Beyond the Classic Virtual Network Embedding: Temporality and Optimal In-Network Processing

Matthias Rost

Technische Universität Berlin

August 12th, 2014
Lehrstuhl für Kommunikationsnetze, TU München

Part I: How to schedule networking tasks

- How and when should data center synchronize their data?
- How and when should tasks be deployed *within* data centers?

Part II: How to deploy in-network processing services

- How can we deploy a multicasting service, e.g. IPTV?
- How can we perform in-network analytics efficiently, e.g. distributed IDS?

Part I: How to schedule networking tasks

- How and when should data center synchronize their data?
- How and when should tasks be deployed *within* data centers?

Part II: How to deploy in-network processing services

- How can we deploy a multicasting service, e.g. IPTV?
- How can we perform in-network analytics efficiently, e.g. distributed IDS?

Algorithmic approach: Mixed-Integer Programming (MIP)

- <u>outline</u> how to obtain *good* or *strong* formulations
- <u>outline</u> how to obtain polynomial time heuristics

Part I: How to schedule networking tasks

- How and when should data center synchronize their data?
- How and when should tasks be deployed *within* data centers?

Part II: How to deploy in-network processing services

- How can we deploy a multicasting service, e.g. IPTV?
- How can we perform in-network analytics efficiently, e.g. distributed IDS?

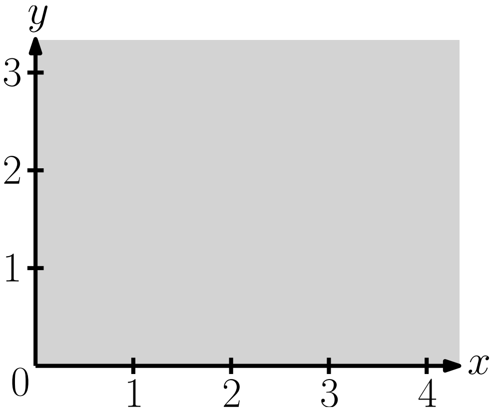
Algorithmic approach: Mixed-Integer Programming (MIP)

- <u>outline</u> how to obtain *good* or *strong* formulations
- <u>outline</u> how to obtain polynomial time heuristics

Please, if you have any questions, ask right away!

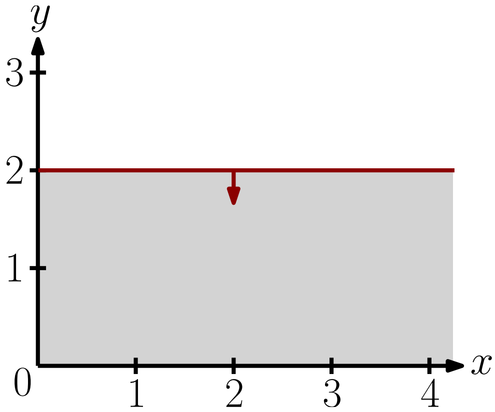
Mixed-Integer Programming Crash Course

Excursion: Linear Programming



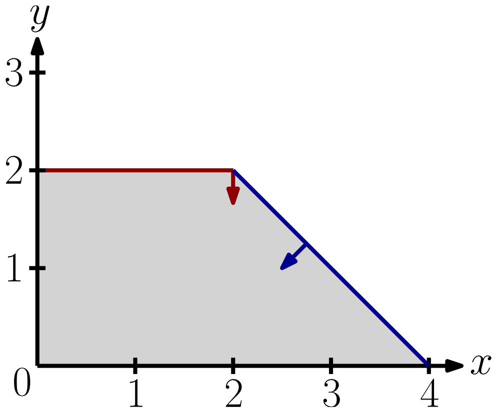
$$x \geq 0$$
$$y \geq 0$$

Excursion: Linear Programming



$$x \geq 0$$
$$y \geq 0$$
$$y \leq 2$$

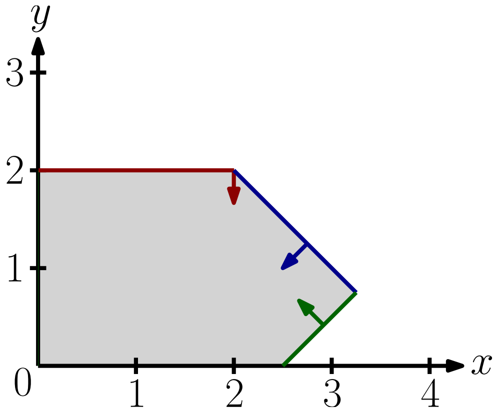
Excursion: Linear Programming



$$x \geq 0$$
$$y \geq 0$$

$$y \leq 2$$
$$x + y \leq 4$$

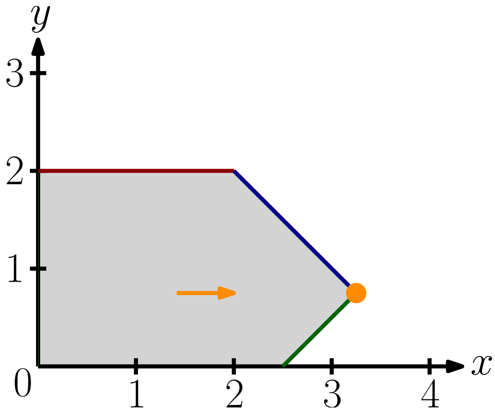
Excursion: Linear Programming



$$x \geq 0$$
$$y \geq 0$$

$$y \leq 2$$
$$x + y \leq 4$$
$$x - y \geq 2.5$$

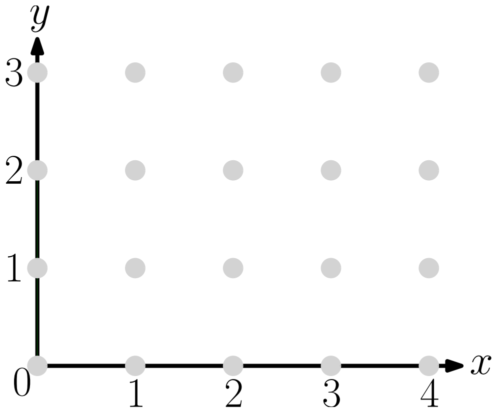
Excursion: Linear Programming



$$x \geq 0$$
$$y \geq 0$$
$$y \leq 2$$
$$x + y \leq 4$$
$$x - y \geq 2.5$$
$$\max x$$

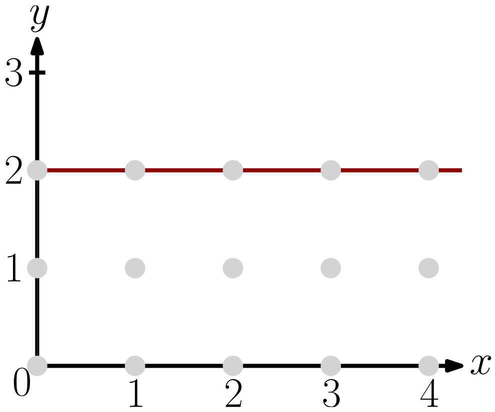
Can be solved in polynomial time!

Excursion: Integer Programming



$x \in \mathbb{N}$
$y \in \mathbb{N}$

Excursion: Integer Programming

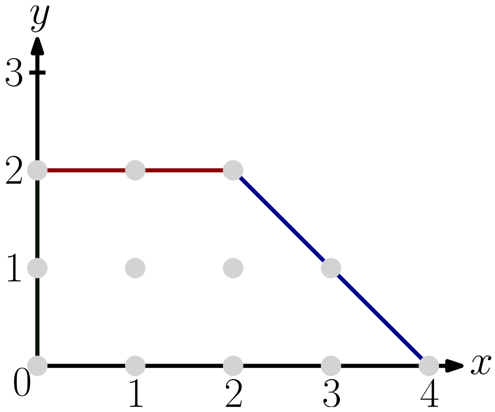


$x \in \mathbb{N}$

$y \in \mathbb{N}$

$y \leq 2$

Excursion: Integer Programming



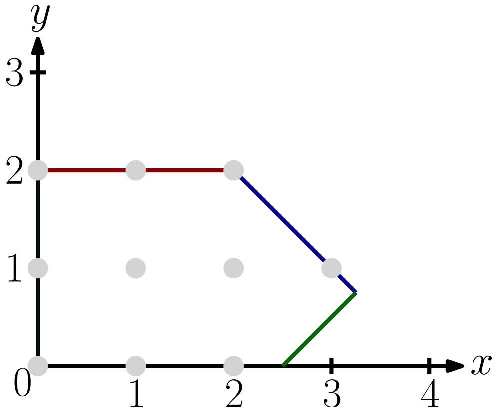
$x \in \mathbb{N}$

$y \in \mathbb{N}$

$y \leq 2$

$x + y \leq 4$

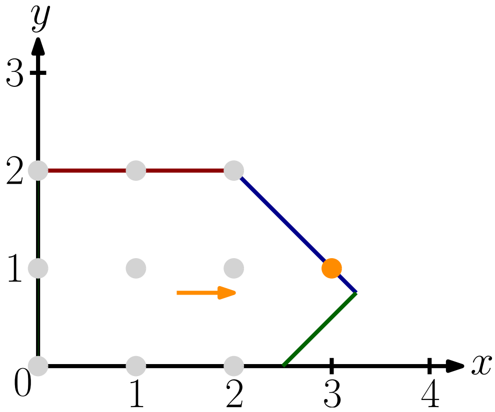
Excursion: Integer Programming



$x \in \mathbb{N}$
$y \in \mathbb{N}$

$y \leq 2$
$x + y \leq 4$
$x - y \geq 2.5$

Excursion: Integer Programming



$x \in \mathbb{N}$

$y \in \mathbb{N}$

$y \leq 2$

$x + y \leq 4$

$x - y \geq 2.5$

$\max x$

In many cases: NP-hard!

Why bother with (Mixed-)Integer Programming?

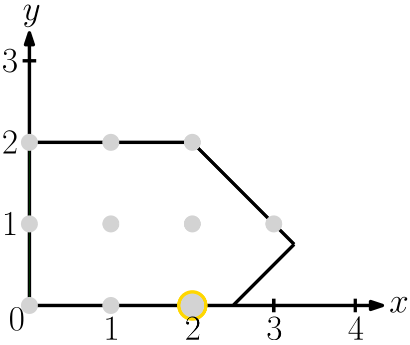
In general:

- Easy to formulate all kinds of optimization problems
- Highly optimized solvers which can even solve large problems

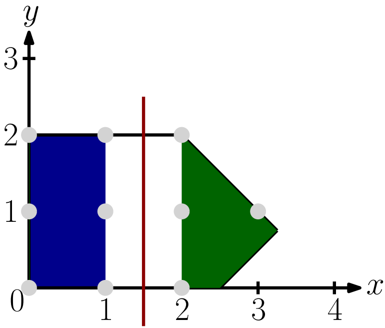
Why we bother:

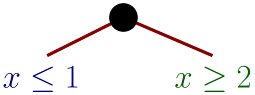
- Important as baseline for polynomial time heuristics
- Allows trading off runtime with solution quality

Excursion: Solving MIPs via Branch-and-Bound

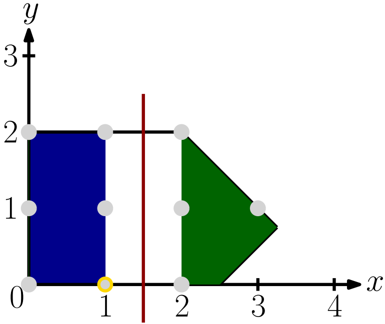


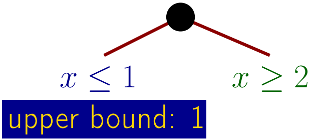
Excursion: Solving MIPs via Branch-and-Bound



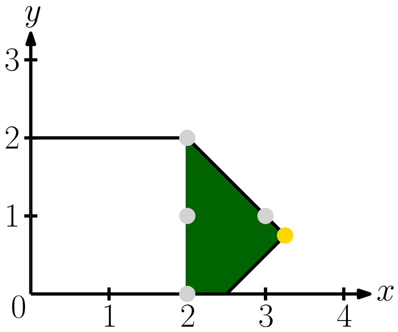


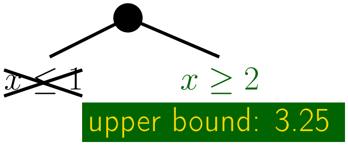
Excursion: Solving MIPs via Branch-and-Bound



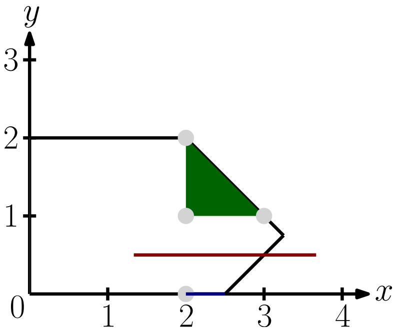


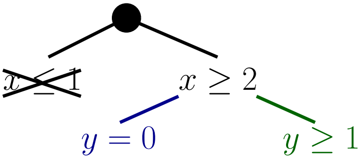
Excursion: Solving MIPs via Branch-and-Bound



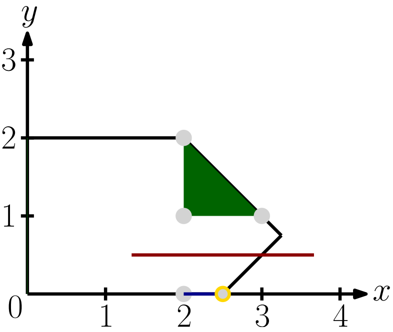


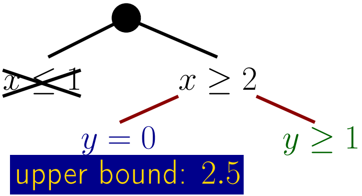
Excursion: Solving MIPs via Branch-and-Bound



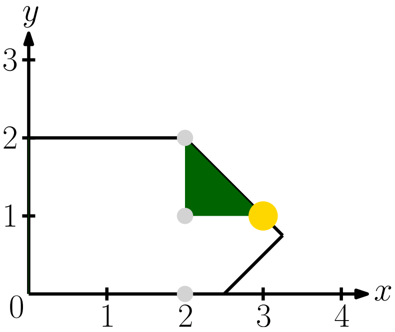


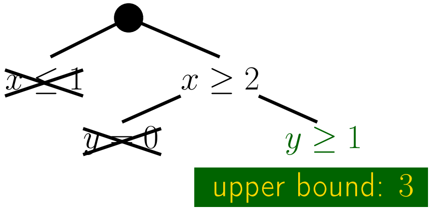
Excursion: Solving MIPs via Branch-and-Bound



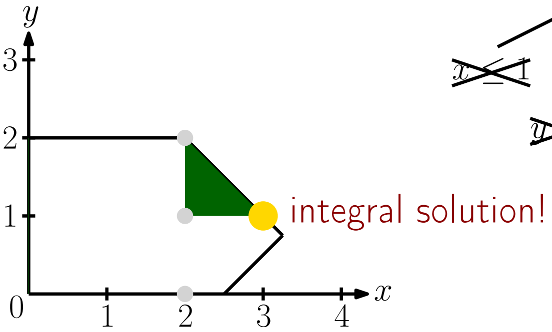


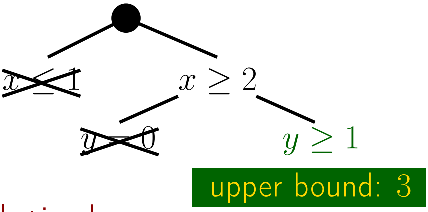
Excursion: Solving MIPs via Branch-and-Bound





Excursion: Solving MIPs via Branch-and-Bound





Branch-and-Bound Summary

Branching Divide the solution space by considering (disjoint) subspaces.

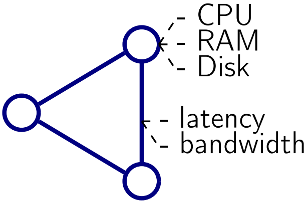
Bounding Compute linear relaxations for subspaces, allowing to cut-off nodes in the search tree.

Dual bounds are computed on-the-fly, allowing to qualify solution quality.

Part I: Scheduling Network Tasks

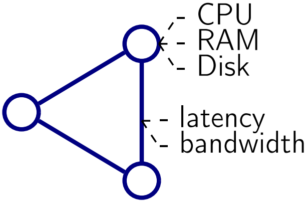
The Virtual Network Embedding Problem (VNEP)

Physical Network

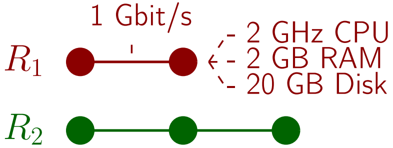


The Virtual Network Embedding Problem (VNEP)

Physical Network

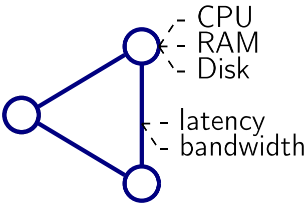


Virtual Network Requests

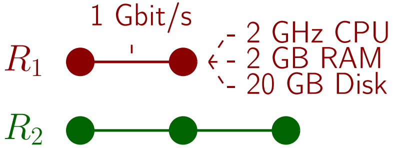


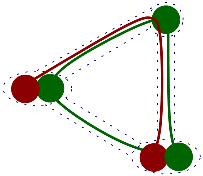
The Virtual Network Embedding Problem (VNEP)

Physical Network



Virtual Network Requests





Embedding

- map virtual onto substrate nodes
- map virtual links onto substrate paths
- obeying the substrate's capacities

Facets of the VNEP

Setting

(De)centralized

Multi-Provider

Reliability

Reconfigurations

Objectives

Access Control

Load Balancing

Energy Savings

Algorithms

Exact

Heuristic

Related Work

TABLE III
TAXONOMY OF CONCISE VNE APPROACHES

Category	Reference	Optimization	Coordination	Contribution
C/S/C	[26] Inführ and Raidl (2011)	Exact	One Stage	Provides delay, location and routing constraints
	[59] Liu et al. (2011)	Exact	One Stage	Exact VNE based on correspondence matrices
	[60] Trinh et al. (2011)	Exact	One Stage	Exact VNE problem with SLA QoS guarantees
	[61] Pages et al. (2012)	Exact/Metaheuristic	One Stage	Introduces the VNE for optical networks
	[44] Lischka and Karl (2009)	Heuristic	One Stage	Provides one stage VNE. Based on SID
	[62] Di et al. (2010)	Heuristic	One Stage	Improvement of the approach in [44]
	[63] Ghazar and Samaan (2011)	Heuristic	One Stage	Introduces hierarchical management of the SN
	[64], [65] Yun et al. (2011-2012)	Heuristic	One Stage	First VNE approach in wireless multihop networks. Introduces metrics and feasibility measures for wireless VNE
	[56] Chen et al. (2012)	Heuristic	One Stage	Reduces resource fragmentation
	[66] Yu et al. (2012)	Heuristic	One Stage	One step VNE that increases coordination
	[67] Liu et al. (2011)	Heuristic	Two Stages	Improves coordination based on nodes proximity
	[24], [25] Sheng et al. (2011-2012)	Heuristic	Two Stages	Opportunistic resource sharing to deal with load fluctuation
	[68] Li et al. (2012)	Heuristic	Two Stages	Topology awareness to enforce VNE coordination
	[69] Lu and Turner (2006)	Heuristic	Uncoordinated	Embedding in specific backbone-star VN topologies
	[32] Yu et al.* (2008)	Heuristic	Uncoordinated	Utilizes the KSP algorithm [33] for VLiM
	[70] Razzaq and Siraj (2010)	Heuristic	Uncoordinated	Different K values in KSP based VLiM
	[71] Razzaq et al. (2011)	Heuristic	Uncoordinated	Investigates the VNE impact of bottlenecked nodes
	[72] Nogueira et al. (2011)	Heuristic	Uncoordinated	VNE considering SN resources heterogeneity
	[73] Leivadeas et al.* (2011)	Heuristic	Uncoordinated	Introduces VNE for wireless network testbeds
	[22], [57] Botero et al. (2011-2013)	Heuristic	Uncoordinated	Introduces hidden hop constraints
	[74] Zhu and Ammar* (2006)	Heuristic	Uncoordinated	Provides a balanced link and node stress in the SN
	[51] Fajjari et al. (2011)	Metaheuristic	One Stage	Max-Min Ant Colony metaheuristic VNE approach
	[75] Cheng et al. (2012)	Metaheuristic	One Stage	Accelerates convergence of PSO VNE metaheuristic with topology aware node ranking [37]
	[76] Zhang et al. (2012)	Heuristic	Uncoordinated	Maps one virtual node in several substrate nodes
	[77] Di et al. (2012)	Heuristic	One Stage	Coordinated VNE reducing the number of backtracks by carefully chosing the first virtual node to map
	[78] Abedifar and Eshghi (2012)	Heuristic	Uncoordinated	Introduces VNE in the optical domain trying to minimize the number of λs per link
	[79] Aris Leivadeas et al. (2012)	Heuristic	Coordinated	Considers importance of virtual nodes for embedding
	[80] Tae-Ho Lee et al. (2012)	Heuristic	InterInP	clustering of virtual networks in multi-provider environment
C/D/C	[19] Fajjari et al. (2011)	Heuristic	One Stage	Migration of nodes with bottlenecked adjacent links
	[81] Bienkowski et al. (2010)	Heuristic	Two Stages	Migration when service access position changes
	[74] Zhu and Ammar* (2006)	Heuristic	Uncoordinated	Reduce the cost of periodic reconfigurations
	[82] Fan and Ammar (2006)	Heuristic	Uncoordinated	Reduces the cost of VNRs reconfiguration
	[83] Cai et al. (2010)	Heuristic	Uncoordinated	Reconfiguration based on SN evolution
	[84] Shun-li and Xue-song (2011)	Heuristic	Uncoordinated	Identifies mapped virtual nodes and links with not optimal mapping and migrate them to save SN resources
	[85] Sun et al. (2012)	Heuristic	Uncoordinated	Introduces the VNE problem for evolving VNRs
D/S/C	[86], [87] Houidi et al. (2010)	Heuristic	Uncoordinated	First distributed approach to solve VNE. Proposes a VNE protocol to manage the communication among substrate nodes
	[88] Xin et al. (2011)	Heuristic	InterInP	Introduces the InterInP VNE for networked clouds
	[89] Lv et al. (2011)	Heuristic	InterInP	InterInP VNE using hierarchical virtual resource organization
	[42] Houidi et al.* (2011)	Exact/Metaheuristic	InterInP	VNR is split assigning each subVN in different InPs. Provides exact and heuristic splitting approaches
	[90] Leivadeas et al.* (2012)	Heuristic	InterInP	Graph partitioning InterInP VNE using a heuristic integrating a min k-cut algorithm followed by subgraph isomorphism
D/D/C	[91] Marquezan et al. (2010)	Heuristic	Uncoordinated	First distributed dynamic approach. Reorganizes the SN when VNs demands change

