Recent Developments on Exact Solvers for the (Prize-Collecting) Steiner Tree Problem

Ivana Ljubić

ESSEC Business School of Paris

The 22nd edition of the
COMEX Belgian Mathematical Optimization Workshop

April 21, 2017, La-Roche-en-Ardennes

This tutorial is based on:

- M. Fischetti, M. Leitner, I. Ljubić, M. Luipersbeck, M. Monaci, M. Resch, D, Salvagnin, M. Sinnl:
  Thinning out Steiner trees: A node based model for uniform edge costs, *Mathematical Programming Computation*, 2016,
  DOI: 10.1007/s12532-016-0111-0, 2016
- M. Leitner, I. Ljubić, M. Luipersbeck, M. Sinnl:
  A dual-ascent-based branch-and-bound framework for the prize-collecting Steiner tree and related problems, 2016.
  `www.optimization-online.org/DB_HTML/2016/06/5509.html`

Forthcoming: **PhD Thesis of Martin Luipersbeck, University of Vienna**

Why Studying Steiner Trees?

Wide range of applications:

- design of infrastructure networks (e.g., telecommunications), network optimization
- routing in communication networks
- handwriting recognition, image/3D movements recognition (machine learning)
- reconstruction of phylogenetic trees
- bioinformatics (analysis of protein-protein interaction networks)

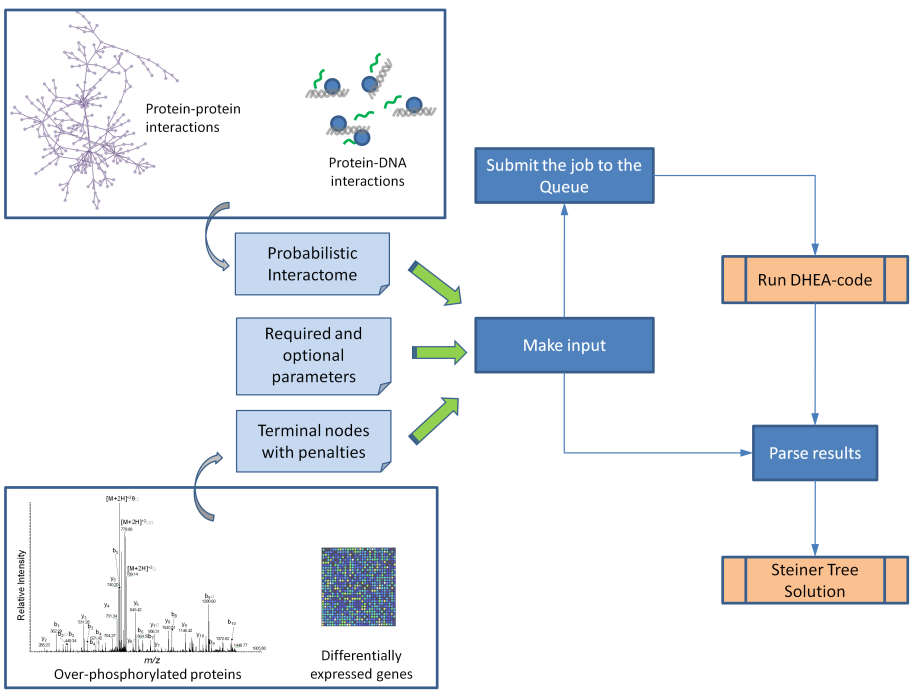


Figure borrowed from The Fraenkel Lab, MIT

Our work was motivated by:

DIMACS ICERM

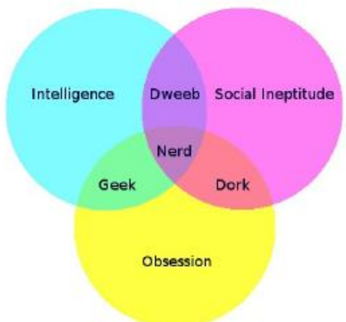
11th DIMACS Implementation Challenge in Collaboration with ICERM:
Steiner Tree Problems

Co-sponsored by
DIMACS, the DIMACS Special Focus on Information Sharing and Dynamic Data Analysis, and by the Institute for Computational and Experimental Research in Mathematics (ICERM)

From the web-site `dimacs11.zib.de/`

DIMACS Implementation Challenges address questions of determining realistic algorithm performance where worst case analysis is overly pessimistic and probabilistic models are too unrealistic: experimentation can provide guides to realistic algorithm performance where analysis fails."

