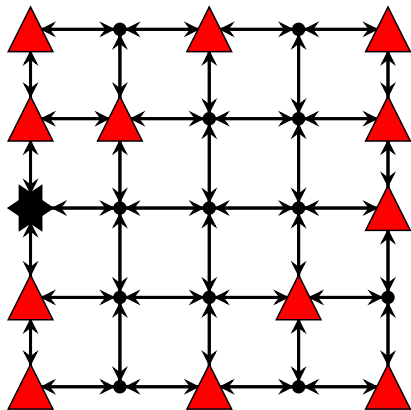
## Aggregation scenario

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

- 43 edges
- 0 processing nodes



receiver



sender

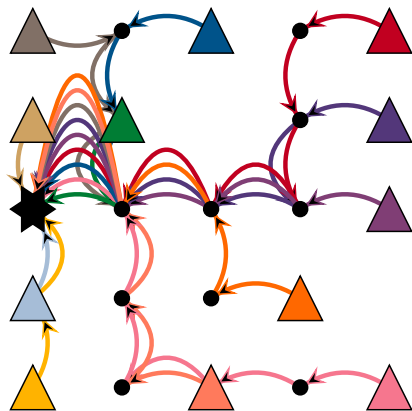


Figure: Unicast solution



# With in-network processing at all nodes

## Solution Method

- Steiner arborescence

## Solution uses

- 16 edges
- 9 processing nodes

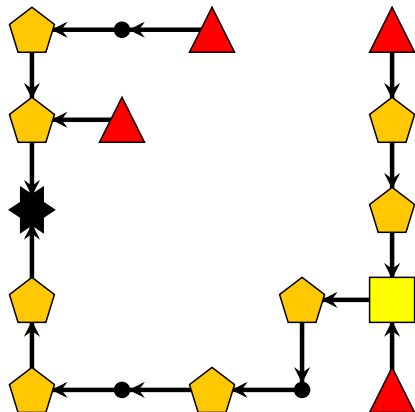
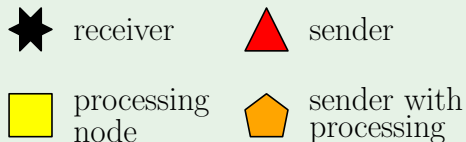
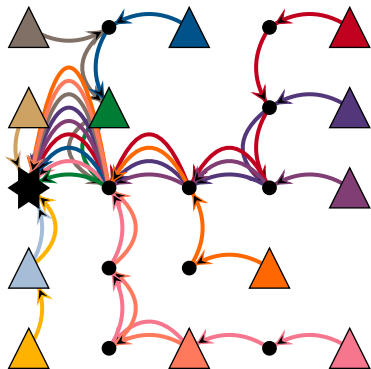
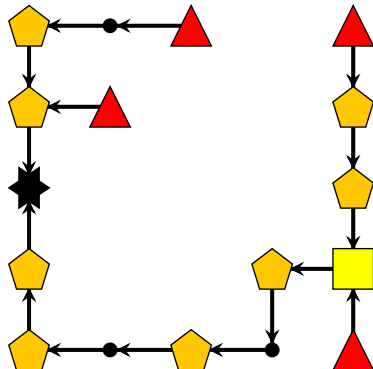


Figure: Aggregation tree

## How to Trade-off?



vs.



# Our Solution: CVSAP & VirtuCast

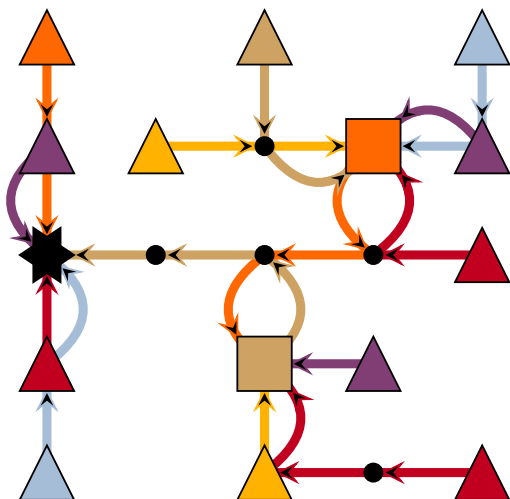
## Solution uses

- 26 edges
- 2 processing nodes

★ receiver

△ sender

□ processing node



# Our Solution: CVSAP & VirtuCast

## Solution uses

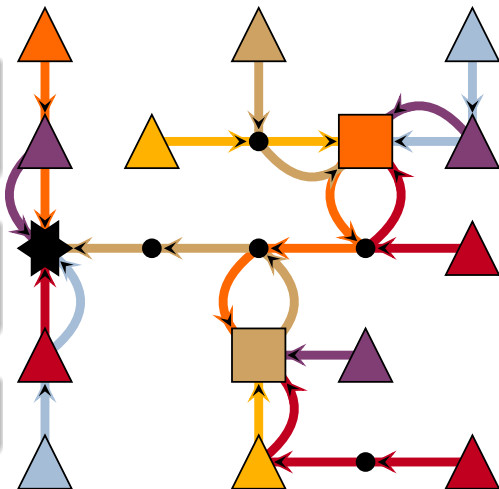
- 26 edges
- 2 processing nodes

## New Model

Constrained Virtual Steiner  
Arborescence Problem (CVSAP)

## New Solution Method

VirtuCast algorithm



# 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 Constrained 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 Constrained 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

# CVSAP Solution

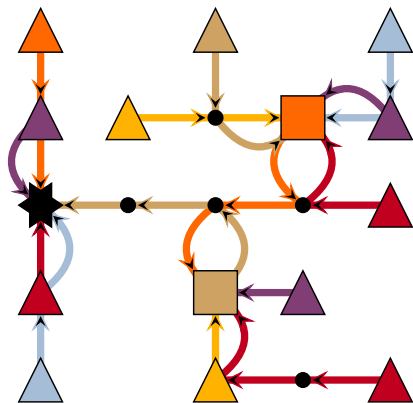
## Virtual Links

sender & processing nodes are connected via *paths*

★ receiver

△ sender

□ processing node



# Solution Structure

## Virtual Arborescence

- directed tree towards root  $r$
- terminals are leaves
- non Steiner sites are forbidden
- if a Steiner site is included, processing functionality is placed
- edges represent paths in underlying network