TABLE IV
TAXONOMY OF REDUNDANT VNE APPROACHES

Category	Reference	Optimization	Coordination	Contribution
C/S/R	[42] Houidi et al.* (2011)	Exact	One Stage	First approach providing an ILP exact solution
	[92] Zhang et al. (2011)	Exact	One Stage	Optimal resilient solution attaining an enhanced QoS mapping. Provides diversified substrate back-up paths
	[43] Botero et al. (2012)	Exact	One Stage	Introduces the energy aware VNE
	[93] Wang and Wolf (2011)	Exact	One Stage	Redefines the VNR as a traffic matrix
	[94], [95], [96] Shamsi and Brockmeyer (2007-2009)	Heuristic	One Stage	Recover link failures by providing backup paths with intermediate nodes
	[97] Koslowski et al. (2010)	Heuristic	One Stage	Introduces reliability as a service offered by the InP. Reliable VNEs based on subgraph isomorphism detection
	[98] Yu et al. (2010)	Heuristic	One Stage	Introduces failure-dependent protection with a back-up solution for each regional failure
	[99] Lv et al. (2012)	Heuristic	One Stage	Introduces looses to multicast VNE in wireless mesh networks
	[36], [27] Chowdhury et al. (2009-2011)	Heuristic	Two Stages	Coordination in VNE using multi-path for VLiM
	[31] Rahman et al. (2010)	Heuristic	Two Stages	Upon a failure, the economic penalty is minimized by the pre-reservation of a bandwidth quota for back-up in SN links
	[53] Butt et al.* (2010)	Heuristic	Two Stages	VNE awareness of the SN bottlenecked resources
	[29] Yeow et al. (2010)	Heuristic	Two Stages	Introduces sharing among back up resources. Reduces resources allocated for redundancy
	[100] Sun et al. (2011)	Heuristic	Two Stages	Resilient VNE optimizing the embedding cost and reducing computational complexity
	[30] Yu et al. (2011)	Heuristic	Two Stages	Resilient VNE analyzing failures in substrate nodes
	[32] Yu et al.* (2008)	Heuristic	Uncoordinated	Introduces the multi-path approach for VLiM
	[101] Gao et al. (2010)	Heuristic	Uncoordinated	Improvement of the approach in [36]
	[102] Yang et al. (2010)	Heuristic	Uncoordinated	Divides the SN in regions to reduce VNE complexity
	[103] Zho et al. (2010)	Heuristic	Uncoordinated	Maps one virtual node to multiple substrate nodes
	[104] Chen et al. (2010)	Heuristic	Uncoordinated	Reactive resiliency protection approach against failures during the online VNE process. Considers just substrate link failures
	[105] Yu et al. (2011)	Heuristic	Uncoordinated	Proactive VNE approach offering protection against SN link failures for links with high stress
	[23] Sun et al. (2011)	Heuristic	Uncoordinated	Introduces stochastic BW demand to the VNE
	[55] Lu et al. (2011)	Heuristic	Uncoordinated	Introduces load balancing in links
	[28] Guo et al. (2011)	Heuristic	Uncoordinated	Proactive resilient VLiM approach sharing back-up paths
	[37] Cheng et al. (2011)	Metaheuristic	Two Stages	Introduces topology-awareness in VNE
	[106] Sheng et al. (2011)	Metaheuristic	Two Stages	Embedding time depends on VNR lifetime. Uses simulated annealing metaheuristic
	[52] Zhang et al. (2012)	Metaheuristic	Two Stages	Introduces particle swarm optimizaton (PSO) metaheuristic
	[107] Sun et al. (2012)	Metaheuristic	Two Stages	Introduces VNE in multi-datacenter environments
	[108] Lv et al. (2012)	Metaheuristic	Uncoordinated	Introduces VNE in wireless mesh networks
	[90] Leivadeas et al.* (2012)	Heuristic	Two Stages	Uses the approach in [27] to solve the VNE for an arbitrary pool of heterogeneous resources
	[54] Masti and Raghavan (2012)	Heuristic	Two Stages	VNE considering the residual capacity of the substrate links
	[109] Zhang et al. (2012)	Exact/Heuristic	One Stage	Recover link failures providing disjoint SN backup paths
C/D/R	[53] Butt et al.* (2010)	Heuristic	Two Stages	Reactive reconfiguration of virtual links and nodes causing rejection to less critical SN regions
	[32] Yu et al.* (2010)	Heuristic	Uncoordinated	Reconfigure the embedding by changing the splitting ratio in the multipath VLiM solution
	[110] Schaffrath et al. (2010)	Exact	One Stage	ILP-based VNE. Dynamically reconfigures existing mappings
	[111] Chen et al. (2011)	Heuristic	Two Stages	Periodic reconfiguration of SN nodes with high utilization
D/S/R	[38] Chowdhury et al. (2010)	Heuristic	InterInP	First InterInP VNE proposal. Mediates between InP and SP interests. VNR is split across InPs and embedded locally
D/D/R	[112] Houidi et al. (2010)	Heuristic	Two Stages	Fault-tolerant VNE that acts upon node and link failures

Related Work

TABLE III
TAXONOMY OF CONCISE VNE APPROACHES

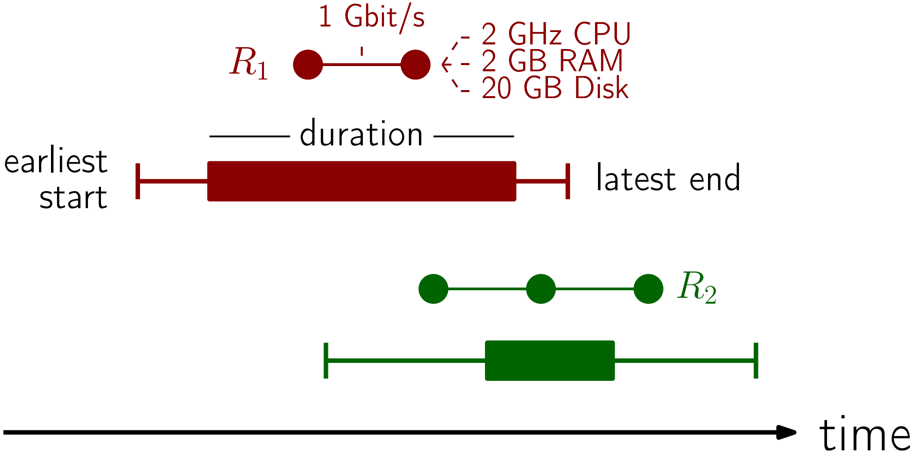
Category	Reference	Optimization	Coordination	Contribution
C/S/C	[26] Infuhr and Raidl (2011)	Exact	One Stage	Provides delay, location and routing constraints
	[59] Liu et al. (2011)	Exact	One Stage	Exact VNE based on correspondence matrices
	[60] Trinh et al. (2011)	Exact	One Stage	Exact VNE problem with SLA QoS guarantees
	[61] Pages et al. (2012)	Exact/Metaheuristic	One Stage	Introduces the VNE for optical networks
	[44] Lischka and Karl (2009)	Heuristic	One Stage	Provides one stage VNE. Based on SID
	[62] Di et al. (2010)	Heuristic	One Stage	Improvement of the approach in [44]
	[63] Ghazar and Samaan (2011)	Heuristic	One Stage	Introduces hierarchical management of the SN
	[64], [65] Yun et al. (2011-2012)	Heuristic	One Stage	First VNE approach in wireless multihop networks. Introduces metrics and feasibility measures for wireless VNE
	[56] Chen et al. (2012)	Heuristic	One Stage	Reduces resource fragmentation
	[66] Yu et al. (2012)	Heuristic	One Stage	One step VNE that increases coordination
	[67] Liu et al. (2011)	Heuristic	Two Stages	Improves coordination based on nodes proximity
	[24], [25] Sheng et al. (2011-2012)	Heuristic	Two Stages	Opportunistic resource sharing to deal with load fluctuation
	[68] Li et al. (2012)	Heuristic	Two Stages	Topology awareness to enforce VNE coordination
	[69] Lu and Turner (2006)	Heuristic	Uncoordinated	Embedding in specific backbone-star VN topologies
	[32] Yu et al.* (2008)	Heuristic	Uncoordinated	Utilizes the KSP algorithm [33] for VLiM
	[70] Razzaq and Siraj (2010)	Heuristic	Uncoordinated	Different K values in KSP based VLiM
	[71] Razzaq et al. (2011)	Heuristic	Uncoordinated	Investigates the VNE impact of bottlenecked nodes
	[72] Nogueira et al. (2011)	Heuristic	Uncoordinated	VNE considering SN resources heterogeneity
	[73] Leivadeas et al.* (2011)	Heuristic	Uncoordinated	Introduces VNE for wireless network testbeds
	[22], [illegible] Botero et al. (2011-2013)	Heuristic	Uncoordinated	Introduces hidden hop co[illegible]
	[74] Zhu and [illegible]mar* (2006)	Heuristic	Uncoordinated	Provides [illegible] balanced link and n[illegible] stress in the SN
	[illegible] Fajjari et [illegible] (20[illegible]1)	Metaheuri[illegible]	O[illegible] Stag[illegible]	Ma[illegible] Min[illegible] Co[illegible] metaheur[illegible] V[illegible] a[illegible]roac[illegible]
	[75] Cheng et [illegible] (20[illegible]2)	Metaheuristi[illegible]	O[illegible] Sta[illegible]	Ac[illegible] conv[illegible]gence [illegible] P[illegible] V[illegible] m[illegible]heuristic with topology aware node ranking [37]
	[76] Zhang et al. (2012)	Heuristic	Uncoordinated	Maps one virtual node in several substrate nodes
	[77] Di et al. (2012)	Heuristic	One Stage	Coordinated VNE reducing the number of backtracks by carefully chosing the first virtual node to map
	[78] Abedifar and Eshghi (2012)	Heuristic	Uncoordinated	Introduces VNE in the optical domain trying to minimize the number of λs per link
	[79] Aris Leivadeas et al. (2012)	Heuristic	Coordinated	Considers importance of virtual nodes for embedding
	[80] Tae-Ho Lee et al. (2012)	Heuristic	InterInP	clustering of virtual networks in multi-provider environment
C/D/C	[19] Fajjari et al. (2011)	Heuristic	One Stage	Migration of nodes with bottlenecked adjacent links
	[81] Bienkowski et al. (2010)	Heuristic	Two Stages	Migration when service access position changes
	[74] Zhu and Ammar* (2006)	Heuristic	Uncoordinated	Reduce the cost of periodic reconfigurations
	[82] Fan and Ammar (2006)	Heuristic	Uncoordinated	Reduces the cost of VNRs reconfiguration
	[83] Cai et al. (2010)	Heuristic	Uncoordinated	Reconfiguration based on SN evolution
	[84] Shun-li and Xue-song (2011)	Heuristic	Uncoordinated	Identifies mapped virtual nodes and links with not optimal mapping and migrate them to save SN resources
	[85] Sun et al. (2012)	Heuristic	Uncoordinated	Introduces the VNE problem for evolving VNRs
D/S/C	[86], [87] Houidi et al. (2010)	Heuristic	Uncoordinated	First distributed approach to solve VNE. Proposes a VNE protocol to manage the communication among substrate nodes
	[88] Xin et al. (2011)	Heuristic	InterInP	Introduces the InterInP VNE for networked clouds
	[89] Lv et al. (2011)	Heuristic	InterInP	InterInP VNE using hierarchical virtual resource organization
	[42] Houidi et al.* (2011)	Exact/Metaheuristic	InterInP	VNR is split assigning each subVN in different InPs. Provides exact and heuristic splitting approaches
	[90] Leivadeas et al.* (2012)	Heuristic	InterInP	Graph partitioning InterInP VNE using a heuristic integrating a min k-cut algorithm followed by subgraph isomorphism
D/D/C	[91] Marquezan et al. (2010)	Heuristic	Uncoordinated	First distributed dynamic approach. Reorganizes the SN when VNs demands change

TABLE IV
TAXONOMY OF REDUNDANT VNE APPROACHES

Category	Reference	Optimization	Coordination	Contribution
C/S/R	[42] Houidi et al.* (2011)	Exact	One Stage	First approach providing an ILP exact solution
	[92] Zhang et al. (2011)	Exact	One Stage	Optimal resilient solution attaining an enhanced QoS mapping. Provides diversified substrate back-up paths
	[43] Botero et al. (2012)	Exact	One Stage	Introduces the energy aware VNE
	[93] Wang and Wolf (2011)	Exact	One Stage	Redefines the VNR as a traffic matrix
	[94], [95], [96] Shamsi and Brockmeyer (2007-2009)	Heuristic	One Stage	Recover link failures by providing backup paths with intermediate nodes
	[97] Koslovski et al. (2010)	Heuristic	One Stage	Introduces reliabilty as a service offered by the InP. Reliable VNEs based on subgraph isomorphism detection
	[98] Yu et al. (2010)	Heuristic	One Stage	Introduces failure-dependent protection with a back-up solution for each regional failure
	[99] Lv et al. (2012)	Heuristic	One Stage	Introduces looses to multicast VNE in wireless mesh networks
	[36], [27] Chowdhury et al. (2009-2011)	Heuristic	Two Stages	Coordination in VNE using multi-path for VLiM
	[31] Rahman et al. (2010)	Heuristic	Two Stages	Upon a failure, the economic penalty is minimized by the pre-reservation of a bandwidth quota for back-up in SN links
	[53] Butt et al.* (2010)	Heuristic	Two Stages	VNE awareness of the SN bottlenecked resources
	[29] Yeow et al. (2010)	Heuristic	Two Stages	Introduces sharing among back up resources. Reduces resources allocated for redundancy
	[100] Sun et al. (2011)	Heuristic	Two Stages	Resilient VNE optimizing the embedding cost and reducing computational complexity
	[30] Yu et al. (2011)	Heuristic	Two Stages	Resilient VNE analyzing failures in substrate nodes
	[32] Yu et al.* (2008)	Heuristic	Uncoordinated	Introduces the multi-path approach for VLiM
	[101] Gao et al. ([illegible]10)	Heuristic	Uncoordinated	Improvement of the [illegible]pproach in [16]
	[102] Yang et al. [illegible]010)	[illegible]	[illegible]	[illegible] the S[illegible] r[illegible]ons [illegible] re[illegible] V[illegible]E complexity
	[103] Zho et al. ([illegible]10)	[illegible]	U[illegible]ordinated	[illegible]ps o[illegible] n[illegible] to [illegible]ltip[illegible] subst[illegible]odes
	[104] Chen et al. ([illegible]010)	[illegible]	[illegible]	Reactive resiliency protection approach ag[illegible]st failures during the online VNE process. Considers just[illegible]strate link failures
	[105] Yu et al. (2011)	Heuristic	Uncoordinated	Proactive VNE approach offering protection against SN link failures for links with high stress
	[23] Sun et al. (2011)	Heuristic	Uncoordinated	Introduces stochastic BW demand to the VNE
	[55] Lu et al. (2011)	Heuristic	Uncoordinated	Introduces load balancing in links
	[28] Guo et al. (2011)	Heuristic	Uncoordinated	Proactive resilient VLiM approach sharing back-up paths
	[37] Cheng et al. (2011)	Metaheuristic	Two Stages	Introduces topology-awareness in VNE
	[106] Sheng et al. (2011)	Metaheuristic	Two Stages	Embedding time depends on VNR lifetime. Uses simulated annealing metaheuristic
	[52] Zhang et al. (2012)	Metaheuristic	Two Stages	Introduces particle swarm optimizaton (PSO) metaheuristic
	[107] Sun et al. (2012)	Metaheuristic	Two Stages	Introduces VNE in multi-datacenter environments
	[108] Lv et al. (2012)	Metaheuristic	Uncoordinated	Introduces VNE in wireless mesh networks
	[90] Leivadeas et al.* (2012)	Heuristic	Two Stages	Uses the approach in [27] to solve the VNE for an arbitrary pool of heterogeneous resources
	[54] Masti and Raghavan (2012)	Heuristic	Two Stages	VNE considering the residual capacity of the substrate links
	[109] Zhang et al. (2012)	Exact/Heuristic	One Stage	Recover link failures providing disjoint SN backup paths
C/D/R	[53] Butt et al.* (2010)	Heuristic	Two Stages	Reactive reconfiguration of virtual links and nodes causing rejection to less critical SN regions
	[32] Yu et al.* (2010)	Heuristic	Uncoordinated	Reconfigure the embedding by changing the splitting ratio in the multipath VLiM solution
	[110] Schaffrath et al. (2010)	Exact	One Stage	ILP-based VNE. Dynamically reconfigures existing mappings
	[111] Chen et al. (2011)	Heuristic	Two Stages	Periodic reconfiguration of SN nodes with high utilization
D/S/R	[38] Chowdhury et al. (2010)	Heuristic	InterInP	First InterInP VNE proposal. Mediates between InP and SP interests. VNR is split across InPs and embedded locally
D/D/R	[112] Houidi et al. (2010)	Heuristic	Two Stages	Fault-tolerant VNE that acts upon node and link failures