We submitted codes: staynerd (['ʃtʌɪnə]) and mozartballs to the DIMACS Challenge



Exact Challenge, 1 Thread

	Gap		Time	
Class	Formula 1	Average	Formula 1	Average
SPG	mozartballs	mozartballs	mozartballs	mozartballs
RPCST	mozartballs scipjack scipjackspx	mozartballs scipjack scipjackspx	scipjack	scipjack
PCSPG	mozartballs	mozartballs	mozartballs	mozartballs
DCST	mozartballs	mozartballs	mozartballs	mozartballs
MWCS	mozartballs	mozartballs	heinz-no-dc	mozartballs

Exact Challenge, 8 Threads

	Gap		Time	
Class	Formula 1	Average	Formula 1	Average
SPG	mozartballs	mozartduet	mozartballs	mozartballs
RPCST	fscipjack fscipjackspx mozartballs	fscipjack fscipjackspx mozartballs	fscipjack	fscipjack
PCSPG	mozartballs	mozartduet	mozartballs	mozartballs
DCST	mozartballs	mozartballs	mozartballs	mozartballs
MWCS	mozartballs	mozartballs	heinz-no-dc	mozartballs

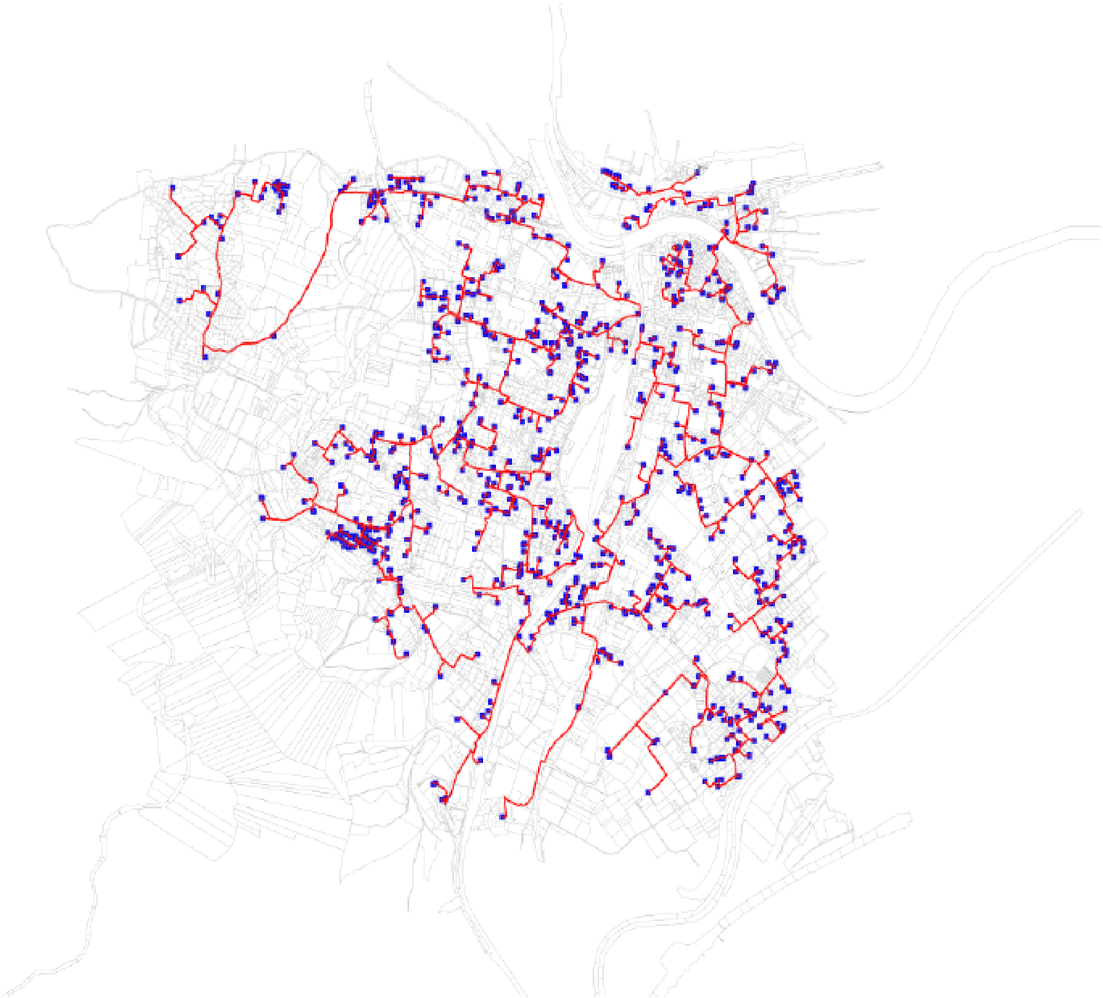
Heuristic Challenge, 1 Thread

	Primal Bound		Primal Integral	
Class	Formula 1	Average	Formula 1	Average
SPG	PUW	mozartballs	PUW	staynerd
RPCST	KTS mozartballs scipjack scipjackspx	KTS mozartballs scipjack scipjackspx	KTS	KTS
PCSPG	staynerd	staynerd	KTS	mozartballs
HCDST	stephop-ls4	stephop-ls4	stephop-ls4	stephop-ls4
DCST	mozartballs	scipjack	mozartballs	mozartballs
STPRBH	viennaNodehopper	viennaNodehopper	viennaNodehopper	viennaNodehopper