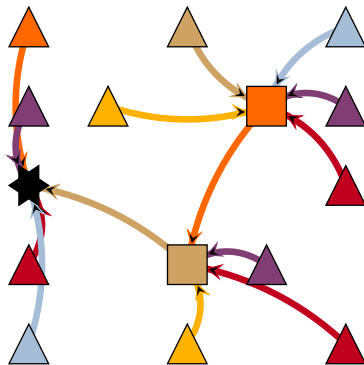


Figure: Virtual Arborescence

# Constrained Virtual Steiner Arborescence Problem

## Definition

Find a Virtual Arborescence such that

### Degree constraints

- degrees of root  $r$  and Steiner sites are bounded by  $u_r$  and  $u_S$

### Reasoning

- aggregation nodes are not able to handle arbitrary many incoming flows
- multicasting nodes are not able to duplicate an incoming stream arbitrarily many times

# Constrained Virtual Steiner Arborescence Problem

## Definition

Find a Virtual Arborescence such that

- Degree constraints

### Edge capacities

- edge capacities in the underlying network are not violated

# Constrained Virtual Steiner Arborescence Problem

## Definition

Find a Virtual Arborescence such that

- Degree constraints
- Edge capacities

minimizing

sum of edge costs + sum of installation costs

# Applications

## Applications

	Network	Application	Technology, e.g.
multicast	ISP	service replication / cache placement [8, 9]	middleboxes / NFV + SDN
	backbone	optical multicast [5]	ROADM <sup>1</sup> + SDH
	all	application-level multicast [12]	different
aggregation	sensor network	value & message aggregation [4, 6]	source routing
	ISP	network analytics: GigaScope [3]	middleboxes / NFV + SDN
	data center	big data / map-reduce: Camdoop [2]	SDN

<sup>1</sup>reconfigurable optical add/drop multiplexer



# Gigascoppe [3]

## Problem

- Network monitoring and analysis of large communication networks
- 500 GB data / day in 2003

## Idea

- Develop structured query language (similar to SQL) for analysis
- Move analysis close to source to reduce traffic
- Implemented and used at AT&T (2003)

## Modeled via aggregation CVSAP

- single receiver instantiates query
- processing nodes can perform e.g. join operations, reducing output

# Camdoop [2]

## Problem

- Optimize data-center for MapReduce (running on the whole system)
- At Facebook: in 82% of cases output has only 5.4% size of input

## Idea

- Use direct-interconnect 3D-torus network
- Devise disjoint set of paths for shuffle operations
- Aggregate values (if possible) at each node to reduce traffic

## Modeled via aggregation CVSAP

- processing nodes can perform aggregation of data
- receiver(s) run final reduce operation and store result

## Solution Approach

# Overview of Solution Approach

## CVSAP

- novel problem
- inapproximable (if  $P \neq NP$ )

## Goal: exact algorithm

- solves CVSAP to optimality
- non-polynomial runtime

# Overview of Solution Approach

## CVSAP

- novel problem
- inapproximable (if  $P \neq NP$ )

## Goal: exact algorithm

- solves CVSAP to optimality
- non-polynomial runtime

## Motivation for exact algorithms

- application dependent: allows trading-off runtime with solution quality, e.g. when designing new networks
- baseline for heuristics

# Overview of Solution Approach

## CVSAP

- novel problem
- inapproximable (if  $P \neq NP$ )

## Goal: exact algorithm

- solves CVSAP to optimality
- non-polynomial runtime

## Motivation for exact algorithms

- application dependent: allows trading-off runtime with solution quality, e.g. when designing new networks
- baseline for heuristics

## Solution Approach: Integer Programming (IP)

- lower bounds are computed on-the-fly

# Our Algorithms for CVSAP

Developed two different IP formulations

## Multi-Commodity Flow based

- bad lower bounds
- cannot be used on large instances

## Single-Commodity Flow based

- good lower bounds
- can be used to solve large instances
- **VirtuCast**

# Single- vs. Multi-Commodity Flows

## Single-Commodity Flow Formulation

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

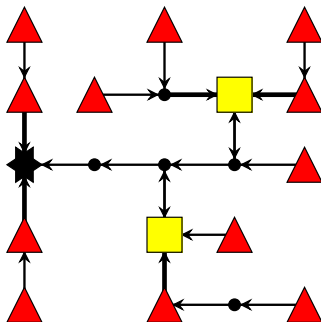


Figure: Single-commodity



# Single- vs. Multi-Commodity Flows

Example: 6000 edges and 200 Steiner sites

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

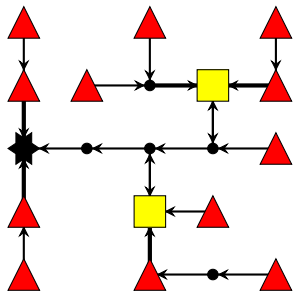


Figure: Single-commodity

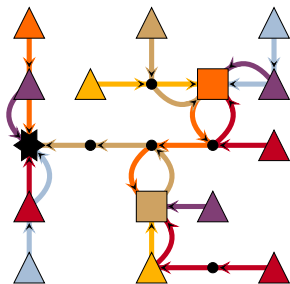


Figure: Multi-commodity

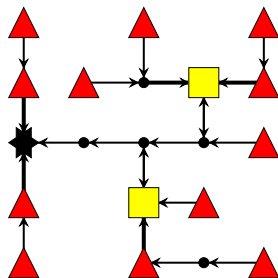
VirtuCast

# VirtuCast Algorithm

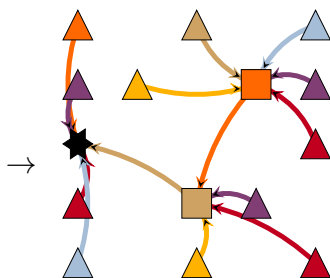
## Outline of VirtuCast

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

How to decompose?