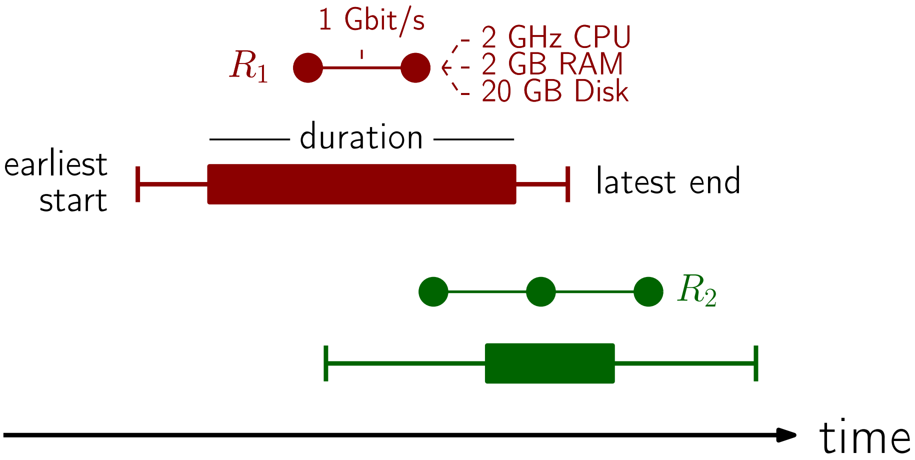
Our Contribution: Temporality

Our Model


1 Gbit/s
R_1
2 GHz CPU
2 GB RAM
20 GB Disk
duration
earliest start
latest end
R_2
time

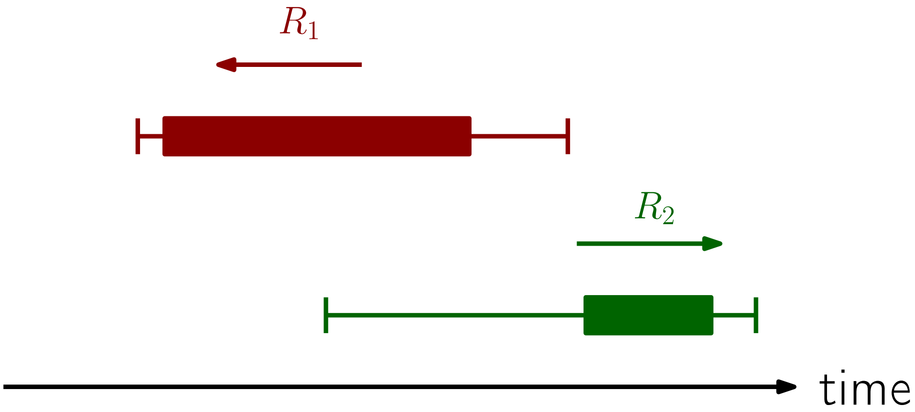
Our Model

Offline scenario

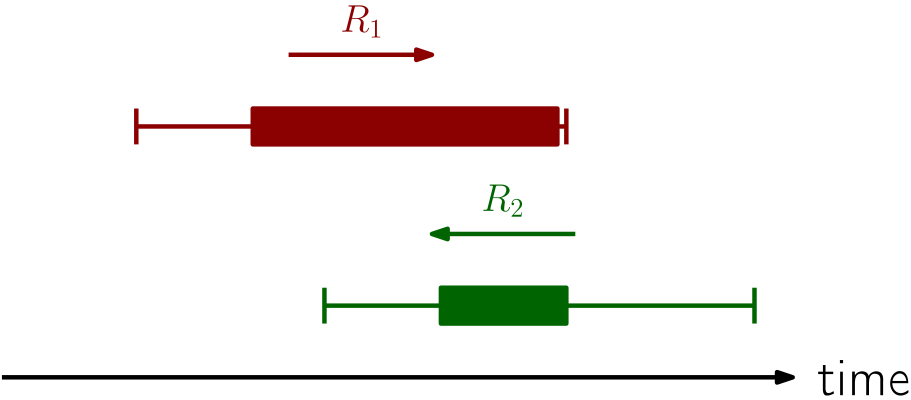


Motivation #1: Business

Provider Incentives: Minimizing Load



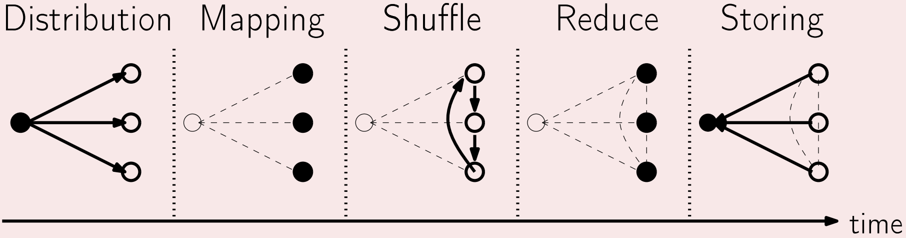
Provider Incentives: Maximizing Utilization by Collocation



Motivation #2: Modeling Opportunities

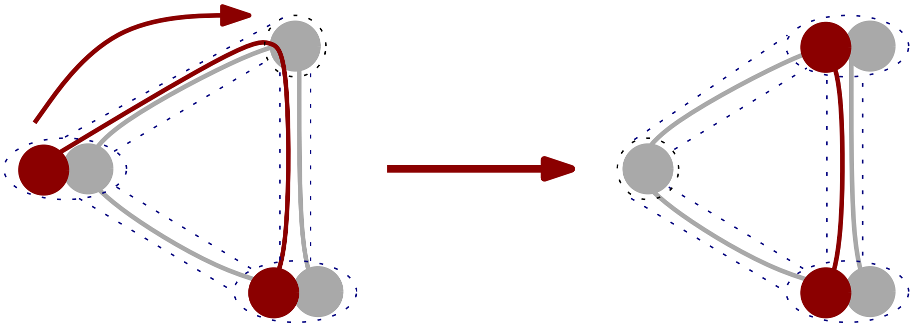
Modeling Opportunities: Evolution of VNets

MapReduce [6]

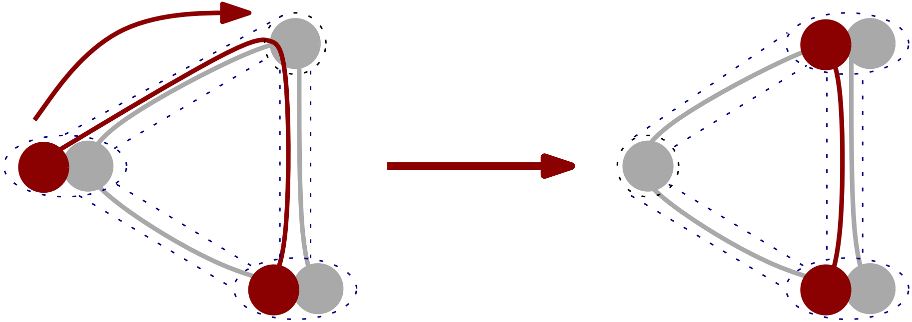


Reservation of maximal allocations over the whole time?

Modeling Opportunities: Migrations

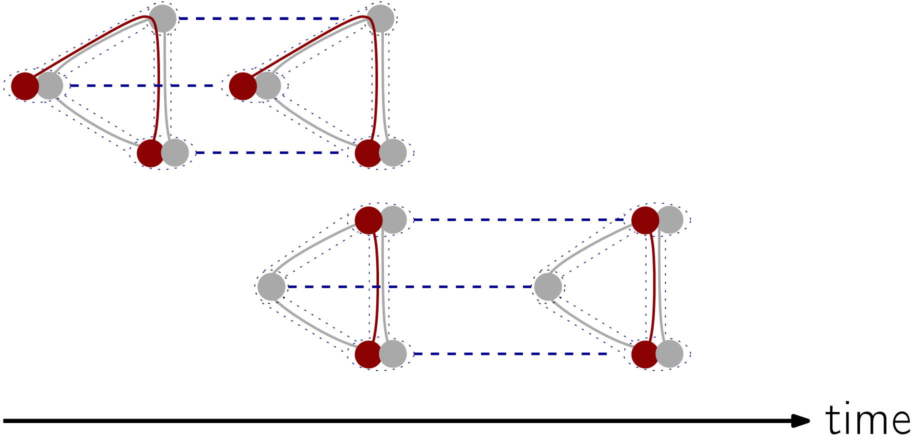


Modeling Opportunities: Migrations

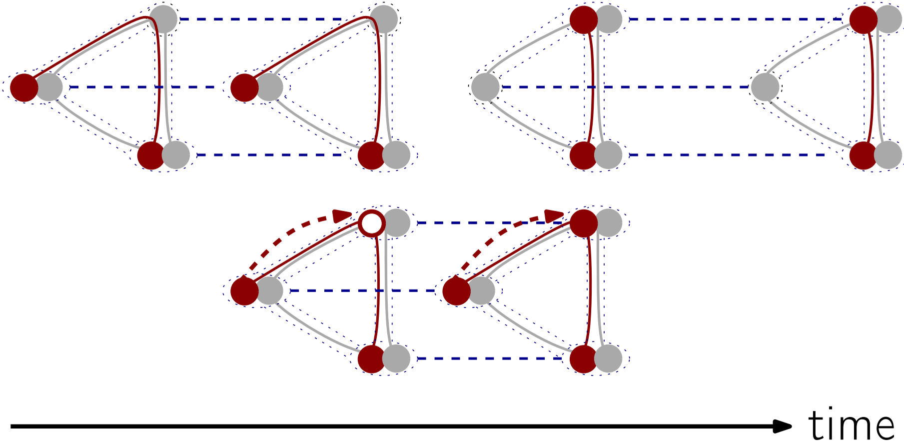


In previous work instantaneous operation!

Modeling Opportunities: Migrations

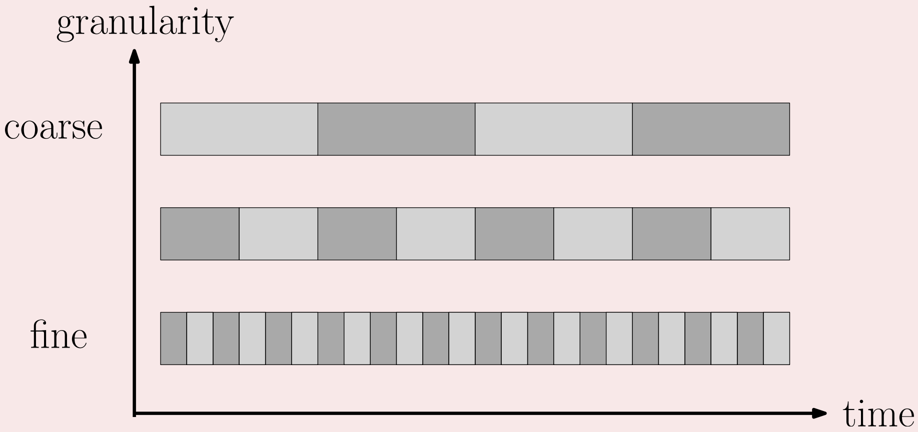


Modeling Opportunities: Fine-grained Migrations



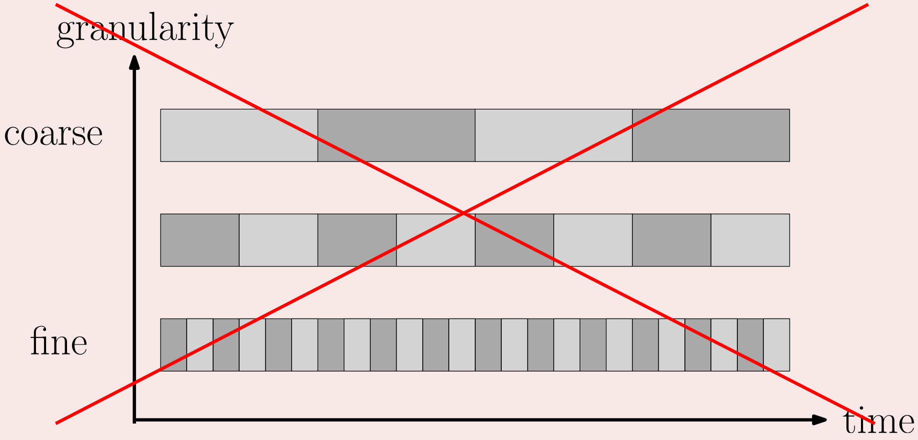
Important Decision: Continuous-Time Model!

Discretization



Important Decision: Continuous-Time Model!

No Discretization!



Problem Statement

Temporal Virtual Network Embedding Problem (TVNEP)

Access Control	Decide which of the requests to embed.
Resource Mapping	Find embeddings for requests.
Scheduling	Find appropriate start and end times for requests.
Feasibility	For *each* point in time the substrate's capacity is not exceeded.

Local Embedding

Classic VNEP Task

Access Control	Decide which of the requests to embed.
Resource Mapping	Find embeddings for requests.

Mapping can be easily done with Mixed-Integer Programming.
Not explained here.

Local Embedding

Assume that for each request $R \in \mathcal{R}$ allocations on each node and each link are given.

$$\mathit{alloc}(\mathrm{R}) : \mathit{Nodes} \cup \mathit{Links} \to \mathbb{R}_{\geq 0}$$

Overview

Overview

Contributions

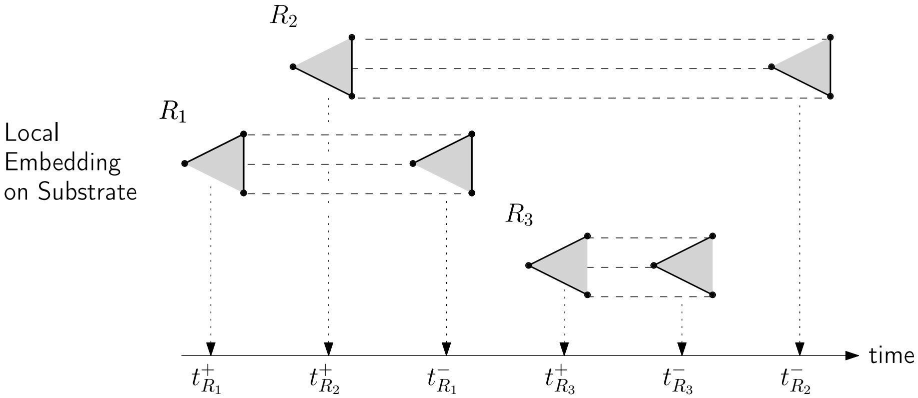
1. Continuous-time Mixed-Integer Programming formulations for TVNEP
2. cΣ-Model utilizes state-space and symmetry reductions to render solving TVNEP (computationally) feasible
3. Greedy polynomial time heuristic which is based on cΣ-Model
4. Initial computational evaluation

Joint work with Stefan Schmid, Anja Feldmann: IEEE IPDPS 2014.

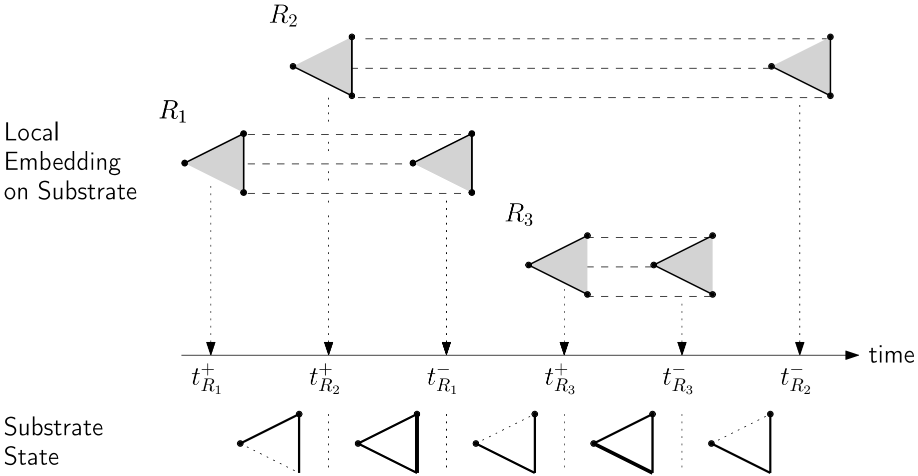
Modeling Continuous-Time: Checking Feasibility

Assume for now:
Local embeddings and start / end times are fixed.

Modeling Continuous-Time: Checking Feasibility

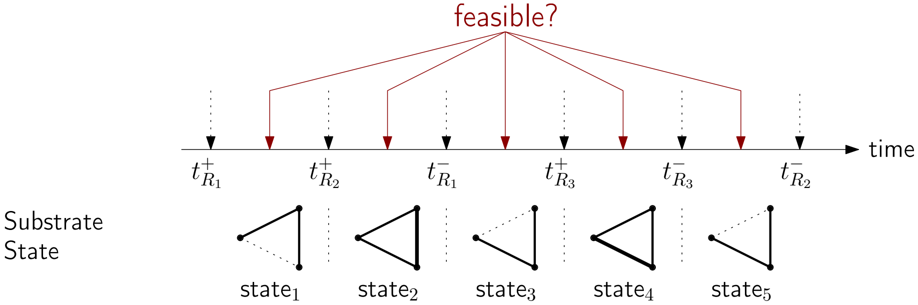


Modeling Continuous-Time: Checking Feasibility



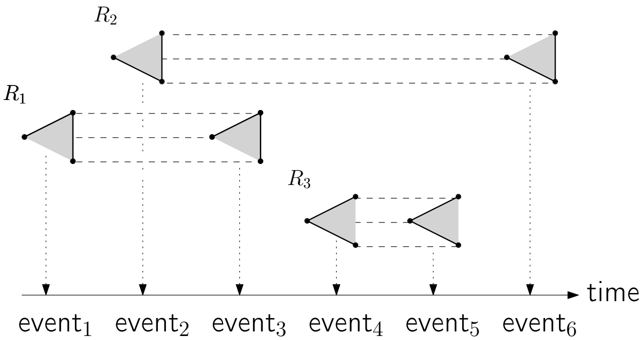
Modeling Continuous-Time: Checking Feasibility

Check the feasibility of the $2 \cdot |\mathcal{R}| - 1$ states.



Abstract Event Model

Modeling Continuous-Time: Abstract Event Model



Mapping Variables

$$\mathcal{E} = \{\mathbf{e}_1, \dots, \mathbf{e}_{2\cdot|\mathcal{R}|}\}$$

$$\forall \mathsf{R} \in \mathcal{R}.\ \chi^+_{\mathsf{R}} \colon \mathcal{E} \to \mathbb{B}$$

$$\forall \mathsf{R} \in \mathcal{R}.\ \chi^-_{\mathsf{R}} \colon \mathcal{E} \to \mathbb{B}$$

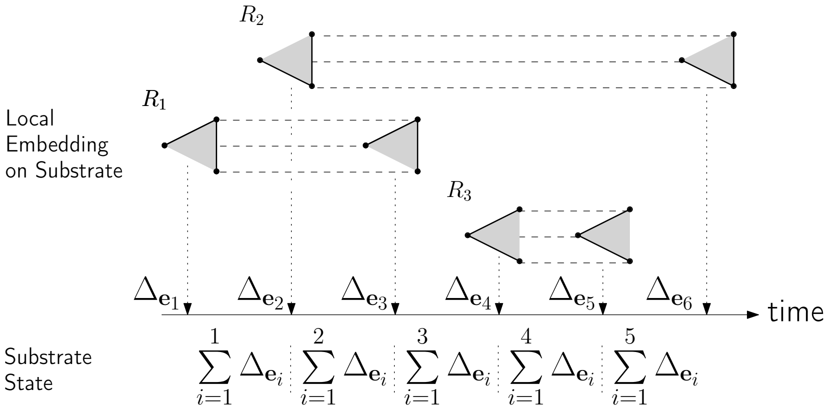
Bijective Mapping

$$\forall \mathsf{R} \in \mathcal{R}.\quad \sum_{\mathbf{e}_i \in \mathcal{E}} \chi^+_{\mathsf{R}}(\mathbf{e}_i) = 1 \wedge \sum_{\mathbf{e}_i \in \mathcal{E}} \chi^-_{\mathsf{R}}(\mathbf{e}_i) = 1$$

$$\forall \mathbf{e}_i \in \mathcal{E}.\quad \sum_{\mathsf{R} \in \mathcal{R}} \chi^+_{\mathsf{R}}(\mathbf{e}_i) + \chi^-_{\mathsf{R}}(\mathbf{e}_i) = 1$$

Δ-Model

Reconstructing States: Δ-Model



Idea

1. Compute state *changes*: $\Delta_{\mathbf{e}_i} : \mathbf{V_S} \cup \mathbf{E_S} \to \mathbb{R}$ via $\chi^+_{\mathsf{R}}(e_i), \chi^-_{\mathsf{R}}(e_i)$
2. Enforce $\sum_{j=1}^{i} \Delta_{\mathbf{e}_i} \leq \mathbf{c_S}$ for each state

Δ-Model: Computing State Changes

Conditional Assignment

$\forall \mathbf{e}_i \in \mathcal{E}.$

$$\Delta_{\mathbf{e}_i} = \begin{cases} +alloc(\mathsf{R}_1) & \text{, if } \chi^{+}_{\mathsf{R}_1}(\mathbf{e}_i) = 1 \\ -alloc(\mathsf{R}_1) & \text{, if } \chi^{-}_{\mathsf{R}_1}(\mathbf{e}_i) = 1 \\ \vdots & \\ +alloc(\mathsf{R}_k) & \text{, if } \chi^{+}_{\mathsf{R}_k}(\mathbf{e}_i) = 1 \\ -alloc(\mathsf{R}_k) & \text{, if } \chi^{-}_{\mathsf{R}_k}(\mathbf{e}_i) = 1 \end{cases}$$

Δ-Model: Computing State Changes

Conditional Assignment via Big-M Constraints

$\forall \mathsf{R} \in \mathcal{R}.\ \forall \mathbf{e}_i \in \mathcal{E}.$

$$\begin{aligned}
\Delta_{\mathbf{e}_i} &\leq +\,\mathit{alloc}(\mathsf{R}) + \mathbf{c_S}(1 - \chi^{+}_{\mathsf{R}}(\mathbf{e}_i)) \\
\Delta_{\mathbf{e}_i} &\geq +\,\mathit{alloc}(\mathsf{R}) - \mathbf{c_S}(1 - \chi^{+}_{\mathsf{R}}(\mathbf{e}_i)) \cdot 2 \\
\Delta_{\mathbf{e}_i} &\leq -\,\mathit{alloc}_V(\mathsf{R}) + \mathbf{c_S}(1 - \chi^{-}_{\mathsf{R}}(\mathbf{e}_i)) \cdot 2 \\
\Delta_{\mathbf{e}_i} &\geq -\,\mathit{alloc}_V(\mathsf{R}) - \mathbf{c_S}(1 - \chi^{-}_{\mathsf{R}}(\mathbf{e}_i))
\end{aligned}$$

Δ-Model: Computing State Changes

Big-M Assignment of Start

$\forall \mathrm{R} \in \mathcal{R}.\ \forall \mathbf{e}_i \in \mathcal{E}.$

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

$\chi^{+}_{\mathrm{R}}(\mathbf{e}_i) = 0$

$$\Delta_{\mathbf{e}_i} \leq +\, alloc(\mathrm{R}) + \mathbf{c_S}$$
$$\Delta_{\mathbf{e}_i} \geq +\, alloc(\mathrm{R}) - 2 \cdot \mathbf{c_S}$$

$\Rightarrow$

unbounded

$$\Delta_{\mathbf{e}_i}(N_s) \leq \mathbf{c_S}$$
$$\Delta_{\mathbf{e}_i}(N_s) \geq -\, \mathbf{c_S}$$

Δ-Model: Computing State Changes

Big-M Assignment of Start

$\forall \mathsf{R} \in \mathcal{R}.\ \forall \mathbf{e}_i \in \mathcal{E}.$

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

$\chi^{+}_{\mathsf{R}}(\mathbf{e}_i) = 1$

$$\Delta_{\mathbf{e}_i} \leq +\, alloc(\mathsf{R})$$
$$\Delta_{\mathbf{e}_i}) \geq +\, alloc(\mathsf{R})$$

$\Rightarrow$

equal

$$\Delta_{\mathbf{e}_i} = alloc(\mathsf{R})$$

Δ-Model Issue: LP Smearings!