MWCS	mozartballs	mozartballs	mozartballs	mozartballs

Outline

1. Basic ILP Model(s) for (PC) Steiner Trees
2. A node-based model for (almost) uniform edge-costs (DIMACS Results)
3. A new branch-and-bound framework (dual ascent approach)

Steiner Trees



Steiner Trees

Definition (Steiner Tree Problem on a Graph (STP))

We are given an undirected graph $G = (V, E)$ with edge weights $c_e \geq 0$, $\forall e \in E$. The node set V is partitioned into required terminal nodes T_r and potential Steiner nodes S, i.e. $S \cup T_r = V$, $S \cap T_r = \emptyset$. The problem is to find a minimum weight subtree $G' = (V', E')$ of G that contains all terminal nodes, i.e., such that:

1. E' is a subtree
2. $T_r \subset V'$ and
3. $\sum_{e \in E'} c_e$ is minimal

Special cases: shortest path, MST

Prize Collecting STP

Definition (Prize Collecting STP (PCSTP))

We are given an undirected graph $G = (V, E)$ with edge weights $c_e \geq 0$, $\forall e \in E$, and node profits $p_i \geq 0$, $\forall i \in V$. The problem is to find a subtree $G' = (V', E')$ of G that yields maximum profit, i.e.

$$\max \sum_{i \in V'} p_i - \sum_{e \in E'} c_e.$$

Equivalently:

$$\min \sum_{e \in E'} c_e + \sum_{i \notin V'} p_i.$$

Remark: For a subtree (V', E') we have:

$$\sum_{i \in V'} p_i - \sum_{e \in E'} c_e = -(\sum_{e \in E'} c_e + \sum_{i \notin V'} p_i) + \sum_{i \in V} p_i$$