(a) IP solution



(b) Virtual Arborescence

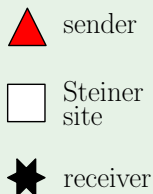
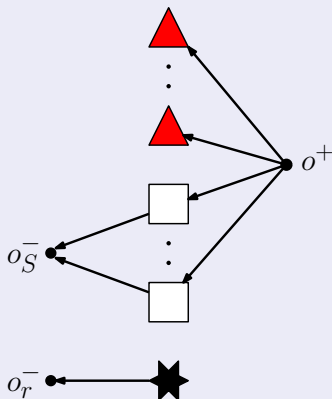
# IP Formulation

# Extended Graph

## Additional nodes

- source  $o^+$
- sinks  $o_r^-$  and  $o_s^-$

## Additional edges



# Outline of IP Formulation

## Variables

$$\forall s \in S. \quad x_s \in \{0, 1\}$$

$$\forall e \in E_{\text{ext.}} \quad f_e \in \mathbb{Z}_{\geq 0}$$

## Constraints

- 1 single-commodity flow on extended graph
- 2 capacity constraints
- 3 connectivity inequalities

# Outline of IP Formulation

## Variables

$$\begin{aligned}\forall s \in S. \quad & x_s \in \{0, 1\} \\ \forall e \in E_{\text{ext}}. \quad & f_e \in \mathbb{Z}_{\geq 0}\end{aligned}$$

## Constraints

- 1 single-commodity flow on extended graph
  - terminals receive one unit of flow
  - activated Steiner sites receive one unit of flow
  - flow preservation on all original nodes
- 2 capacity constraints
- 3 connectivity inequalities

# Outline of IP Formulation

## Variables

$$\begin{aligned}\forall s \in S. \quad & x_s \in \{0, 1\} \\ \forall e \in E_{\text{ext}}. \quad & f_e \in \mathbb{Z}_{\geq 0}\end{aligned}$$

## Constraints

- 1 single-commodity flow on extended graph
- 2 capacity constraints
  - enforce degree constraints
  - enforce that edge capacities hold
- 3 connectivity inequalities



# Outline of IP Formulation

## Variables

$$\forall s \in S. \quad x_s \in \{0, 1\}$$

$$\forall e \in E_{\text{ext.}} \quad f_e \in \mathbb{Z}_{\geq 0}$$

## Constraints

- 1 single-commodity flow on extended graph
- 2 capacity constraints
- 3 connectivity inequalities

## Connectivity Inequalities

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

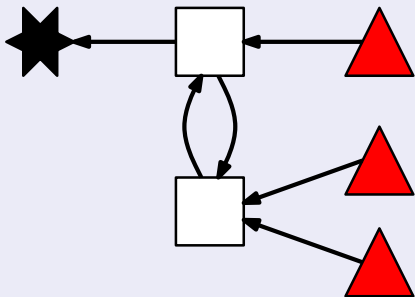
From each activated Steiner site, there exists a path towards  $o_r^-$ .

Exponentially many constraints, but ...

can be separated in polynomial time.

# Example

## Scenario



sender



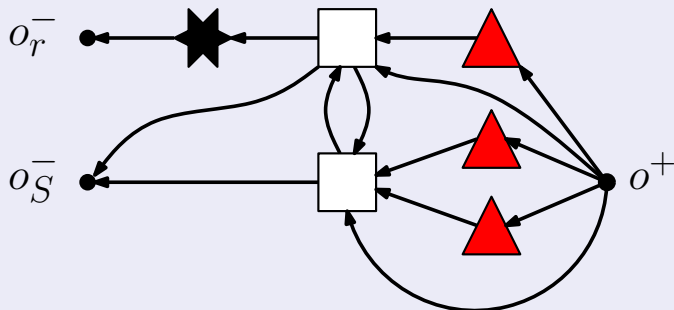
Steiner  
site



receiver

## Example

## Extended Graph



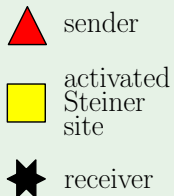
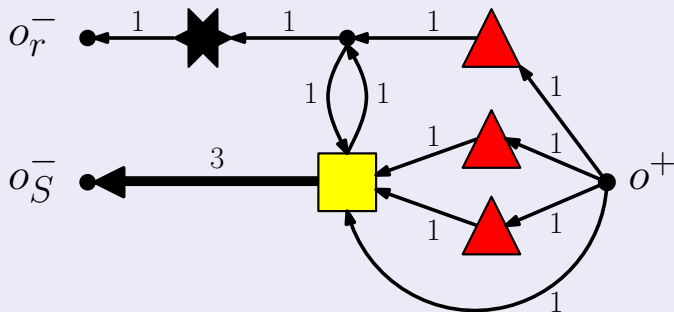
sender

Steiner  
site

receiver

## Example

## Solution



# Decomposition Algorithm

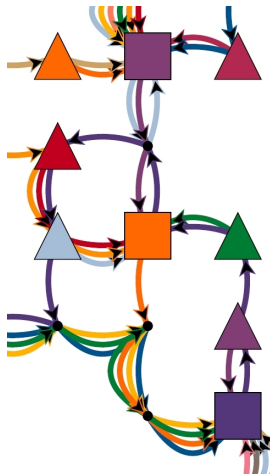
# Decomposing flow is non-trivial.

## Flow solution is ...

- not a tree and
- not a DAG [7].

## 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

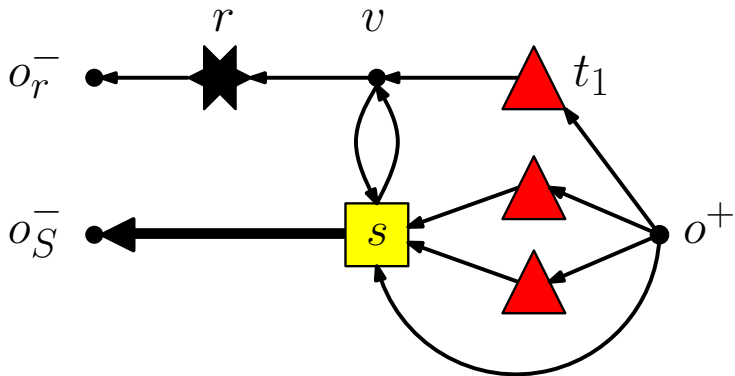
Reduced problem must be feasible

Removing flow must not invalidate any connectivity inequalities.