LP Relaxation Example

$\chi^{+}_{\mathrm{R_j}}(\mathbf{e}_j) = 0.5$ for $j \in \{1, 2\}$:

$$-\mathbf{c_S} + alloc(\mathrm{R_j}) \leq \Delta_{\mathbf{e}_j} \leq alloc(\mathrm{R_j}) + 0.5 \cdot \mathbf{c_S}$$

Δ-Model Issue: LP Smearings!

LP Relaxation Example

$\chi^{+}_{\mathrm{R_j}}(\mathbf{e}_j) = 0.5$ for $j \in \{1, 2\}$:

$$-\mathbf{c_S} + alloc(\mathrm{R_j}) \leq \Delta_{\mathbf{e}_j} \leq alloc(\mathrm{R_j}) + 0.5 \cdot \mathbf{c_S}$$

Implications

1. $\Delta_{\mathbf{e}_j} \leq 0$ is always feasible (when $\chi^{+}_{\mathrm{R_j}}(\mathbf{e}_j) = 0.5$ for $j \in \{1, 2\}$)
2. $\Delta_{\mathbf{e}_j} < 0$ possible if $alloc(\mathrm{R_j}) < \mathbf{c_S}$

Δ-Model Issue: LP Smearings!

LP Relaxation Example

$\chi^{+}_{\mathrm{R_j}}(\mathbf{e}_j) = 0.5$ for $j \in \{1,2\}$:

$$-\mathbf{c_S} + \mathit{alloc}(\mathrm{R_j}) \leq \Delta_{\mathbf{e}_j} \leq \mathit{alloc}(\mathrm{R_j}) + 0.5 \cdot \mathbf{c_S}$$

Implications

1. $\Delta_{\mathbf{e}_j} \leq 0$ is always feasible (when $\chi^{+}_{\mathrm{R_j}}(\mathbf{e}_j) = 0.5$ for $j \in \{1,2\}$)
2. $\Delta_{\mathbf{e}_j} < 0$ possible if $\mathit{alloc}(\mathrm{R_j}) < \mathbf{c_S}$

This is really bad!

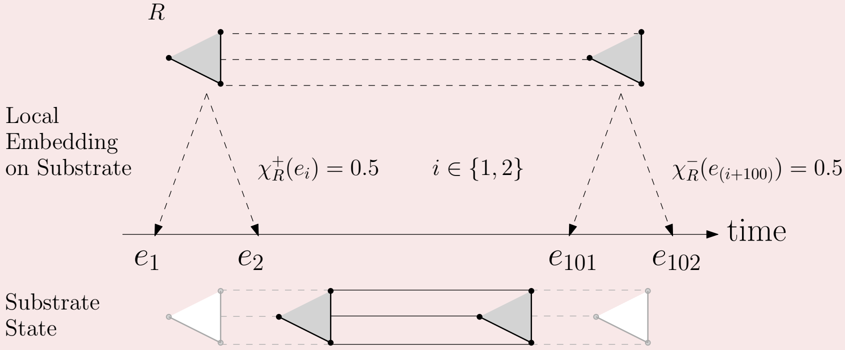
1. states do not 'materialize' well in LP relaxations: allocations will *never* be accounted for in the substrate's state
2. bounding is unable to reduce search space

Σ-Model

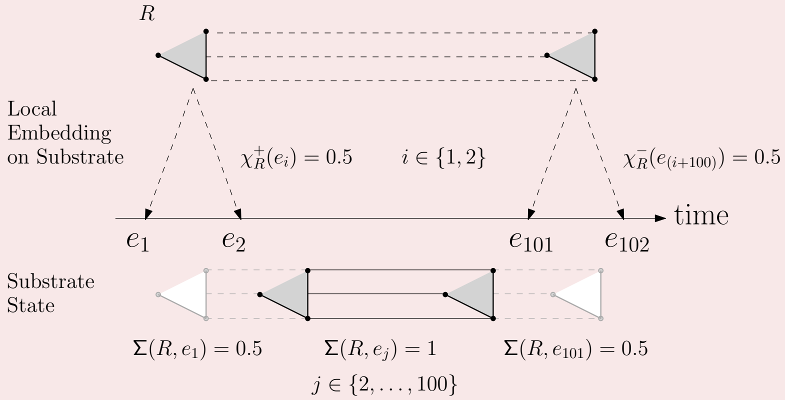
Σ-Model: Intuition

Requirement

Resource allocations must materialize in the substrate's state.

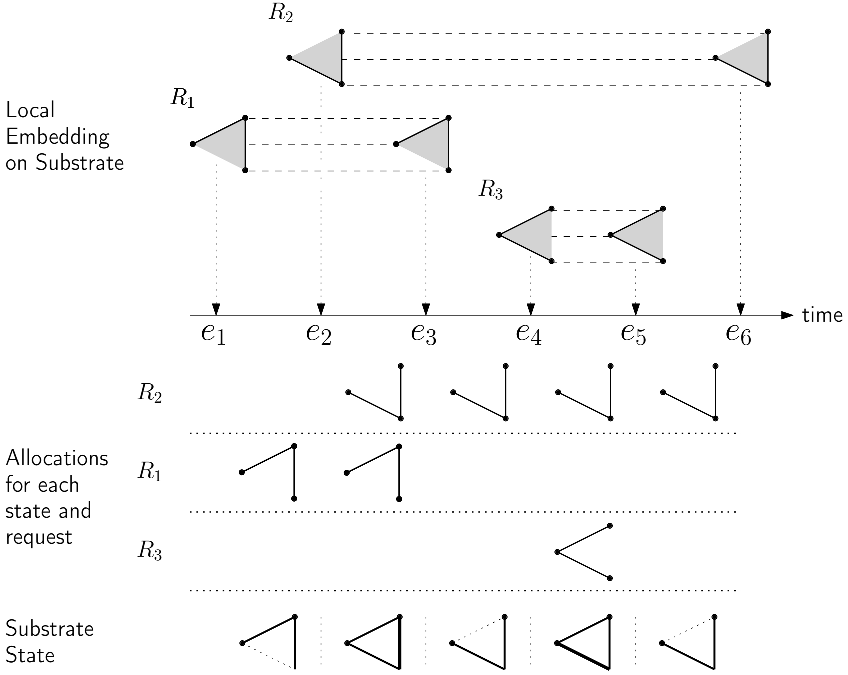


Σ-Model: Intuition



$\forall \mathsf{R} \in \mathcal{R}. \forall \mathbf{e}_i \in \mathcal{E}.$

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



Local Embedding on Substrate
R_2
R_1
R_3
time
e_1 e_2 e_3 e_4 e_5 e_6
Allocations for each state and request
R_2
R_1
R_3
Substrate State

cΣ-Model

cΣ-Model: Outline

Computational Trade-Off

- The Σ-Model is provably stronger than the Δ-Model.
- However, the Σ-Model uses (approximately) $2 \cdot |\mathcal{R}|$ more variables!

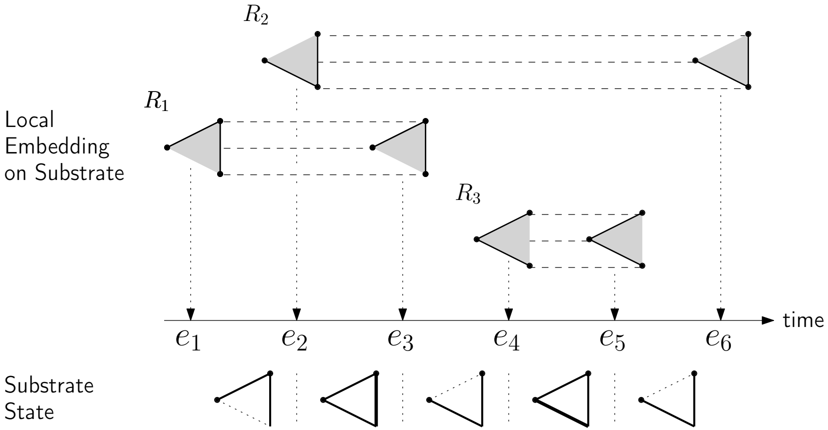
Σ-Model can be strengthened: cΣ-Model

Compactification Consider only *partial* event order. Yields *state-space* and *symmetry reductions*.

User cuts Use temporal information to reduce *state-space* and strengthen formulation.

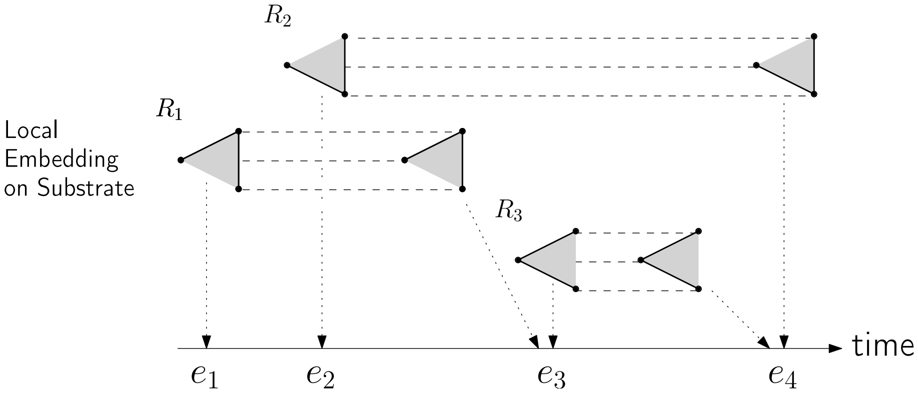
cΣ Optimization: State Compactification

cΣ-Model: State Compactification



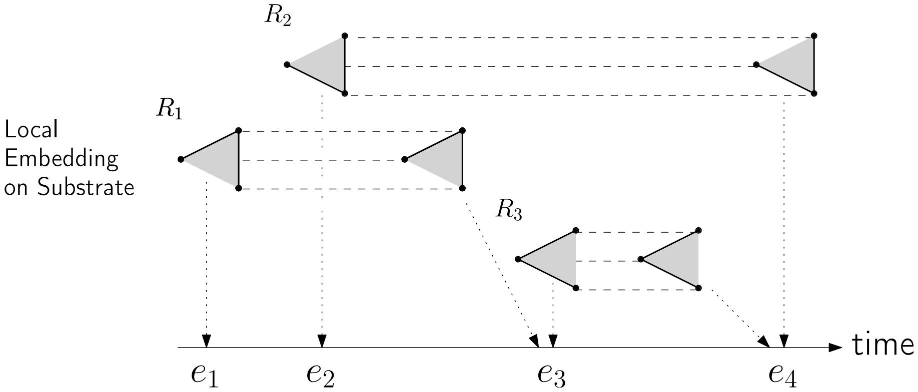
We only need to check feasibility after a request's start!

cΣ-Model: State Compactification



- consider only $|\mathcal{R}| + 1$ event points
- injective mapping of request starts onto first $|\mathcal{R}|$ event points
- mapping of request R's end onto event $\mathbf{e}_j$:
  R ends after $\mathbf{e}_{j-1}$ and before $\mathbf{e}_j$

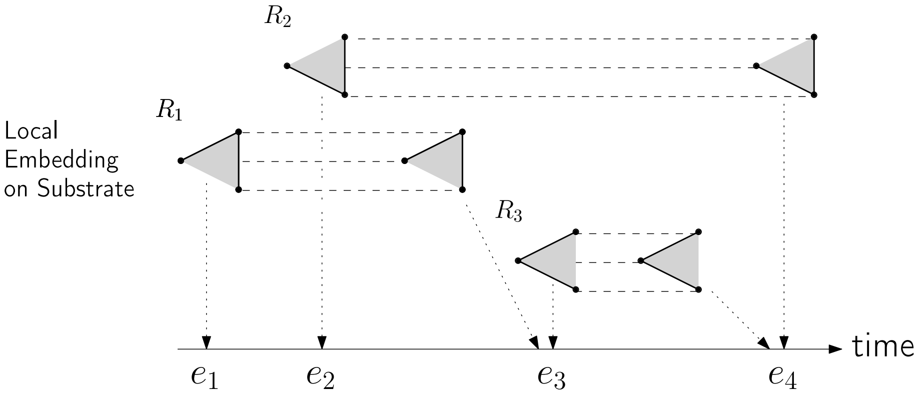
cΣ-Model: State Compactification



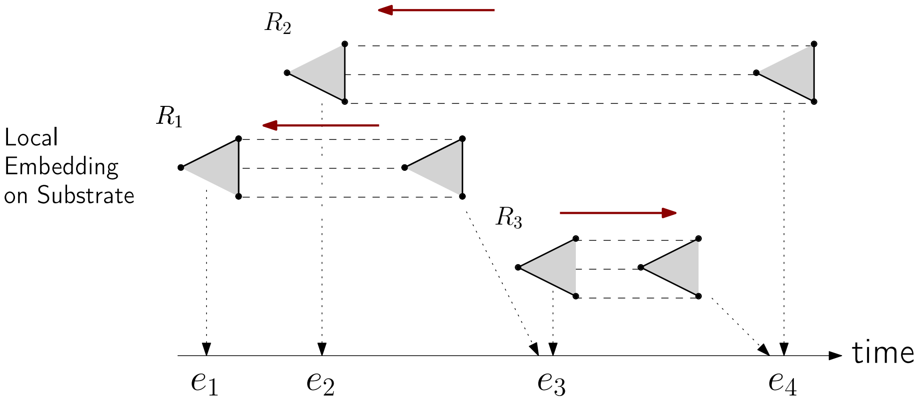
State-space reduction

Number of states is halved $\Rightarrow$ number of variables is halved.

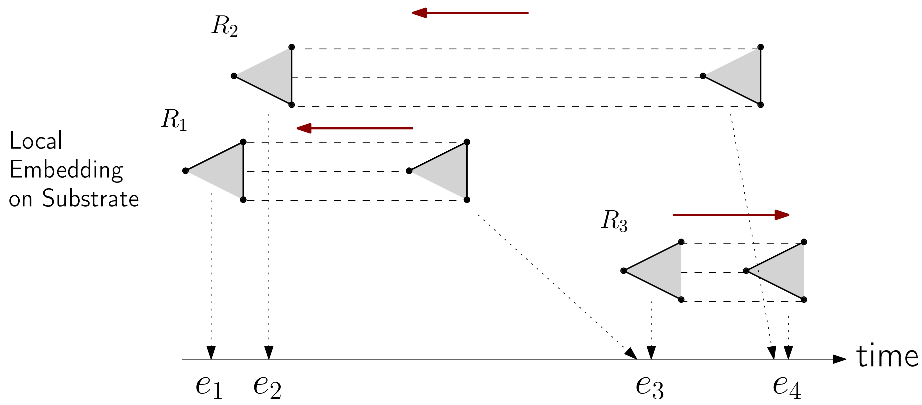
cΣ-Model: State Compactification is Symmetry Reduction



cΣ-Model: State Compactification is Symmetry Reduction

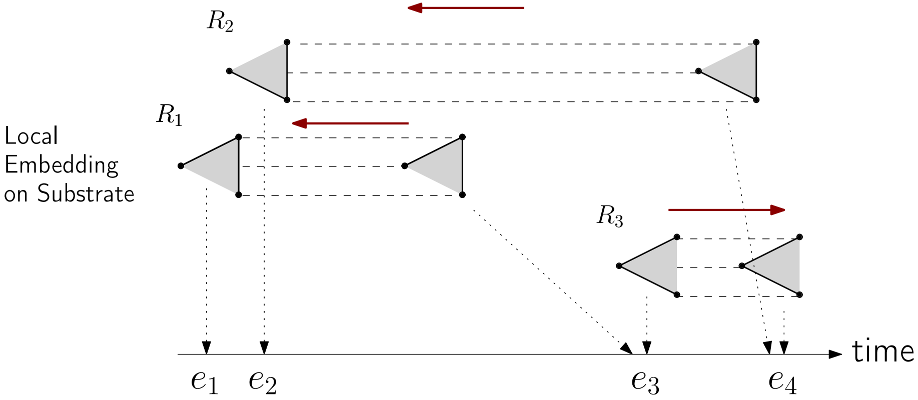


cΣ-Model: State Compactification is Symmetry Reduction



Same order as before!

cΣ-Model: State Compactification is Symmetry Reduction



Theorem

Compactification is symmetry reduction.

Greedy Heuristic $c\Sigma_A^G$

Greedy Heuristic $c\Sigma_A^G$

Outline

1. Order requests according to their earliest start time.
2. Iteratively try to embed requests as soon as possible using $c\Sigma$-Model
    1. If the request was embedded: fix start and end time.

Greedy Heuristic $c\Sigma_A^G$

Outline

1. Order requests according to their earliest start time.
2. Iteratively try to embed requests as soon as possible using $c\Sigma$-Model
    1. If the request was embedded: fix start and end time.

Theorem: $c\Sigma_A^G$ is polynomial-time algorithm

There are maximally $|\mathcal{R}|$ many possible orderings to consider.

Important

All link allocations are re-computed in each iteration.

Computational Evaluation

Scenario: One day workload

- 20 requests (star-graphs) are to be embedded on 4×5 grid
- Expected inter-arrival time of one hour [Poisson]
- Expected duration of 3.5 hours [Weibull: heavy-tailed]
- Node-mappings are fixed to concentrate on temporal aspects
- Link-mappings are not fixed
- Increasing temporal flexibility: $0, 30, 60, \ldots, 300$ minutes.