Principle: Repair & Redirect

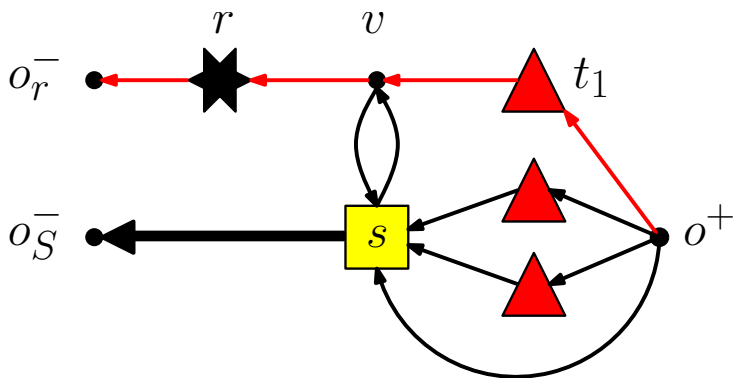
- decrease flow on path edge by edge
- if connectivity inequalities are violated
  - repair** increment flow on edge to remain feasible
  - redirect** choose another path from the current node

## Decomposition Example I



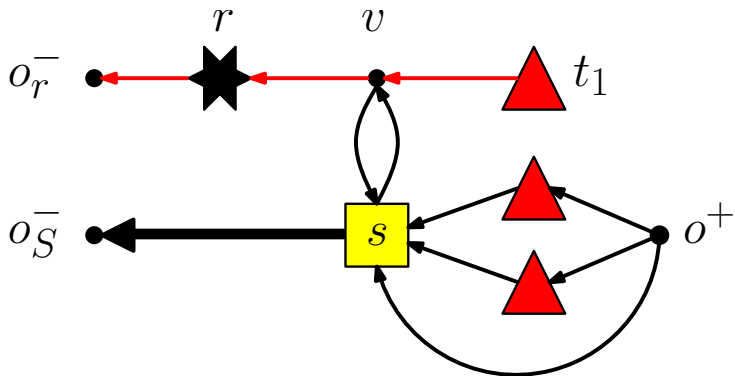
## Decomposition Example I

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



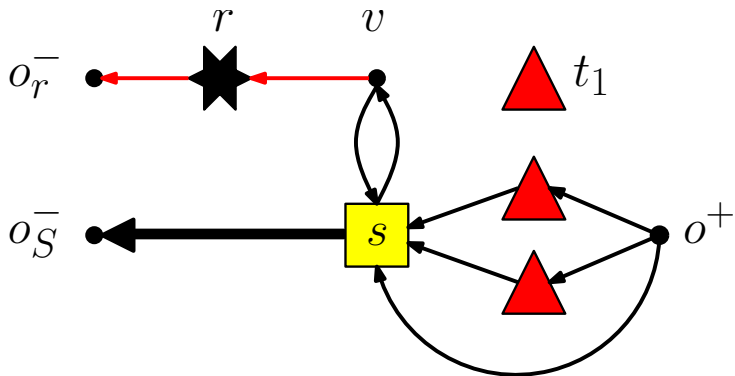
## Decomposition Example I

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



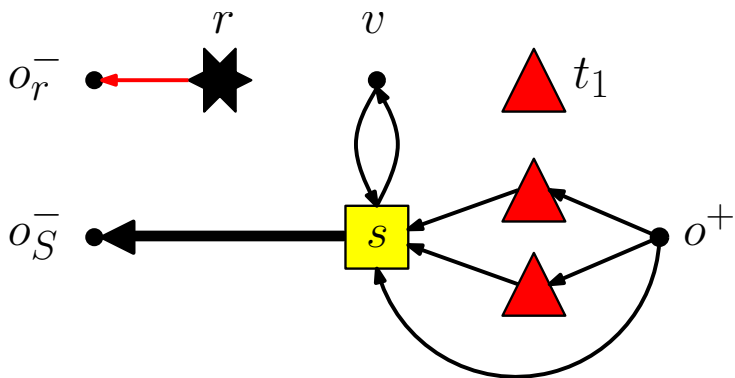
## Decomposition Example I

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

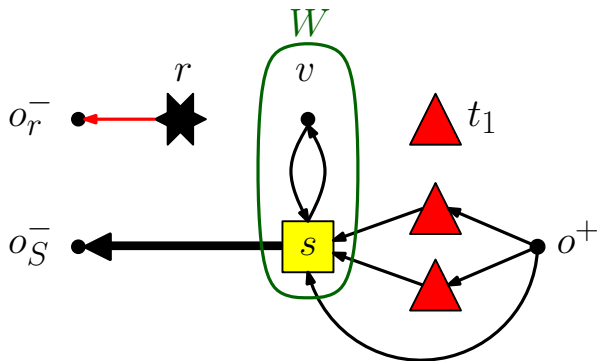


## Decomposition Example I

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



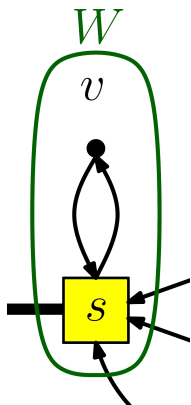
# Redirecting Flow



Violation of Connectivity Inequality

$$f(\delta_{E_{\text{ext}}}^+(W)) \geq x_s \quad \forall W \subseteq V_G, s \in W \cap S \neq \emptyset$$

# Redirecting Flow

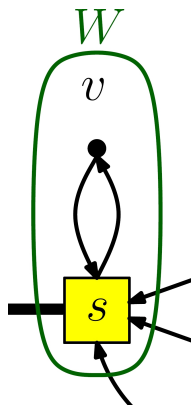


Redirection towards  $o_S^-$  is possible!

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



# Redirecting Flow



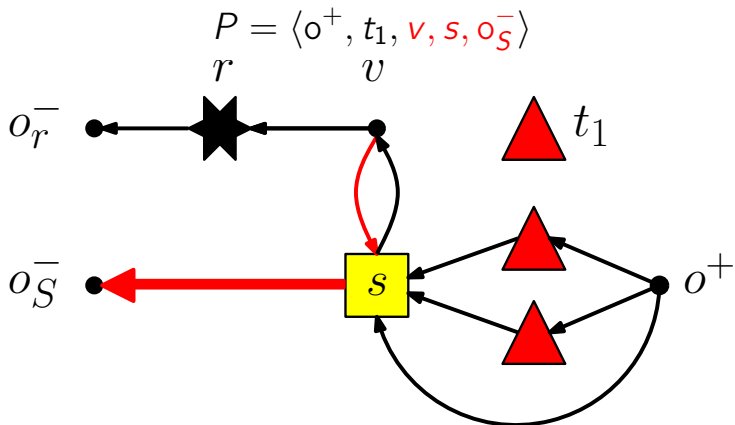
Redirection towards  $o_s^-$  is possible!

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

## Reasoning

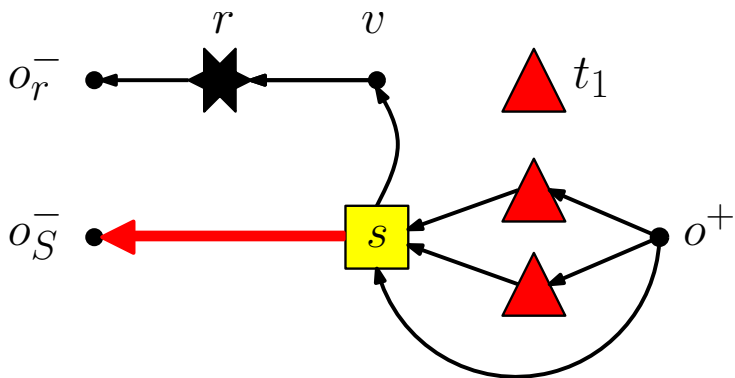
- ① Flow preservation holds within  $W$ .
- ②  $s$  could reach  $o_r^-$  via  $v$  before the reduction of flow.
- ③  $v$  receives at least one unit of flow.
- ④ Flow leaving  $v$  must eventually terminate at  $o_s^-$ .

## Decomposition Example II

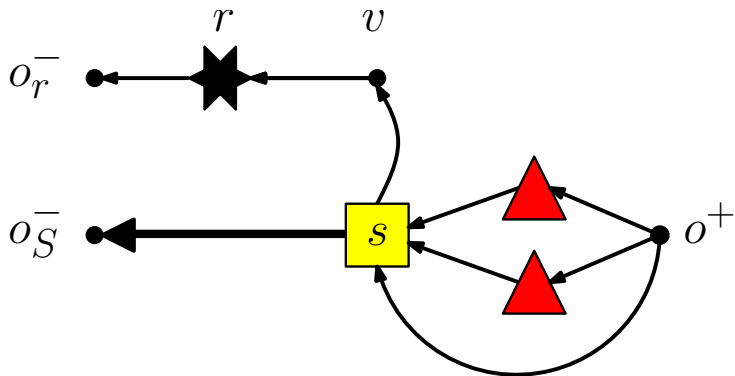


## Decomposition Example II

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



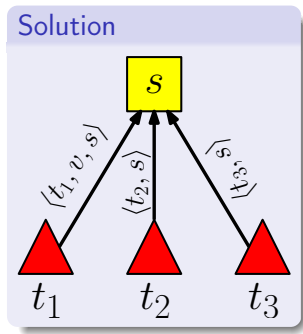
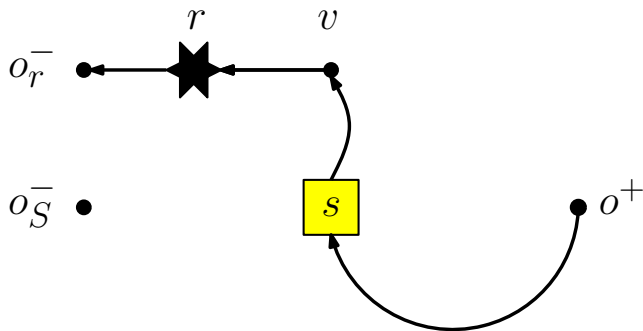
## Decomposition Example II



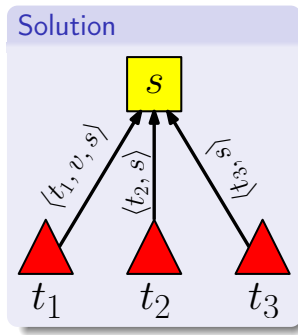
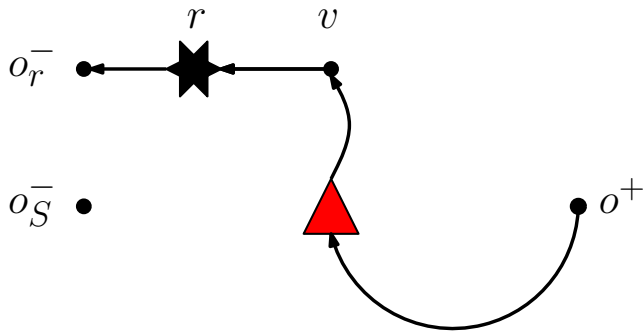
Solution



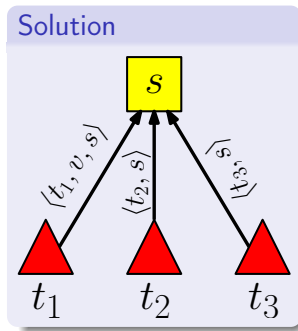
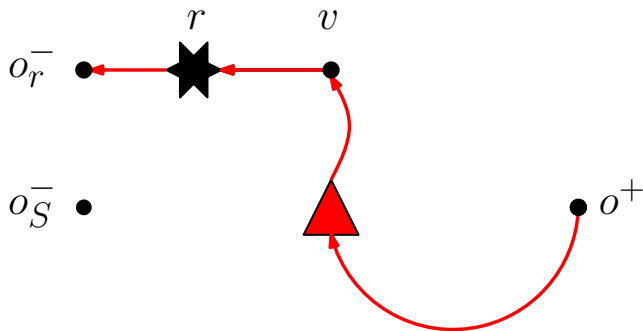
## Decomposition Example II



## Decomposition Example II

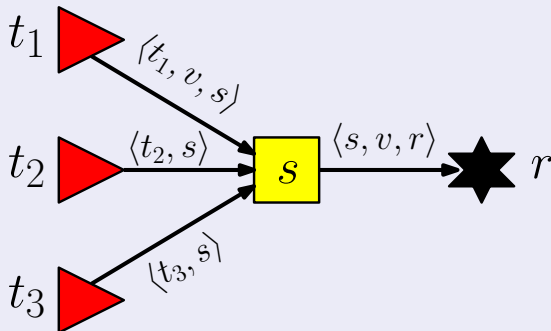


## Decomposition Example II



# Decomposition Example II

## Final Solution





# Runtime of Decomposition Algorithm

## Theorem

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

# Proof of Correctness

# Outline of Proof

## Cost-preserving mapping



## Theorem

*Algorithm VirtuCast solves CVSAP to optimality.*

# Implementation

# Overview over Implementation

- VirtuCast is implemented in C++ using SCIP [1].
- Separation of connectivity inequalities is implemented using the Edmonds-Karp algorithm.
- FlowDecoRound heuristic to generate primal solutions during the branch-and-bound process [11].

FlowDecoRound Heuristic

# Outline

## Goal

Develop a fast primal heuristic for generating solutions during the branch-and-bound process.

## Important Note

The algorithm takes as input a fractional solution  $(\hat{x}, \hat{f}) \in \mathcal{F}_{LP}$ .