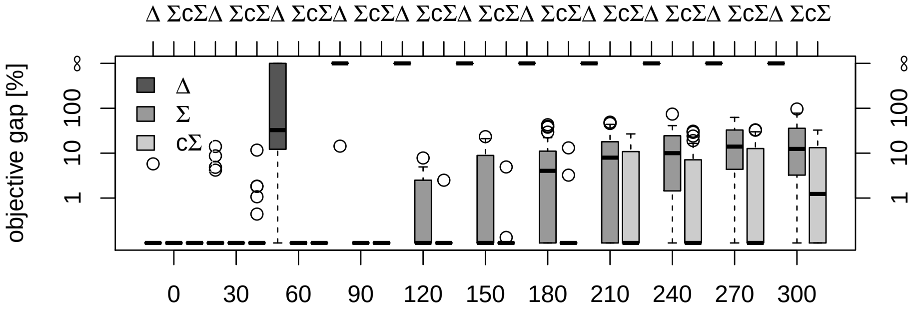
Computational Setup

- 24 independently generated scenarios
- Limited runtime of one hour for MIPs [Gurobi]

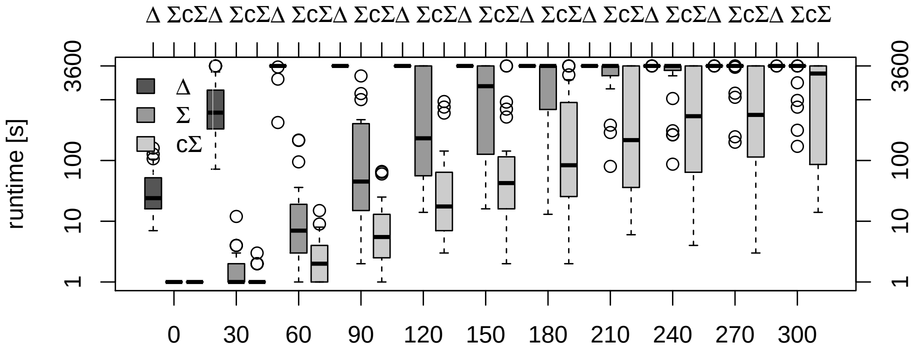
Task: Maximize revenue $\propto$ load $\cdot$ duration

1. Decide which requests to embed (access control).
2. Find time-invariant embedding (routing of data).
3. Decide when to embed the requests.

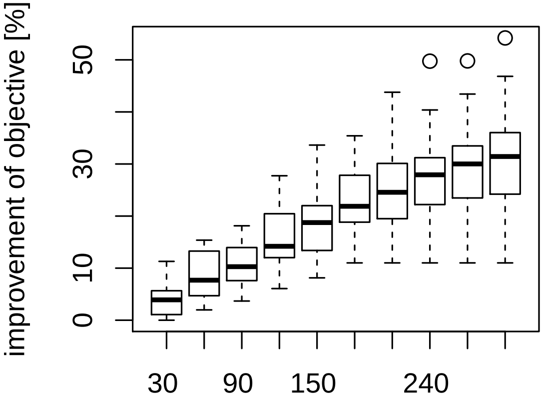
Objective Gap: MIP Formulations



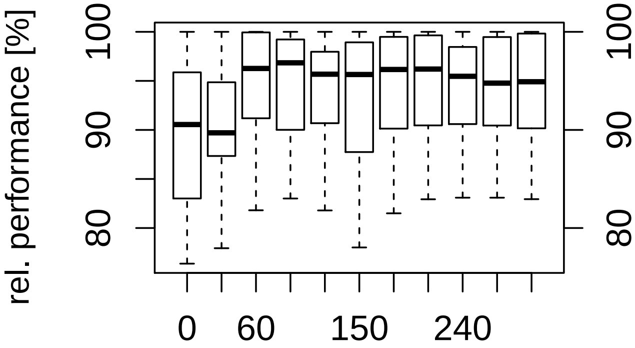
Runtime: MIP Formulations



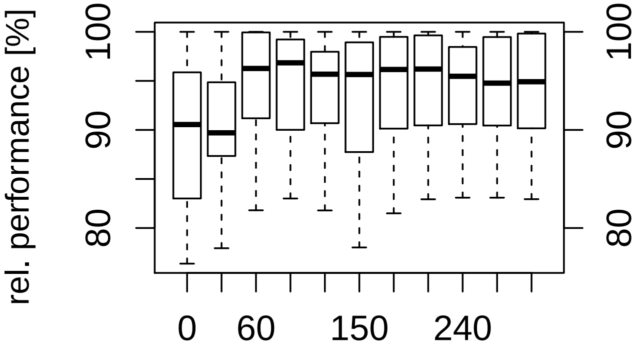
Benefit of Flexibility



Performance of $c\Sigma_A^G$



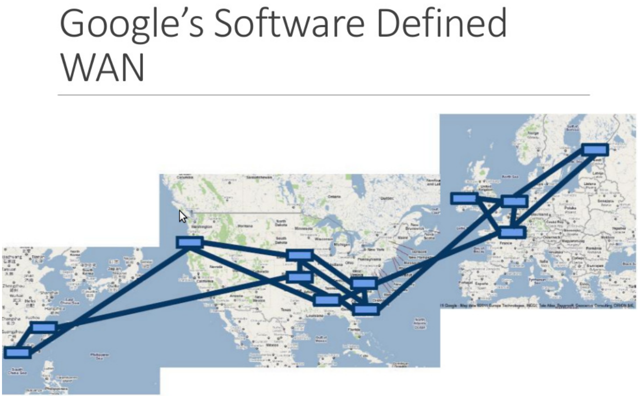
Performance of $c\Sigma_A^G$



Fast: runtime of few seconds.

Conclusion: Modeling Opportunities / Discussion

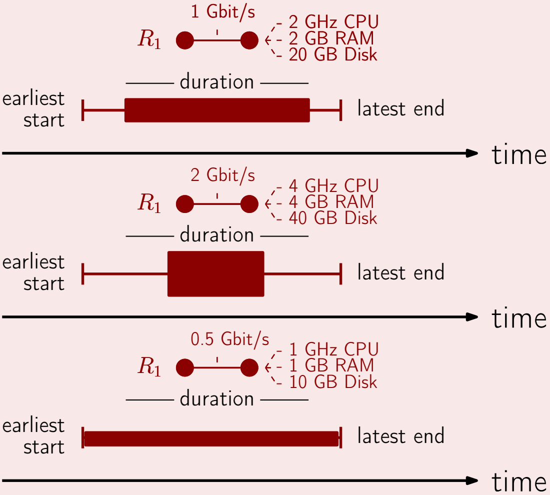
A possible application: Google B4



Goal: Utilize close to 100% of available bandwidth.

A possible application: Google B4

Intersting Extension: Delay-tolerant requests



Important Notice

We can schedule *anything

- The way embeddings are computed can be changed arbitrarily.
- We can use the presented approach to e.g. schedule in-network processing service deployments!

Takeaway message part I

1. The ‘way’ a MIP is formulated can have a significant impact on the solver’s ability to *efficiently* solve it.
2. Our approach proposes a very general framework for offline scheduling of networking tasks.

References I

[1] A. Fischer, J. Botero, M. Beck, H. De Meer, and X. Hesselbach.
Virtual network embedding: A survey.
2013.

[2] C. A. Floudas and X. Lin.
Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review.
Computers & Chemical Engineering, 28(11), 2004.

[3] T. A. Henzinger, A. V. Singh, V. Singh, T. Wies, and D. Zufferey.
A marketplace for cloud resources.
In *Proc. of the ACM EMSOFT '10*, 2010.

[4] L. Mai, E. Kalyvianaki, and P. Costa.
Exploiting time-malleability in cloud-based batch processing systems.
In *Proceeding of the ACM SIGOPS LADIS Workshop '13*, 2013.

[5] S. Jain et al.
B4: experience with a globally-deployed software defined wan.
In *Proc. of the ACM SIGCOMM '13*, 2013.

References II

[6] A. Singla, A. Singh, K. Ramachandran, L. Xu, and Y. Zhang.
Proteus: A topology malleable data center network.
In *Proc. of the ACM SIGCOMM HotNets Workshop '10*, 2010.

The End

1. Abstract event point model
2. Δ-, Σ- and $c\Sigma$-Model
   - state-space reductions
   - symmetry reduction
3. Greedy heuristic $c\Sigma^{G}_{A}$ based on $c\Sigma$
4. Initial computational evaluation
   - $\Delta \ll \Sigma < c\Sigma$
   - $c\Sigma$: near optimal solutions within one hour
   - $c\Sigma^{G}_{A}$ only approx. 5-10% off optimum

Future Work / Discussion

Modeling

- Consider flexible duration of requests.
- Consider delay-tolerant VNets.
- Consider more complex scenarios, e.g. migrations.

Algorithmic

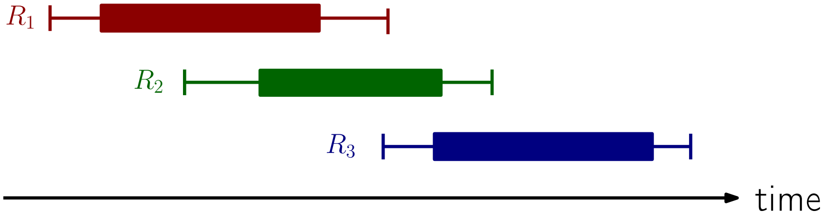
- Incorporate other heuristical embedding approaches.
- Develop local-search algorithms for the TVNEP.

Related Work

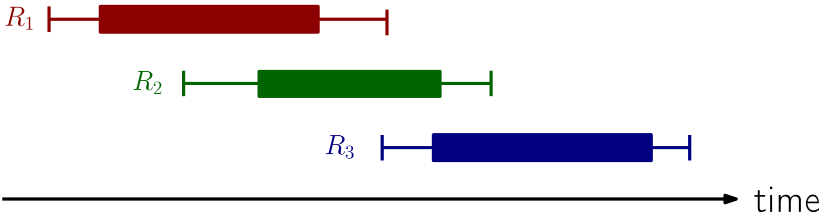
Chemical plants [2]	Utilize similar event abstraction, but no resource sharing.
Business Perspective [3]	Marketplace based on temporal flexibilities.
MapReduce [4]	Consider temporally predictable jobs (MapReduce-like) and allow for temporally interleaved resource sharing.
VNet Survey [1]	There is no comparable work on TVNEP.
Google B4 [5]	Software-defined network (wide-area) connecting data centers. Only some dozen locations.

Optimization: Temporal Dependency Graph

Temporal Dependency Graph



Temporal Dependency Graph



Latest possible point in time for R_1 to start is less than the earliest point in time at which R_2 can start.

$\Rightarrow$ We know that R_1 must start before R_2.

Temporal Dependency Graph

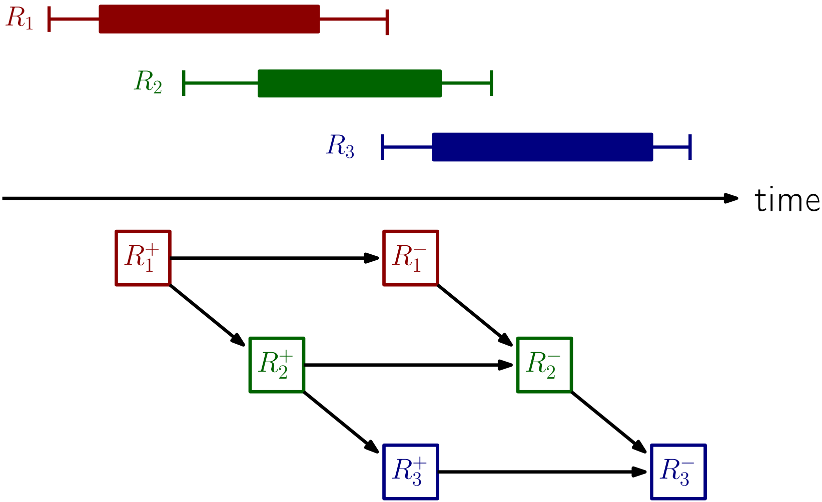


Figure: Temporal Dependency Graph

Temporal Dependency Graph (Formal)

Definition

- $G_{dep}(\mathcal{R}) = (V_{dep}, E_{dep})$
- $V_{dep} = \mathcal{R} \times \{start, end\}$
- $E_{dep} = \{(v, w) \in V_{dep}^2 | latest(v) < earliest(w)\}$

$$earliest((\mathrm{R}, t) \in V_{dep}) = \begin{cases} t_R^s & \text{, if } t = start \\ t_R^s + \mathbf{d}_\mathrm{R} & \text{, if } t = end \end{cases}$$

$$latest((\mathrm{R}, t) \in V_{dep}) = \begin{cases} t_R^e - \mathbf{d}_\mathrm{R} & \text{, if } t = start \\ t_R^e & \text{, if } t = end \end{cases}$$

Weighted Temporal Dependency Graph

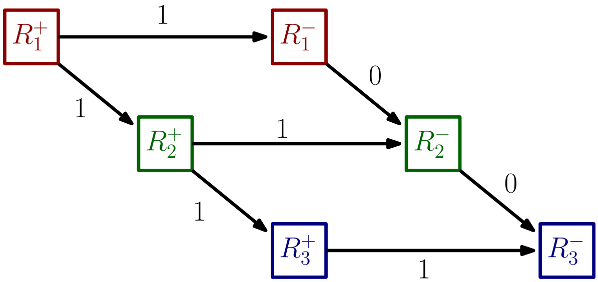


Figure: Temporal Dependency Graph with weights

By computing maximal distances (in polynomial time) we obtain:

- Start of R_1: e_1
- Start of R_2: e_2
- Start of R_3: e_3
- End of R_1: e_2, e_3, e_4
- End of R_2: e_3, e_4
- End of R_1: e_4

Overview cΣ-Model

Access Control & Resource Mapping

Variables

Access Control		$\mathrm{x}_{\mathcal{R}} : \mathcal{R} \to \mathbb{B}$
Node Mapping	$\forall \mathrm{R} \in \mathcal{R}.$	$\mathrm{x}_V : \mathbf{V}_\mathrm{R} \times \mathbf{V}_\mathbf{S} \to \mathbb{B}$
Link Mapping	$\forall \mathrm{R} \in \mathcal{R}.$	$\mathrm{x}_E : \mathbf{E}_\mathrm{R} \times \mathbf{E}_\mathbf{S} \to [0,1]$

Node mapping: $\forall \mathrm{R} \in \mathcal{R}.\ \forall N_v \in \mathbf{V}_\mathrm{R}.$

$$\mathrm{x}_{\mathcal{R}}(\mathrm{R}) = \sum_{N_s \in \mathbf{V}_\mathbf{S}} \mathrm{x}_V(N_v, N_s)$$

Link mapping: $\forall \mathrm{R} \in \mathcal{R}. \forall L_v = (N_v^+, N_v^-) \in \mathbf{E}_\mathrm{R}. \forall N_s \in \mathbf{V}_\mathbf{S}$

$$\sum_{L_s \in \delta^+(N_s)} \mathrm{x}_E(L_v, L_s) - \sum_{L_s \in \delta^-(N_s)} \mathrm{x}_E(L_v, L_s) = \mathrm{x}_V(N_v^-, N_s) - \mathrm{x}_V(N_v^+, N_s)$$

Macro $alloc_V(R, N_s)$: $\forall R \in \mathcal{R}. \forall N_s \in \mathbf{V}_\mathbf{S}$

$alloc_V(R, N_s) = \sum_{N_v \in \mathbf{V}_\mathrm{R}} \mathbf{c}_\mathrm{R}(N_v) \cdot \mathrm{x}_V(N_v, N_s)$

Macro $alloc_V(R, N_s)$: $\forall R \in \mathcal{R}. \forall L_s \in \mathbf{E}_\mathbf{S}$

$alloc_E(R, L_s) = \sum_{L_v \in \mathbf{E}_\mathrm{R}} \mathbf{c}_\mathrm{R}(L_v) \cdot \mathrm{x}_E(L_v, L_s)$

Access Control & Resource Mapping

Mapping onto Event Points

Variables

- $\forall \mathsf{R} \in \mathcal{R}.\ \chi^{+}_{\mathsf{R}} : \mathcal{E} \rightarrow \mathbb{B}$
- $\forall \mathsf{R} \in \mathcal{R}.\ \chi^{-}_{\mathsf{R}} : \mathcal{E} \rightarrow \mathbb{B}$

Mapping each start / end: $\forall \mathsf{R} \in \mathcal{R}.$

$$\sum_{\mathbf{e}_i \in \{\mathbf{e}_1, \ldots, \mathbf{e}_{|\mathcal{R}|}\}} \chi^{+}_{\mathsf{R}}(\mathbf{e}_i) = 1 \qquad \sum_{\mathbf{e}_i \in \{\mathbf{e}_2, \ldots, \mathbf{e}_{|\mathcal{R}|+1}\}} \chi^{-}_{\mathsf{R}}(\mathbf{e}_i) = 1$$

Mapping start injectively: $\forall \mathbf{e}_i \in \{\mathbf{e}_1, \ldots, \mathbf{e}_{|\mathcal{R}|}\}.$

$$\sum_{\mathsf{R} \in \mathcal{R}} \left(\chi^{+}_{\mathsf{R}}(\mathbf{e}_i)\right) = 1$$

Access Control & Resource Mapping

Mapping onto Event Points

Guaranteeing State Feasibility

Variables

$alloc_V : \mathcal{R} \times \mathcal{S} \times \mathbf{V_S} \to \mathbb{R}_{\geq \not{0}}$ $\quad$ $alloc_E : \mathcal{R} \times \mathcal{S} \times \mathbf{E_S} \to \mathbb{R}_{\geq \not{0}}$

Computing allocations at states: $\forall \mathrm{R} \in \mathcal{R}. \forall \mathbf{s}_i \in \mathcal{S}. \forall N_s \in \mathbf{V_S} / \forall L_s \in \mathbf{E_S}$.

- $alloc_V(\mathrm{R}, \mathbf{s}_i, N_s) \geq alloc_V(\mathrm{R}, N_s) - \mathbf{c_S}(N_s) \cdot (1 - \Sigma(R, \mathbf{e}_i))$
- $alloc_E(\mathrm{R}, \mathbf{s}_i, L_s) \geq alloc_E(\mathrm{R}, L_s) - \mathbf{c_S}(L_s) \cdot (1 - \Sigma(R, \mathbf{e}_i))$

Ensuring feasibility: $\forall \mathbf{s}_i \in \mathcal{S}. \forall N_s \in \mathbf{V_S} / L_s \in \mathbf{E_S}$.

- $\mathbf{c_S}(N_s) \geq \sum_{\mathrm{R} \in \mathcal{R}} alloc_V(\mathrm{R}, \mathbf{s}_i, N_s)$
- $\mathbf{c_S}(L_s) \geq \sum_{\mathrm{R} \in \mathcal{R}} alloc_E(\mathrm{R}, \mathbf{s}_i, L_s)$

Access Control & Resource Mapping

Mapping onto Event Points

Guaranteeing State Feasibility

Guaranteeing Temporal Feasibility

Variables

$\forall \mathsf{R} \in \mathcal{R}. t^+_{\mathsf{R}}, t^-_{\mathsf{R}} \in \mathbb{R}_{\geq 0} \qquad \forall \mathbf{e}_i \in \mathcal{E}. t_{\mathbf{e}_i} \in \mathbb{R}_{\geq 0}$

$\forall \mathsf{R} \in \mathcal{R}.$

$\mathbf{d}_{\mathsf{R}} = t^-_{\mathsf{R}} - t^+_{\mathsf{R}}$

Setting start times: $\forall \mathsf{R} \in \mathcal{R}. \forall \mathbf{e}_i \in \{\mathbf{e}_1, \ldots, \mathbf{e}_{|\mathcal{R}|}\}$.

$$t^+_{\mathsf{R}} \leq t_{\mathbf{e}_i} + (1 - \sum_{j=1,\ldots,i} \chi^+_{\mathsf{R}}(\mathbf{e}_j, \mathsf{R})) \cdot \mathbf{T} \qquad t^+_{\mathsf{R}} \geq t_{\mathbf{e}_i} - (1 - \sum_{j=i,\ldots,|\mathcal{E}|} \chi^+_{\mathsf{R}}(\mathbf{e}_j, \mathsf{R})) \cdot \mathbf{T}$$

Setting end times: $\forall \mathsf{R} \in \mathcal{R}. \forall \mathbf{e}_i \in \{\mathbf{e}_2, \ldots, \mathbf{e}_{|\mathcal{R}|+1}\}$.

$$t^-_{\mathsf{R}} \leq t_{\mathbf{e}_i} + (1 - \sum_{j=2,\ldots,i} \chi^-_{\mathsf{R}}(\mathbf{e}_j, \mathsf{R})) \cdot \mathbf{T} \qquad t^-_{\mathsf{R}} \geq t_{\mathbf{e}_{i-1}} - (1 - \sum_{j=i,\ldots,|\mathcal{E}|} \chi^-_{\mathsf{R}}(\mathbf{e}_j, \mathsf{R})) \cdot \mathbf{T}$$

Access Control & Resource Mapping

Mapping onto Event Points

Guaranteeing State Feasibility

Guaranteeing Temporal Feasibility

Temporal Dependency Graph User Cuts

$\forall v \in V_{dep}.$

$$\sum_{i=|dist^{+}_{\max}(v)|+1}^{|\mathcal{R}|+1-|dist^{-}_{\max}(v)|} \chi_{Event}(\mathbf{e}_i, v) = 1$$

$\forall v \in V_{dep}. \forall w \in dist^{-}_{\max}(v). \forall \mathbf{e}_i \in \mathcal{E}, dist_{\max}(v, w) + 1 \leq i \leq |\mathcal{R}|.$

$$\sum_{j=1}^{i} \chi_{Event}(\mathbf{e}_j, w) \leq \sum_{\substack{\mathbf{e}_j \in \mathcal{E} \\ \text{with } j \leq i - dist^{-}_{\max}(v,w)}} \chi_{Event}(\mathbf{e}_j, v)$$

Access Control & Resource Mapping

Mapping onto Event Points

Guaranteeing State Feasibility

Guaranteeing Temporal Feasibility

Temporal Dependency Graph User Cuts

Some further optimizations

- Big-M constants are chosen as *tight* as possible
- virtual links can be aggregated if their virtual source or their virtual destination is the same

Applications

Data center

- e.g. MapReduce cycles through different phases, traffic only during 30-60% of execution [6]
- price incentives for customers and providers to allow for / harness temporal flexibility [4]

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

Part II: Virtualized In-Network Processing

Mindset

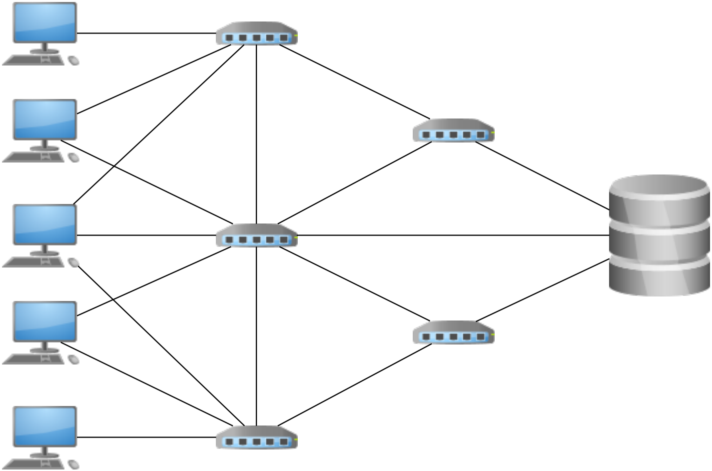
Service Deployment $\neq$ VNet Embedding

- Customer requests communication *service* between locations.
- Service provider *finds* an appropriate topology.