# Outline

- 1 Select terminal randomly and connect it according to local flow decomposition
  - If node  $t$  is connected to an inactive Steiner site  $s \in S$ , place  $s$  into the set of terminals.
- 2 Connect all unconnected terminals using shortest paths.
- 3 Prune active Steiner nodes.



# Phase 1

- ① Randomly choose a terminal  $t$ 
  - (a) Compute flow decomposition from  $t$  to  $o_S^-$ ,  $o_r^-$  with flow  $f(o^+, t)$
  - (b) Prune infeasible paths
  - (c) Choose a path uniformly at random according to the flow value and connect  $t$  to the respective node.
  - (d) If  $t$  was connected to inactive Steiner site  $s \in S$ , place  $s$  into the set of terminals.
  - (e) Remove  $t$  from the set of terminals.

## Phase 2

- ② Choose an unconnected terminal  $t$  randomly
  - Compute shortest path towards any of the activated Steiner nodes, while not introducing a cycle into the Virtual Arborescence.
  - Connect node according to found shortest path or abort if no path was found.

## Phase 3

- ③ Iterate over active Steiner nodes  $s \in S$  in decreasing order of its cost divided by the number of incoming connections.
  - Temporarily, disconnect all nodes connected to  $s$  and remove the outgoing connection of  $s$ .
  - Try to reconnect all unconnected nodes using shortest paths.
  - If a cheaper solution was found, accept it. Otherwise restore previous solution.

# Computational Evaluation

# Test Set I: $n \times n$ Grid Graphs

## uniform costs

- uniform edge costs
- uniform installation costs

## Sizes

$n$	nodes	edges	Steiner sites	terminals
16	256	960	51	64
20	400	1520	80	100

# Test Set II: Synthetic ISP Topologies [10]

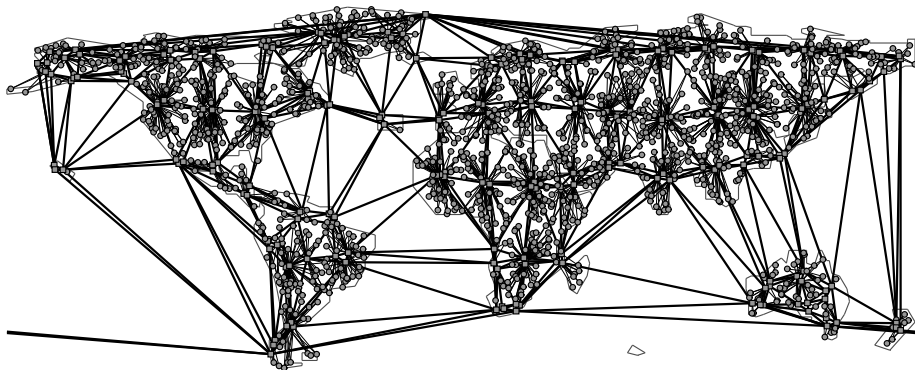


Figure: IGen topology with 1600 nodes

# Test Set II: Synthetic ISP Topologies [10]

## non-uniform costs

- metric edge costs
- uniformly distributed installation costs

## Size

Name	nodes	edges	Steiner sites	terminals
IGen.1600	1600	6816	200	300
IGen.3200	3200	19410	400	600

# Computational Setup

## General

- 25 instances for each test set and each graph size.
- Terminate experiments after 2 hours of runtime.

## Multi-commodity flow formulation

- is solved with CPLEX

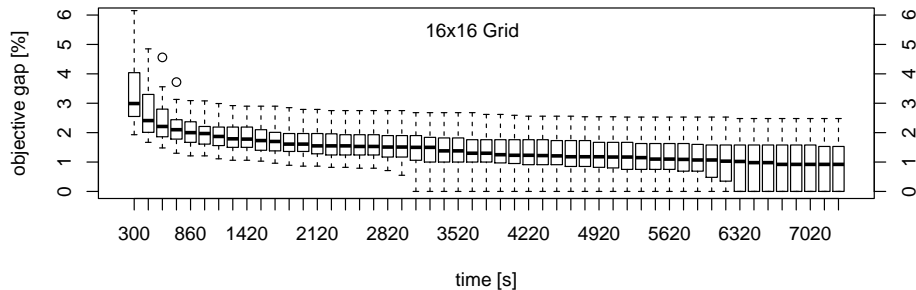


# VirtuCast Performance

# VirtuCast - Objective Gap: Grids

16 × 16

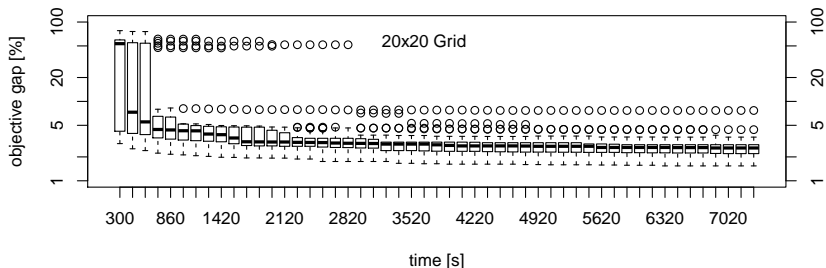
- After 30 minutes: median gap around 2 %
- After 120 minutes: median gap around 1 %



# VirtuCast - Objective Gap: Grids

20 × 20

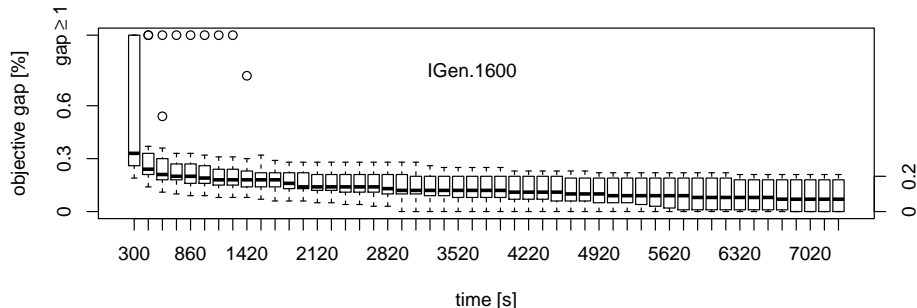
- After 30 minutes: median gap around 4 %
- After 120 minutes: median gap around 3 %