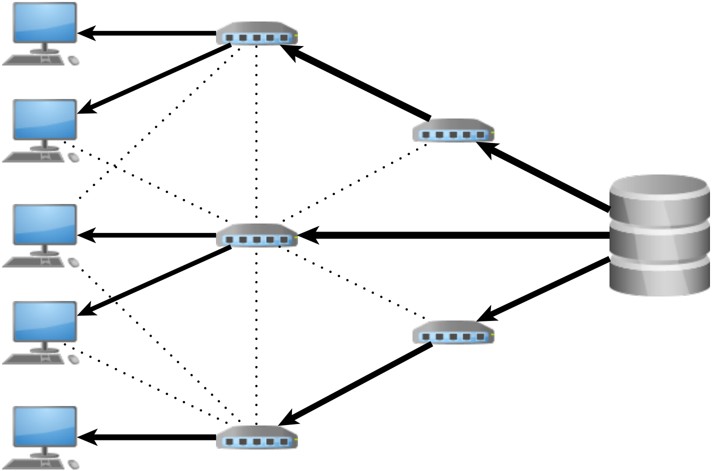
Publications

- Joint work with Stefan Schmid: OPODIS 2013, arXiv 2013
- My thesis January 2014

Service Replication

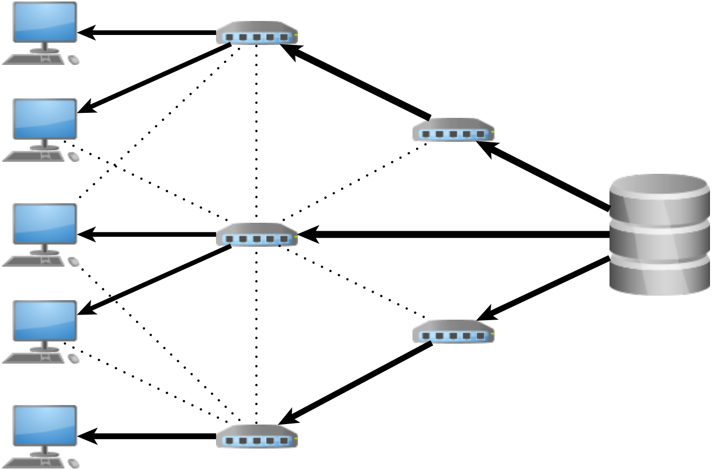


Service Replication

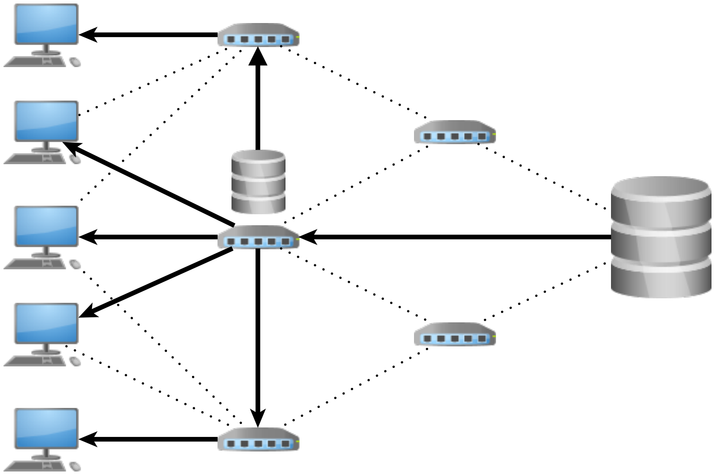


Service Replication

What if backend links are congested?

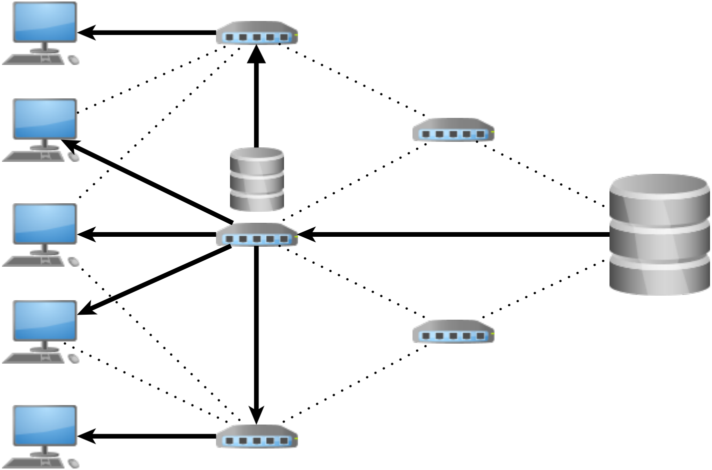


Service Replication

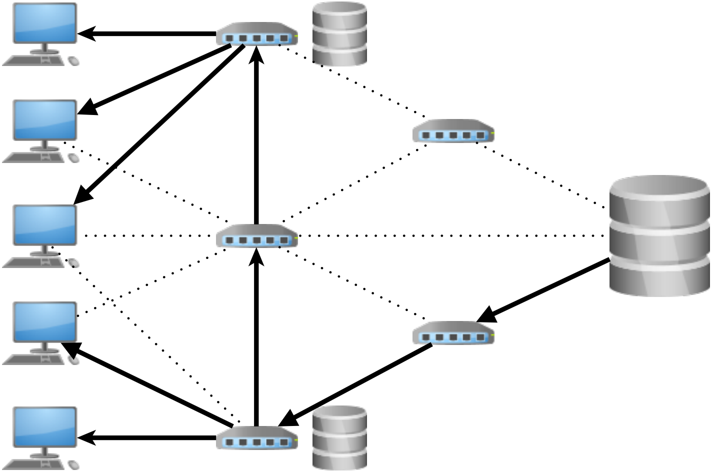


Service Replication

What if only '3' users can be handled?



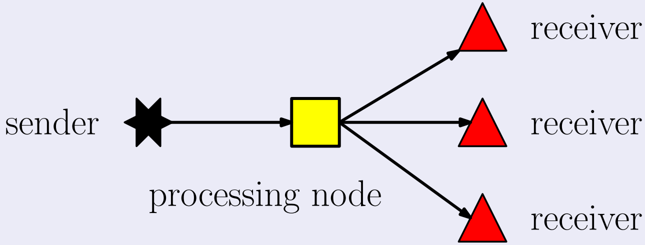
Service Replication



Generalized Communication Services

Communication Services: Multicast

processing = duplication + reroute



Communication Services: Multicast

processing = duplication + reroute

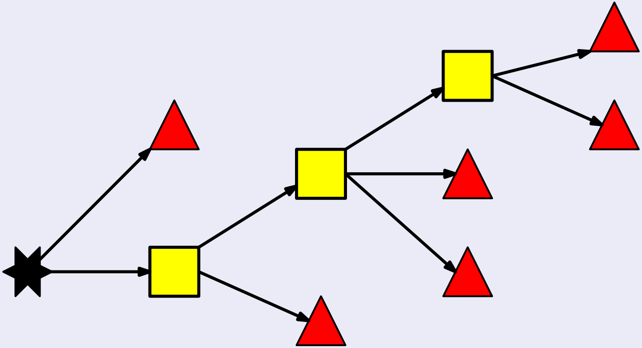
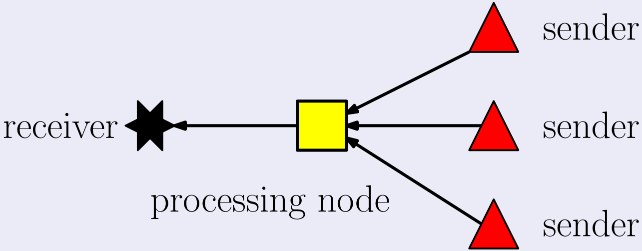


Figure: Hierarchy of processing nodes

Communication Services: Aggregation

processing = merge + reroute



Communication Services: Aggregation

processing = merge + reroute

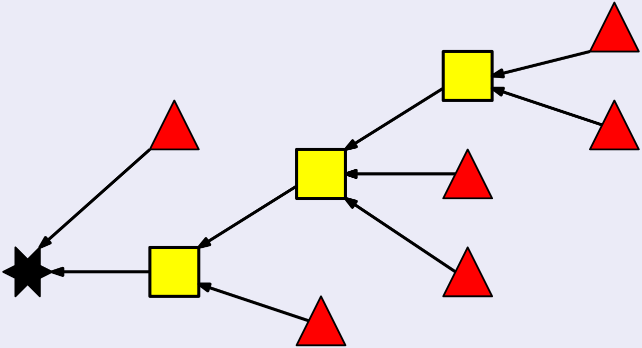


Figure: Hierarchy of processing nodes

Problem Statement

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

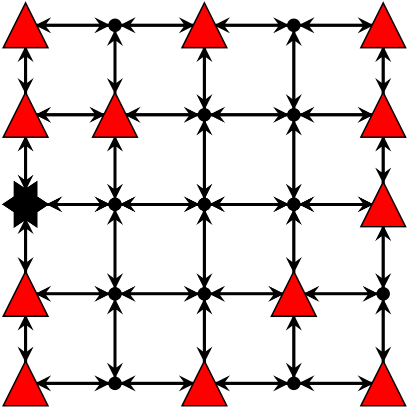
Introductory Example

Aggregation scenario

grid graph: 14 senders, one receiver

Virtualized links

data can be routed arbitrarily



Without in-network processing: Unicast

Solution Method

- minimal cost flow

Solution uses

- 41 edges
- 0 processing nodes

receiver sender

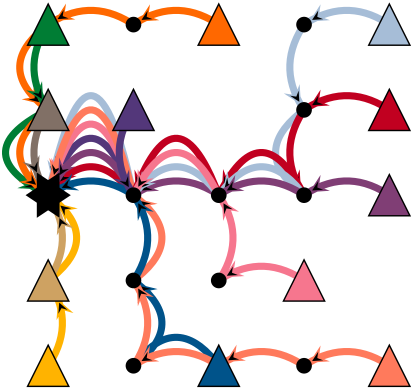


Figure: Unicast solution

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

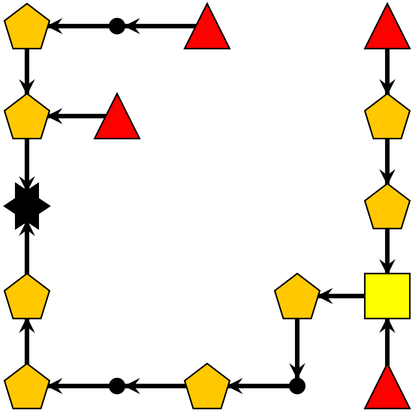
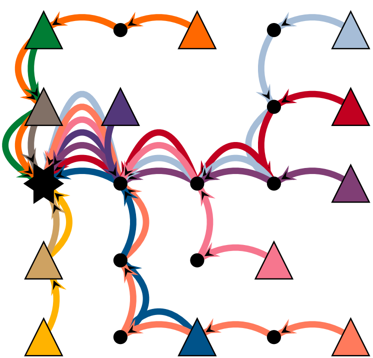
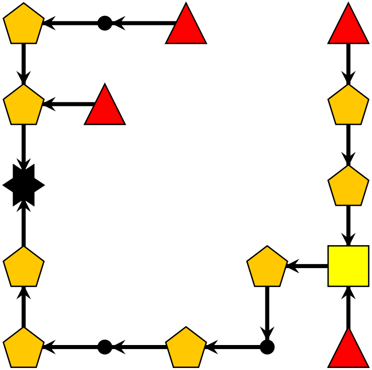


Figure: Aggregation tree

How to Trade-off?



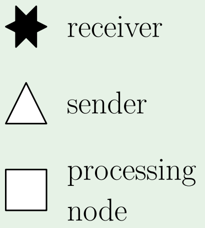
vs.

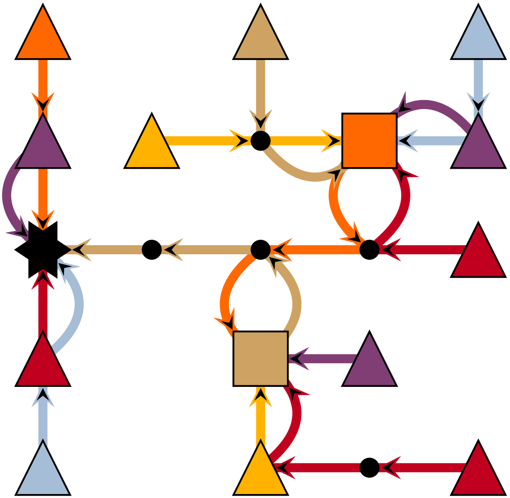


What we aim for

Solution uses

- 26 edges
- 2 processing nodes





Solution Structure

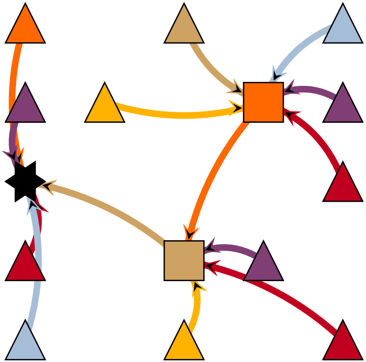


Figure: Virtual Arborescence

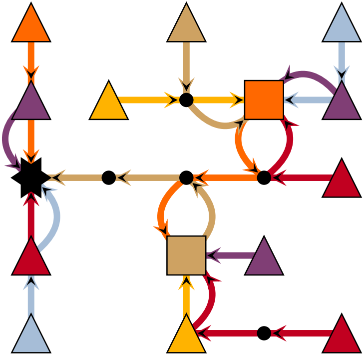


Figure: underlying routes

New Model: Constrained Virtual Steiner Arborescence Problem

Definition: CVSAP

Find a Virtual Arborescence connecting senders to the single receiver, s.t.

1. bandwidth of substrate is not exceeded,
2. inner nodes are capable of processing flow,
3. the processing nodes' capacities are not exceeded,

minimizing the joint cost for bandwidth allocations and function placement.

Applications Overview

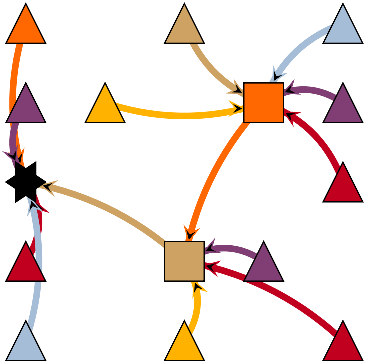
	Network	Application	Technology, e.g.
multicast	ISP	service replication / cache placement [7, 8]	middleboxes / NFV + SDN
	backbone	optical multicast [4]	ROADM + SDH
	all	application-level multicast [11]	different
aggregation	sensor network	value & message aggregation [3, 5]	source routing
	data center	big data / map-reduce: Camdoop [1]	SDN
	ISP	network analytics: Gigascope [2]	middleboxes / NFV + SDN

MIP Formulations

Multi-Commodity Flow (MCF) Integer Program

First approach: MCF IP

- explicitly represent virtual arborescence by the explicit construction of paths for all processing nodes
- enforces loop-freeness of connections



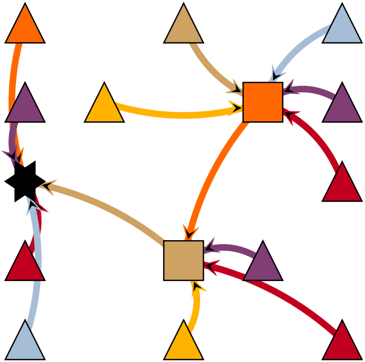
Multi-Commodity Flow (MCF) Integer Program

First approach: MCF IP

- explicitly represent virtual arborescence by the explicit construction of paths for all processing nodes
- enforces loop-freeness of connections

Does not scale well

- number of binary variables: #processing nodes · #edges
- 'not so good relaxations' due to weak loop-freeness formulation



Integer Program 1: A-CVSAP-MCF

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

subject to

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

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

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

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

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

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

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

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

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

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

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

Single-Commodity Flow IP

Single-commodity flow formulation

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

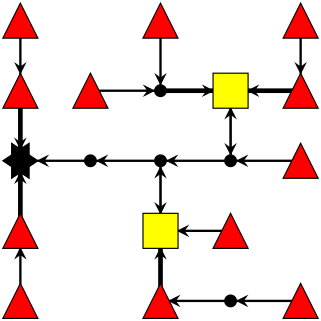


Figure: Single-commodity

Multi- vs Single-Commodity

Example: 6000 edges and 200 Steiner sites

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

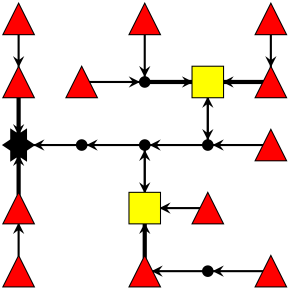


Figure: Single-commodity

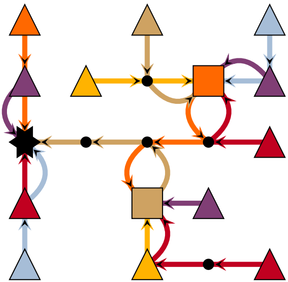


Figure: Multi-commodity

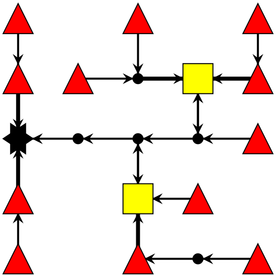
VirtuCast Algorithm

VirtuCast Algorithm

Outline of VirtuCast

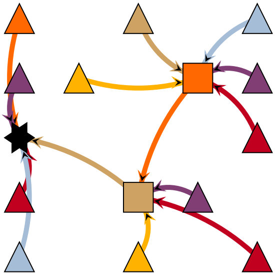
1. Solve single-commodity flow IP formulation.
2. Decompose IP solution into Virtual Arborescence.

How to decompose?



(a) IP solution

$\rightarrow$



(b) Virtual Arborescence

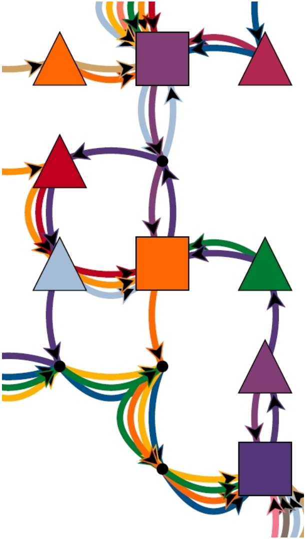
Decomposing flow is non-trivial (5 pages proof)!

Flow solution . . .

- contains cycles and
- represents *arbitrary* hierarchies.

Main Results

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



Integer Program 2: IP-A-CVSAP

$$\begin{aligned}
\text{minimize} \quad & C_{\mathrm{IP}}(x,f) = \sum_{e \in E_G} \mathbf{c}_e f_e + \sum_{s \in S} \mathbf{c}_s x_s && && \text{(IP-OBJ)}\\
\text{subject to} \quad & f(\delta^+_{E_{\mathrm{ext}}}(v)) = f(\delta^-_{E_{\mathrm{ext}}}(v)) && \forall\ v \in V_G && \text{(IP-1)}\\
& f(\delta^+_{E^R_{\mathrm{ext}}}(W)) \geq x_s && \forall\ W \subseteq V_G, s \in W \cap S \neq \emptyset && \text{(IP-2)}\\
& f(\delta^+_{E^R_{\mathrm{ext}}}(W)) \geq 1 && \forall\ W \subseteq V_G, T \cap W \neq \emptyset && \text{(IP-3}^\star\text{)}\\
& f_e \geq x_s && \forall\ e = (s, \mathrm{o}^-_S) \in E^{S-}_{\mathrm{ext}} && \text{(IP-4}^\star\text{)}\\
& f_e \leq \mathbf{u}_s x_s && \forall\ e = (s, \mathrm{o}^-_S) \in E^{S-}_{\mathrm{ext}} && \text{(IP-5)}\\
& f_{(r,\mathrm{o}^-_r)} \leq \mathbf{u}_r && && \text{(IP-6)}\\
& f_e \leq \mathbf{u}_e && \forall\ e \in E_G && \text{(IP-7)}\\
& f_e = 1 && \forall\ e \in E^{T+}_{\mathrm{ext}} && \text{(IP-8)}\\
& f_e = x_s && \forall\ e = (\mathrm{o}^+, s) \in E^{S+}_{\mathrm{ext}} && \text{(IP-9)}\\
& x_s \in \{0,1\} && \forall\ s \in S && \text{(IP-10)}\\
& f_e \in \mathbb{Z}_{\geq 0} && \forall\ e \in E_{\mathrm{ext}} && \text{(IP-11)}
\end{aligned}$$

VirtuCast Heuristics

Outline

- Solving the linear relaxation of the VirtuCast can be done reasonably fast (magnitudes faster than the multi-commodity flow formulation)
- Allows for LP based heuristics that iteratively recompute relaxations

FlowDecoRound Heuristic

- Given a LP solution, computes a flow decomposition and connects nodes randomly according to it
- processing nodes are activated if another node node connects to it, must be connected itself
- failsafe: shortest paths
- uses only a single LP relaxation

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 MIP Formulations

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 $u(e) \triangleq \begin{cases} u_E(e) & \text{, if } e \in E_G \\ u_r(r) & \text{, if } e = (r, \mathrm{o}_r^-) \\ u_S(s) & \text{, if } e = (s, \mathrm{o}_S^-) \in E_{\mathrm{ext}}^{S-} \\ 1 & \text{, else} \end{cases}$ **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** $\Gamma_t \triangleq \texttt{MinCostFlow}\left(G_{\mathrm{ext}}, \hat{f}, \hat{f}(\mathrm{o}^+, t), t, \{\mathrm{o}_S^-, \mathrm{o}_r^-\}\right)$
- **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** $e = (s, \mathrm{o}_S^-) \in E_{\mathrm{ext}}^{S-}$ **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** $P \leftarrow \texttt{ShortestPath}\left(G^u_{\mathrm{ext}}, c_E, t, \{\mathrm{o}_S^-, \mathrm{o}_r^-\}\right)$ **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`$(\hat{\mathcal{T}}_G)$

MultipleShots

- treats node variables as probabilities and iteratively places processing functionality accordingly
- tries to generate feasible solutions by employing a generalized variant of Prim's MST algorithm

Algorithm 2: 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 MIP Formulations

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 3: GreedyDiving

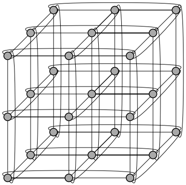
Input : Network $G = (V_G, E_G, c_E, u_E)$, Request $R_G = (r, S, T, u_r, c_S, u_S)$, LP relaxation solution $(\hat{x}, \hat{f}) \in \mathcal{F}_{\mathrm{LP}}$ to MIP Formulations

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

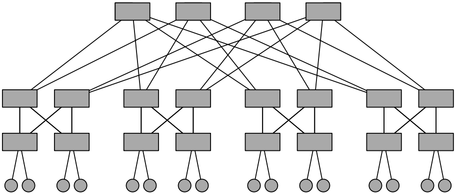
set $\lfloor S \rfloor \triangleq \{s \in S | \hat{x}_s \leq 0.01\}$ **and** $\lceil S \rceil \triangleq \{s \in S | \hat{x}_s \geq 0.99\}$
`addConstraintsLocally`$(\{x_s = 0 | s \in \lfloor S \rfloor\} \cup \{x_s = 1 | s \in \lceil S \rceil\})$
set $\dot{S} \triangleq \lfloor S \rfloor \cup \lceil S \rceil$ **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 $\dot{E} \neq E_{\mathrm{ext}}$ **then**
set $\lfloor E \rfloor \triangleq \{e \in E_{\mathrm{ext}} | \, |\hat{f}_e - \lfloor \hat{f}_e \rfloor| \leq 0.001\}$, $\lceil E \rceil \triangleq \{e \in E_{\mathrm{ext}} | \, |\hat{f}_e - \lceil \hat{f}_e \rceil| \leq 0.001\}$
`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 $\hat{E} \triangleq E_{\mathrm{ext}} \setminus \dot{E}$
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** $e \in E_{\mathrm{ext}} \setminus \dot{E}$
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

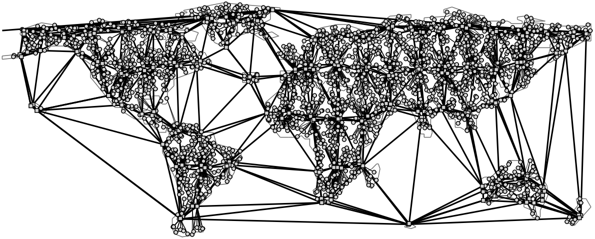
Topologies



3D torus



Fat tree



An ISP topology generated by IGen with 2400 nodes.

Instances

Generation Parameters

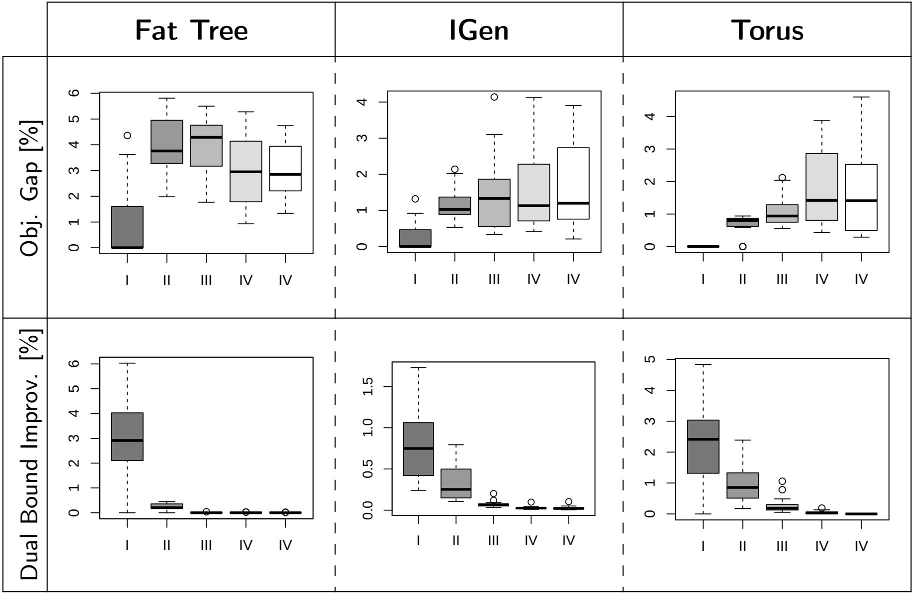
- five graph sizes I-V
- 15 instances per graph size: different Steiner costs, different edge capacities

	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

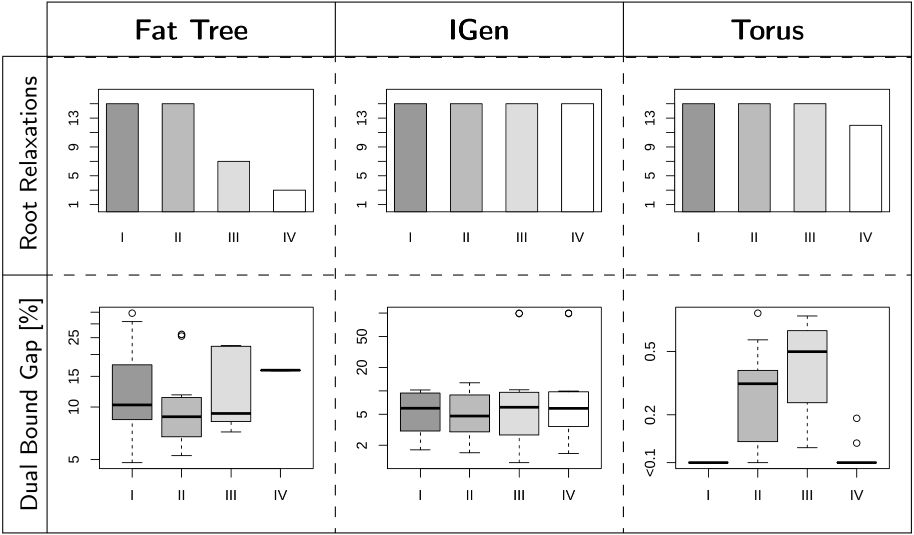
VirtuCast + LP-based Heuristics

VirtuCast + LP-based Heuristics



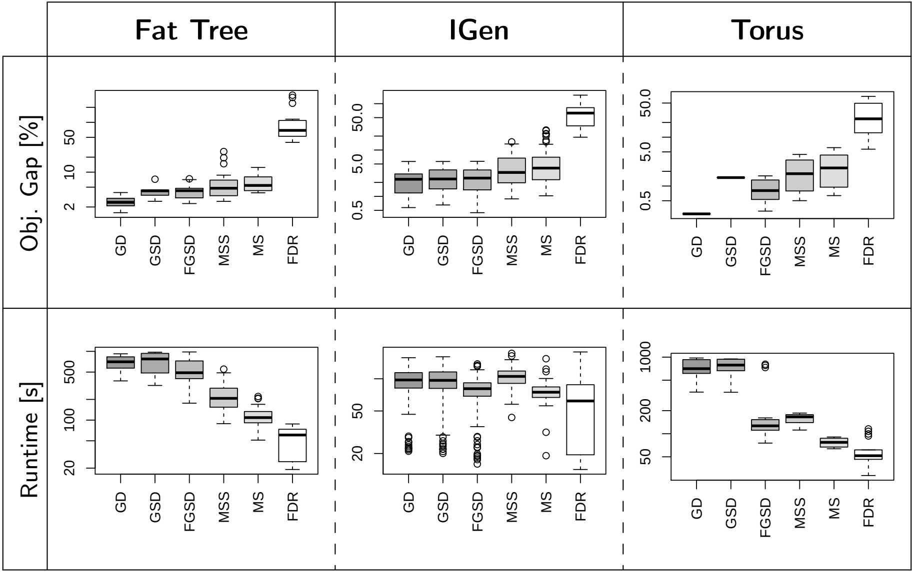
MCF-IP

MCF-IP: Performance



VirtuCast LP Heuristics

LP-based Heuristics: Performance on graph size V



Conclusion

Summary

Applications → Concise definition of CVSAP

Algorithmic Study

Inapproximability

Approximations

- NVSTP
- VSTP
- VSAP

Exact Algorithms

- multi-commodity flow
- single-commodity flow → VirtuCast

Heuristics

- FlowDecoRound
- MultipleShots
- GreedyDiving
- GreedySelect

Extensive explorative Computational Evaluation

Related Work

Molnar: Constrained Spanning Tree Problems [6]

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

Oliveira et. al: Flow Streaming Cache Placement Problem [8]

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

Shi: Scalability in Overlay Multicasting [11]

- Provided heuristic and showed improvement in scalability.

Applicability

Multicast scenarios can be modeled accurately

Aggregation scenarios...

Sensor Networks	MapReduce	GigaScope
- commutative - associative	good aggregation in general possible	aggregation factor of n:1 unrealistic

fully applicable ⟷ not really applicable

Future Work

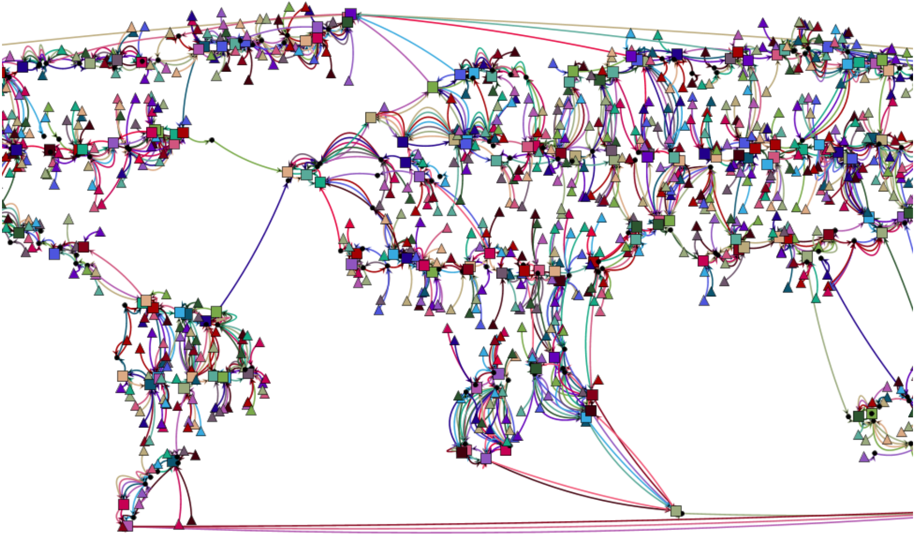
Model Extensions

- prize-collecting variants
- concurrent multicast / aggregation sessions

Application Modeling

- Stratosphere II: Big Data
- UNIFY EU Project: flow analytics

End Part II



References I

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

References II

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

[7] 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.

[8] 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.

[9] 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.

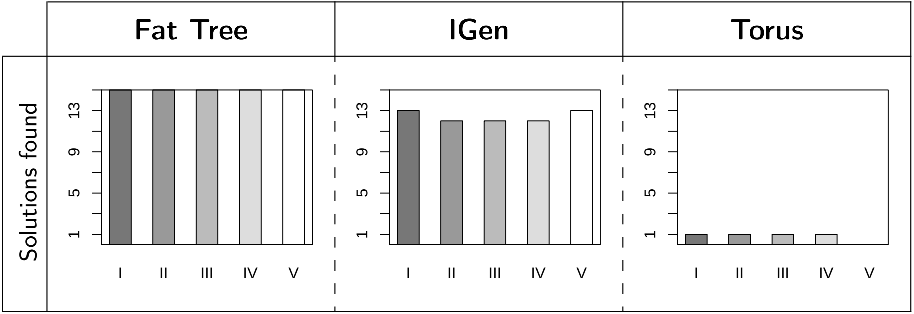
[10] M. Rost and S. Schmid.
Virtucast: Multicast and aggregation with in-network processing.
In R. Baldoni, N. Nisse, and M. Steen, editors, *Principles of Distributed Systems*, volume 8304 of *Lecture Notes in Computer Science*, pages 221–235. Springer International Publishing, 2013.

References III

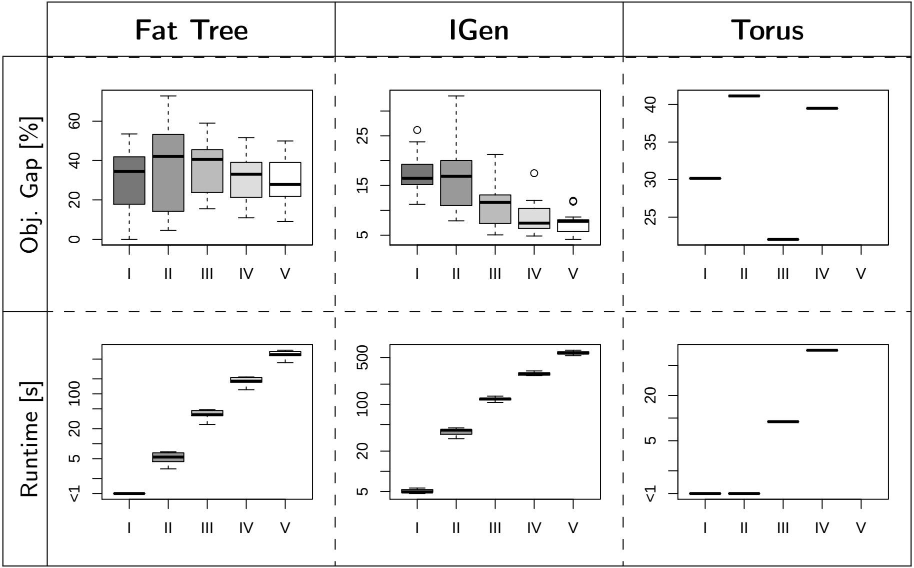
[11] S. Shi.
A proposal for a scalable internet multicast architecture.
In *Washington Universtiy*, 2001.

GreedySelect

GreedySelect: Efficacy

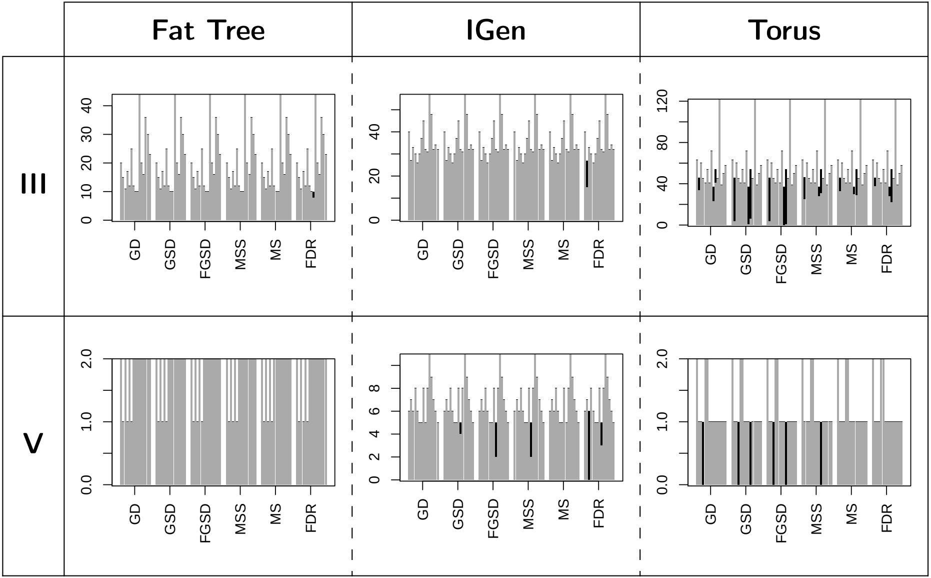


GreedySelect: Performance

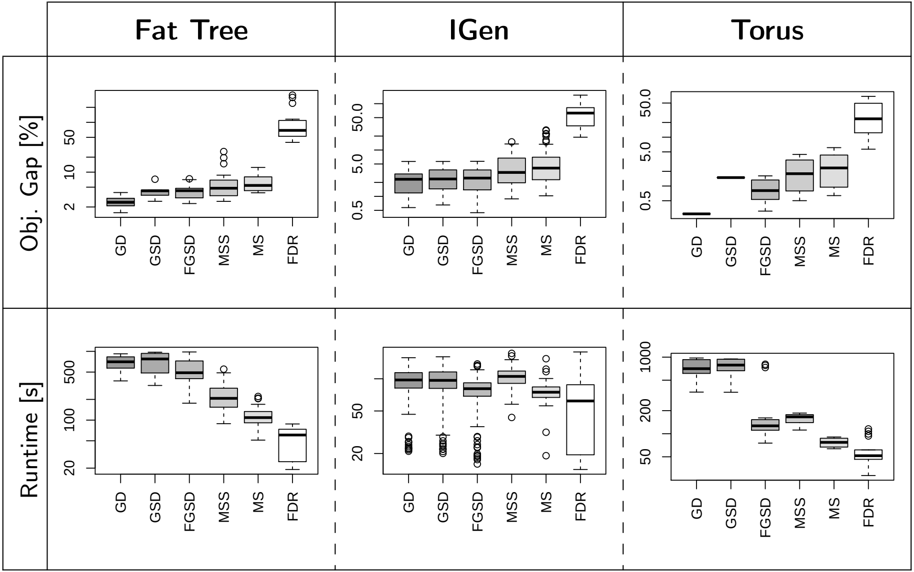


LP-based Heuristics

LP-based Heuristics: Efficacy



LP-based Heuristics: Performance on graph size V



Publications

Matthias Rost, Stefan Schmid: OPODIS 2013 & arXiv [10, 9]

Applications → Concise definition of CVSAP

Algorithmic Study

Inapproximability

Approximations

- NVSTP
- VSTP
- VSAP

Exact Algorithms

- multi-commodity flow
- single-commodity flow
  → VirtuCast

Heuristics

- FlowDecoRound
- MultipleShots
- GreedyDiving
- GreedySelect

Extensive explorative Computational Evaluation

Solution Approaches

Solution Approaches

Wishful thinking: there exists a

- polynomial time algorithm
- solving CVSAP to optimality
- considering all constraints.