# VirtuCast - Objective Gap: IGen

## IGen.1600

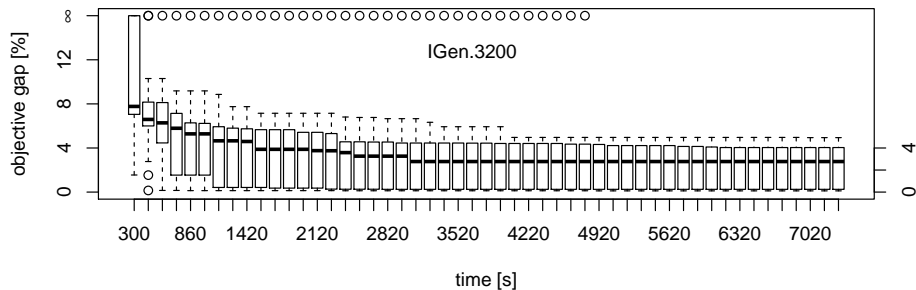
- After 30 minutes: gap below 0.3 %
- After 120 minutes: median gap below 0.1 %



# VirtuCast - Objective Gap: IGen

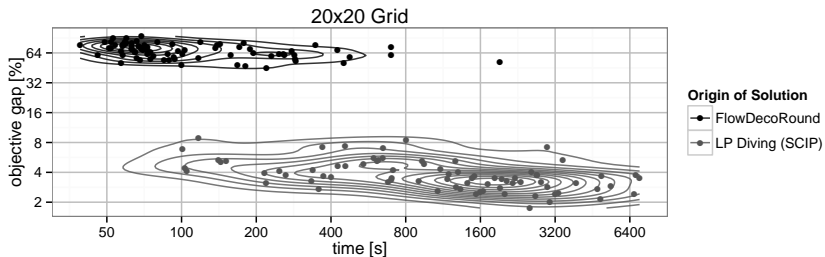
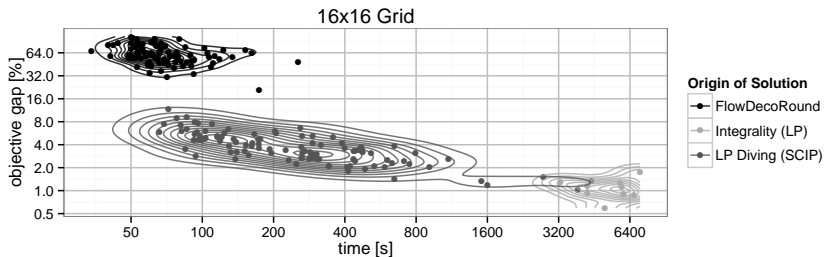
## IGen.3200

- After 30 minutes: median gap around 4 %
- After 120 minutes: median gap around 3 %

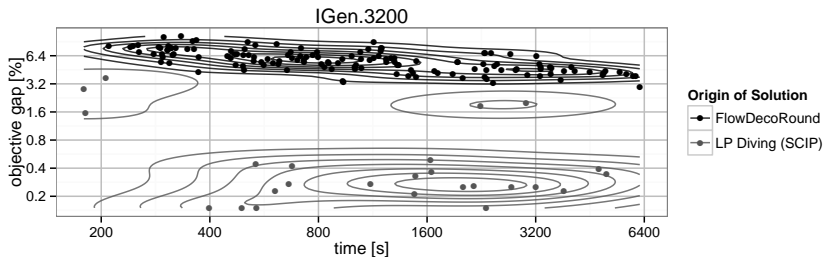
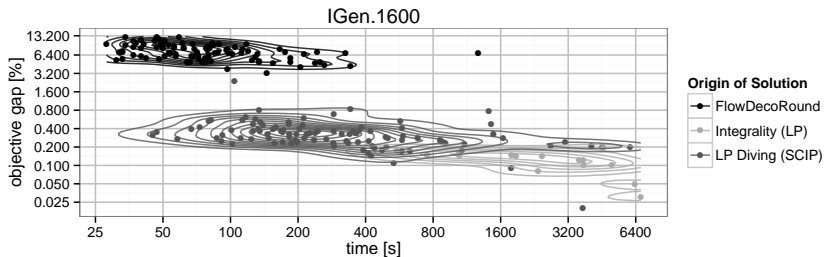


# Performance of FlowDecoRound

# Performance of FlowDecoRound: Grids



# Performance of FlowDecoRound: IGen





## Comparison with MCF

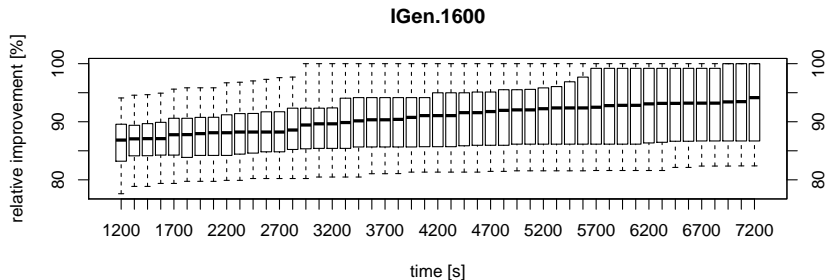
# Computational Results of MCF

## IGen.3200

Cannot be solved (efficiently) using MCF formulation: more than 6,000,000 variables

## IGen.1600: Strength of MCF formulation

VirtuCast's lower bound improves upon MCF's lower bound by around 90% w.r.t to the best known solution.



Conclusion

## Related Work

### Molnar: Constrained Spanning Tree Problems [7]

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

### Oliveira et. al: Flow Streaming Cache Placement Problem [9]

- Consider a weaker variant of multicasting CVSAP without bandwidth
- Give weak approximation algorithm

### Shi: Scalability in Overlay Multicasting [12]

- Provided heuristic and showed improvement in scalability.

# Future Work

## Model Extensions

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

## Heuristics for CVSAP

- Currently testing different approaches.
- Algorithmically challenging problem.