Solution Approaches

Wishful thinking: there exists a

- polynomial time algorithm
- solving CVSAP to optimality
- considering all constraints.

Theorem: Inapproximability of CVSAP

Finding a feasible solution is NP-complete!

Solution Approaches

Wishful thinking: there exists a

- polynomial time algorithm
- solving CVSAP to optimality
- considering all constraints.

Theorem: Inapproximability of CVSAP

Finding a feasible solution is NP-complete!

Approximations

- polynomial
- <u>quality guarantee</u>
- <u>weaker models</u>

Exact Algorithms

- <u>non-polynomial</u>
- optimality
- full model

Heuristics

- polynomial
- <u>no solution guarantee</u>
- full model

Comprehensive algorithmic study

Algorithms

Approximations
- NVSTP
- VSTP
- VSAP

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

LP-based Heuristics
- FlowDecoRound
- MultipleShots
- GreedyDiving

Combinatorial Heuristic
- GreedySelect

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

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

Bottom Line

- Better understanding of how to incorporate *virtualized links*.
- Obtained lower bounds via reductions.

Exact Algorithms for CVSAP

Overview

Why exact algorithms matter

- allow trading-off runtime with solution quality
- baseline for heuristics

Choice: Integer Programming (IP)

- successfully employed for solving related problems (STP, CFLP, . . .)
- generates lower bounds on-the-fly

Combinatorial Heuristic: GreedySelect

Combinatorial Heuristics

On Chickens and Eggs

- How and when to place processing nodes?
- How and when to reserve bandwidth for routes?
- How to react to infeasibilities?

Our Approach

- Place processing functionality and reserve bandwidth jointly.
- Try to avoid infeasibilities by proactive routing decisions.

GreedySelect Heuristic

Greedily either ...

- connect a single node to the connected component of the receiver or
- connect multiple nodes to an inactive processing node

minimizing the averaged discounted cost per connected node.

Selecting processing node + terminals + paths : $\mathcal{O}(|V| \cdot |E| + |V|^2 \log |V|)$

$$
\begin{aligned}
\textbf{compute } & \mathcal{P}_{\bar{s}} \triangleq (\bar{s} \in \bar{S}, T' \subseteq \bar{T}, \mathcal{P}_{T'} = \{P_{t,\bar{s}} | t \in T'\}), \\
\textbf{such that } & P_{t,\bar{s}} \text{ connects } t \text{ to } \bar{s}, \\
& u^{\bar{s}}(e) - |\mathcal{P}_{T'}[e]| \geq 0 \text{ for all } e \in E_G, \\
& 2 \leq |T'| \leq u_{r,S}(\bar{s}) \\
\textbf{minimizing } & c_{\bar{s},T'} \triangleq \left(\sum_{t \in T'} (c_E(P_{t,\bar{s}}) - c_E(P_{t,R})) + c_E(P_{\bar{s},R}) + c_S(\bar{s}) \right) / |T'|
\end{aligned}
$$

LP-based 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 4: 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 $(\hat{x}, \hat{f}) \in \mathcal{F}_{\text{LP}}$ to MIP Formulations
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 $u(e) \triangleq \begin{cases} u_E(e) & \text{, if } e \in E_G \\ u_r(r) & \text{, if } e = (r, o_r^-) \\ u_S(s) & \text{, if } e = (s, o_S^-) \in E_{\text{ext}}^{S-} \\ 1 & \text{, else} \end{cases}$ **for all** $e \in E_{\text{ext}}$
while $U \neq \emptyset$ **do**
  choose $t \in U$ uniformly at random **and** set $U \leftarrow U - t$
  set $\Gamma_t \triangleq$ `MinCostFlow` $\left(G_{\text{ext}}, \hat{f}, \hat{f}(o^+, t), t, \{o_S^-, o_r^-\}\right)$
  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** $e = (s, o_S^-) \in E_{\text{ext}}^{S-}$ **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` $\left(G^u_{\text{ext}}, c_E, t, \{o_S^-, o_r^-\}\right)$
    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 $\hat{\pi}(e) \leftarrow \langle P_1, \dots, P_{|P|-1} \rangle$
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 5: 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 6: 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 MIP Formulations

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 7: GreedyDiving

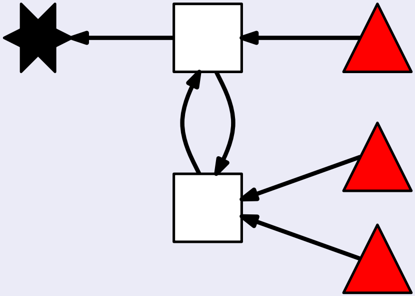
Input : Network $G = (V_G, E_G, c_E, u_E)$, Request $R_G = (r, S, T, u_r, c_S, u_S)$, LP relaxation solution $(\hat{x}, \hat{f}) \in \mathcal{F}_{\text{LP}}$ to MIP Formulations

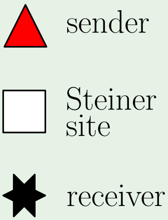
Output: A Feasible Virtual Arborescence $\hat{\mathcal{T}}_G$ or `null`

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`$(G, R_G, \hat{\mathcal{T}}_G^P)$

Example

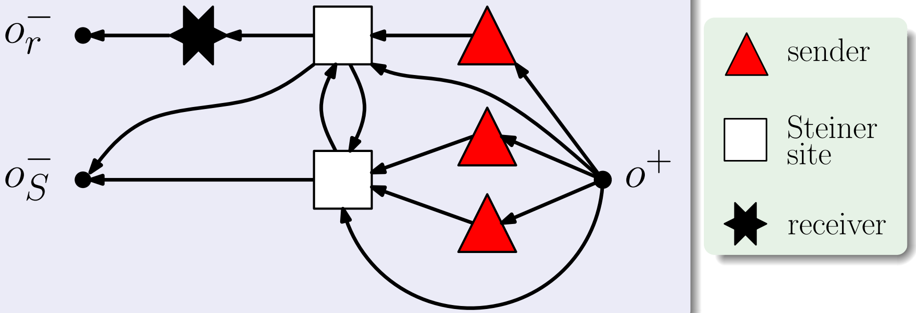
Scenario





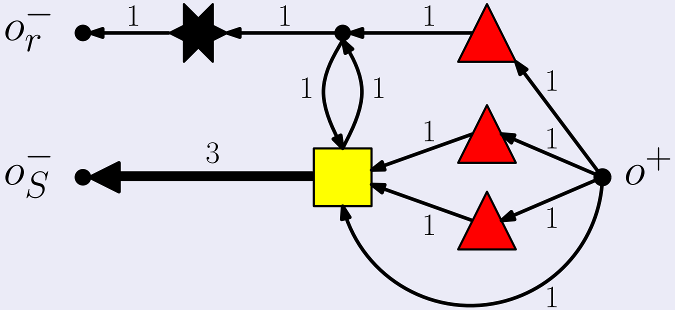
Example

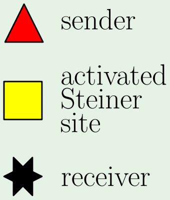
Extended Graph



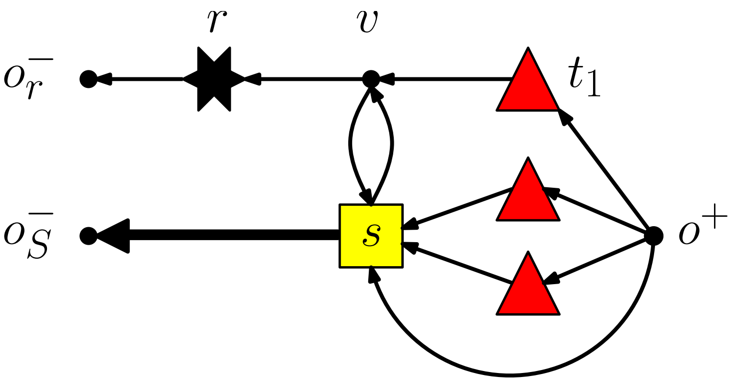
Example

Solution



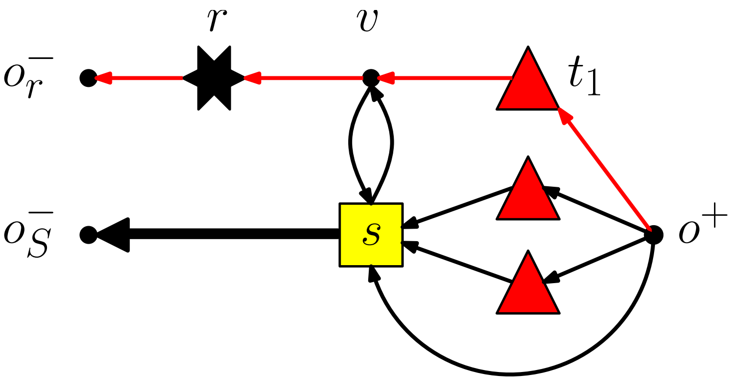


Decomposition Example I



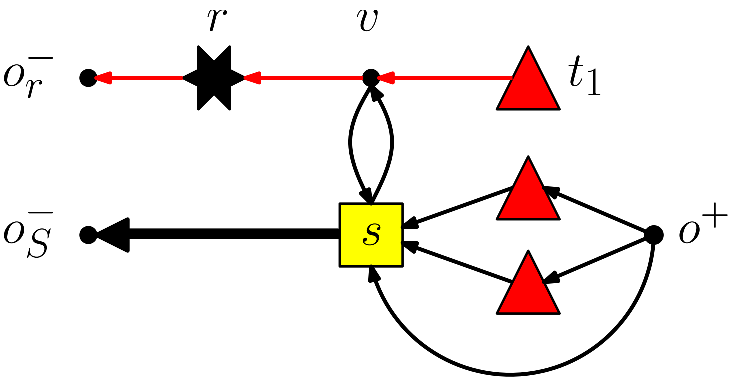
Decomposition Example I

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



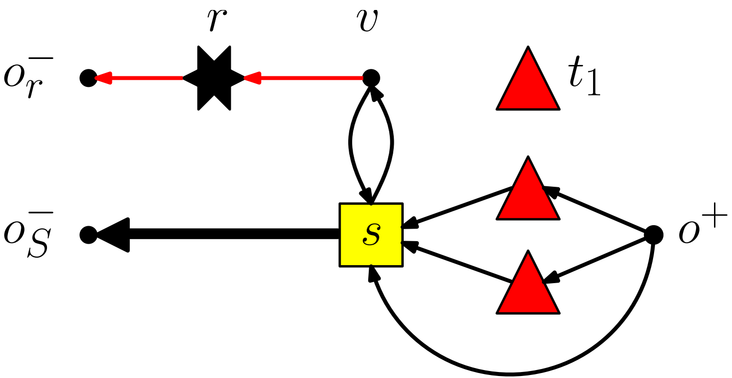
Decomposition Example I

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



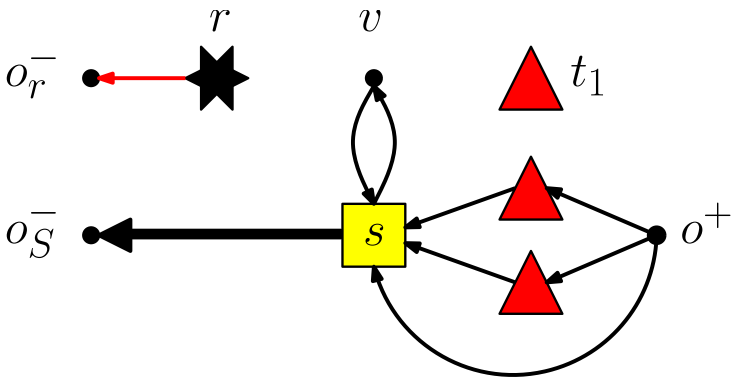
Decomposition Example I

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

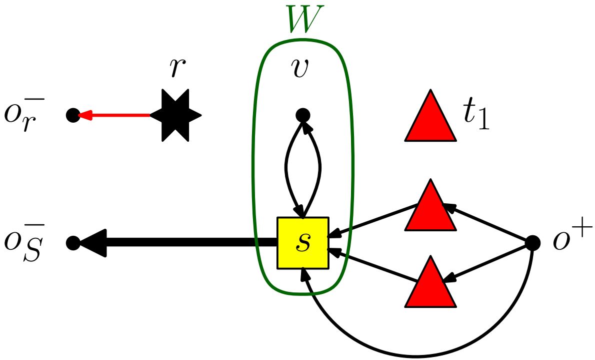


Decomposition Example I

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



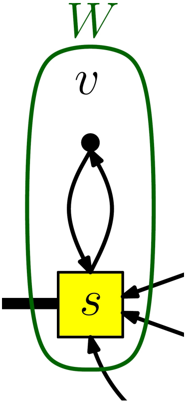
Redirecting Flow



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



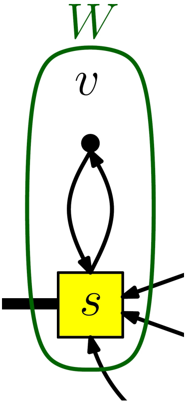
Redirection towards o_S^- is possible!

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

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!

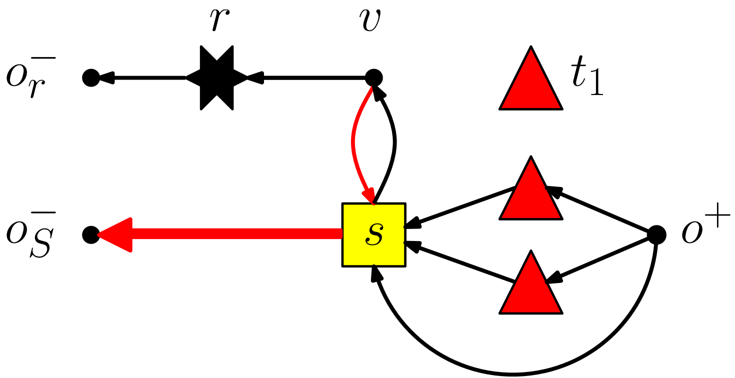
There exists a path from v towards o_S^- in W.

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^-.

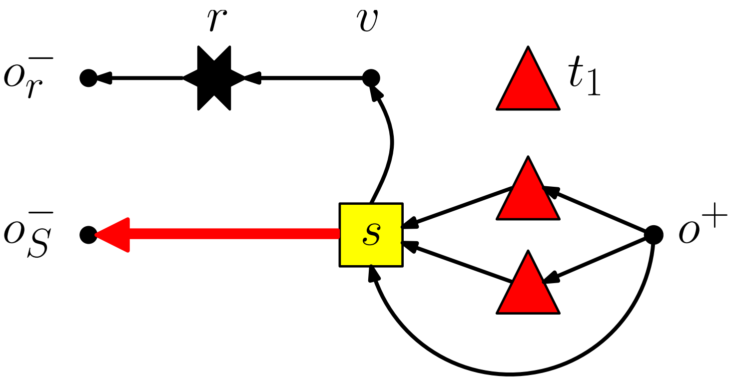
Decomposition Example II

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

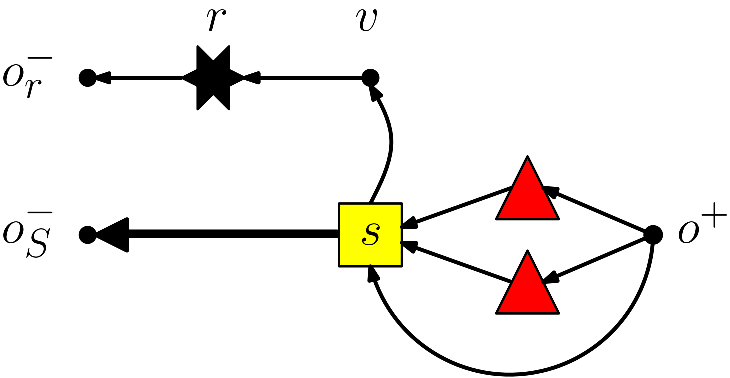


Decomposition Example II

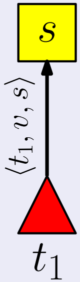
$$P = \langle o^+, t_1, v, s, o_S^- \rangle$$



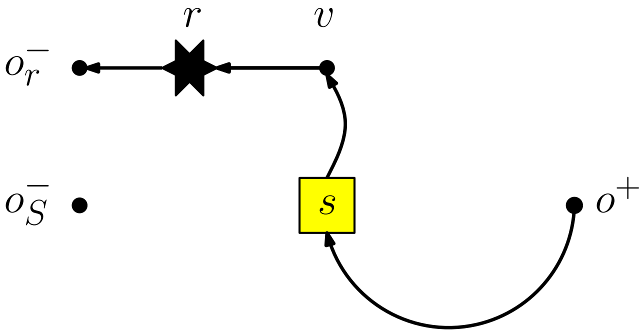
Decomposition Example II



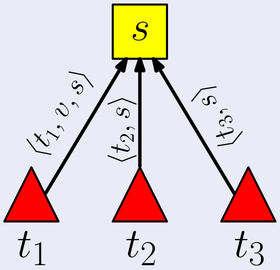
Solution



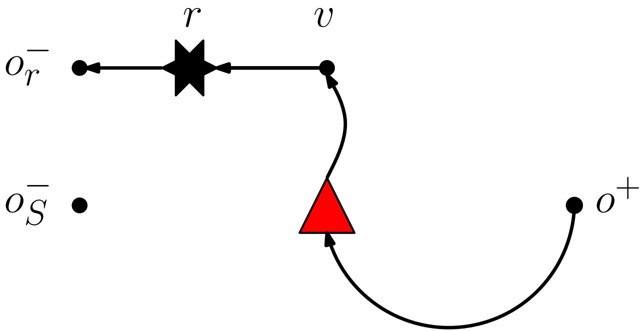
Decomposition Example II



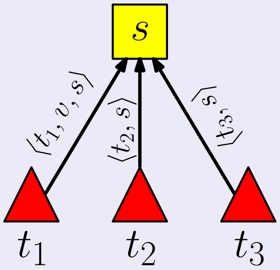
Solution



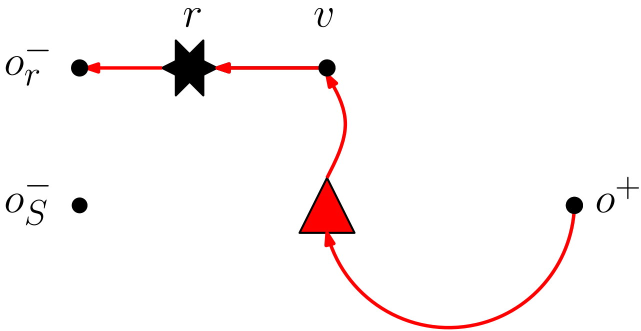
Decomposition Example II



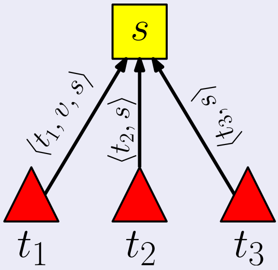
Solution



Decomposition Example II

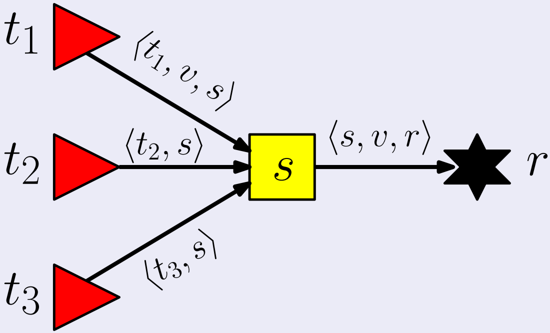


Solution



Decomposition Example II

Final Solution



Related Work

Molnar: Constrained Spanning Tree Problems [6]

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

Oliveira et. al: Flow Streaming Cache Placement Problem [8]

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

Shi: Scalability in Overlay Multicasting [11]

- Provided heuristic and showed improvement in scalability.