# 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.

Project homepage

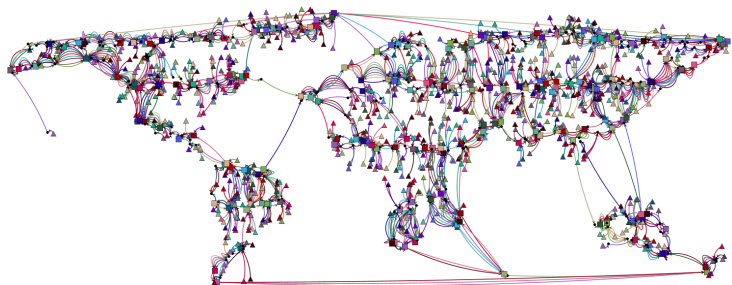
[www.net.t-labs.tu-berlin.de/~stefan/cvsap.html](http://www.net.t-labs.tu-berlin.de/~stefan/cvsap.html)

OPODIS '13

[link.springer.com/chapter/10.1007/978-3-319-03850-6\\_16](http://link.springer.com/chapter/10.1007/978-3-319-03850-6_16)

Technical Report

[arxiv.org/abs/1310.0346](http://arxiv.org/abs/1310.0346)





## Other Current Work

# IPDPS Paper 2014

## It's About Time: On Optimal Virtual Network Embeddings under Temporal Flexibilities

- Joint work with Stefan Schmid und Anja Feldmann
- Algorithms for embedding & scheduling virtual networks under temporal flexibilities

# Outline

## Temporal Virtual Network Embedding Problem

- Requests consisting of node allocations and link allocations need to be embedded over time
- Temporal specification allows for flexibility in scheduling requests

## Task

Find embedding of requests and a schedule to ...

- maximize number of embedded requests
- minimize makespan
- ...

# Contribution

## Continuous-Time

- Requests may be scheduled at (real valued) times
- Avoids discretization (errors)
- Uses fewer variables

## IP formulations

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

## Greedy Heuristic

- based on  $c\Sigma$ -model

# Computational Evaluation: One-day workload

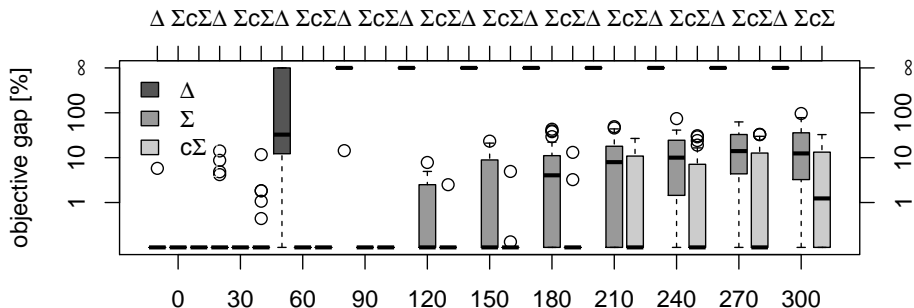
## Scenario

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

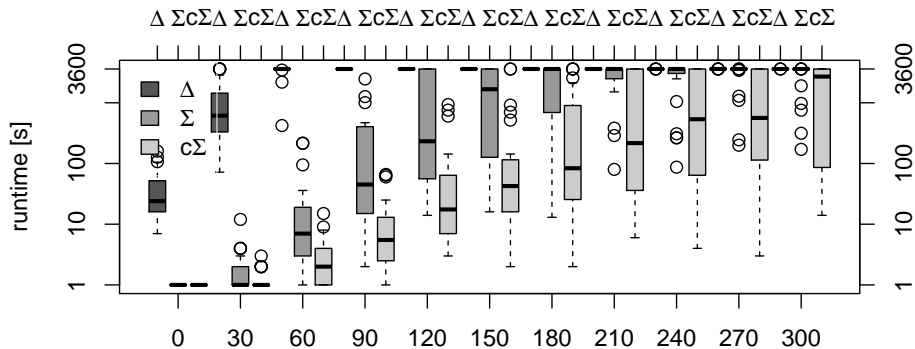
## Task

- Decide which requests to embed &
- when to embed &
- how to route flow.

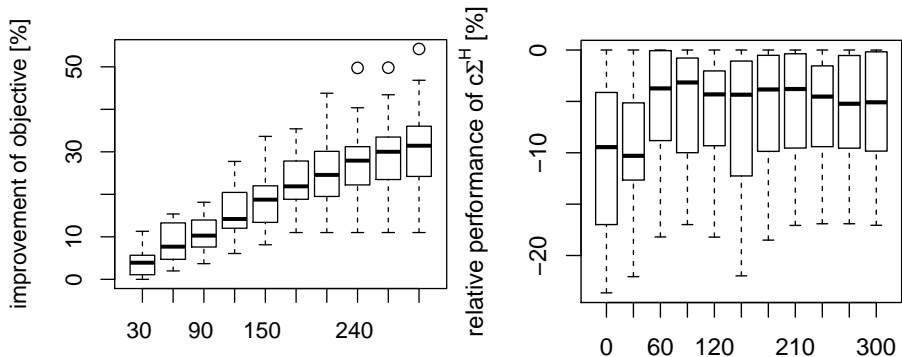
## Objective Gap



## Runtime



# Benefit of Flexibility & Performance of Heuristic





## Discussion & Future Work

### Future Work

- Incorporate flexible duration of requests
- Develop heuristics for other objectives as well
- Evaluate our approach in conjunction with embedding heuristics

### Applications to BigFoot?

- Scheduling and routing of (time insensitive) background data transfers?
- Plan bandwidth intense jobs like VM migration ahead?
- ...

# 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.  
Gigascop: 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.  
01285913.pdf.
- [5] C. Hermsmeyer, E. Hernandez-Valencia, D. Stoll, and O. Tamm.  
Ethernet aggregation and core network models for efficient and reliable IPTV services.  
*Bell Labs Technical Journal*, 12(1):57–76, 2007.

## References II

- [6] B. Krishnamachari, D. Estrin, and S. Wicker.  
Modelling data-centric routing in wireless sensor networks.  
In *IEEE infocom*, volume 2, pages 39–44, 2002.
- [7] M. Molnár.  
Hierarchies to Solve Constrained Connected Spanning Problems.  
Technical Report Irimm-00619806, University Montpellier 2, LIRMM, 2011.
- [8] 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.
- [9] 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.
- [10] B. Quoitin, V. Van den Schrieck, P. François, and O. Bonaventure.  
IGen: Generation of router-level Internet topologies through network design heuristics.  
In *Proc. 21st International Teletraffic Congress (ITC)*, pages 1–8, 2009.

## References III

- [11] M. Rost and S. Schmid.  
*CVSAP-Project Website*.  
<http://www.net.t-labs.tu-berlin.de/~stefan/cvsap.html>, 2013.
- [12] S. Shi.  
A Proposal for A Scalable Internet Multicast Architecture.  
Technical Report WUCS-01-03, Washington University, 2001.