PCSTP: Example

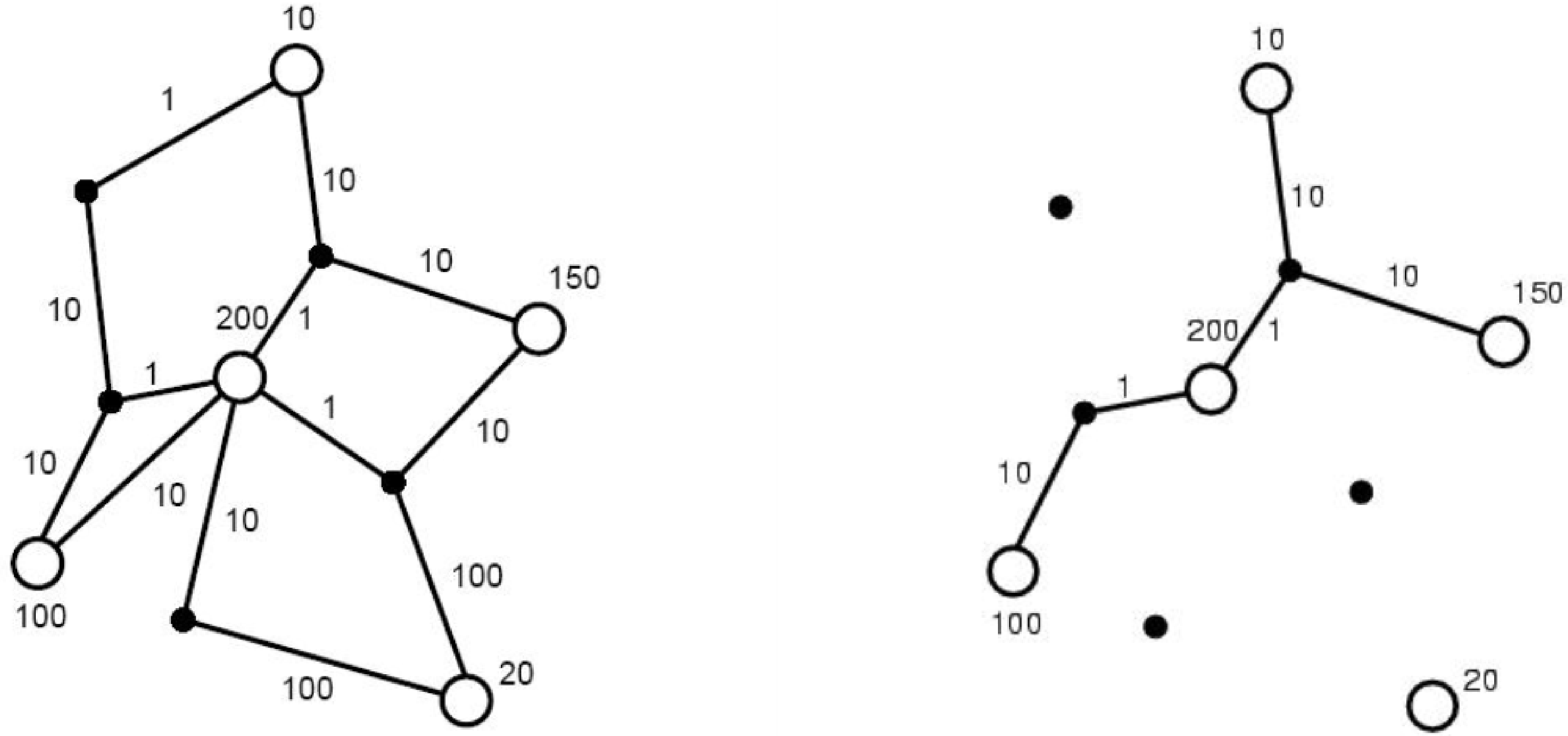


Figure : Input graph and a feasible PCSTP solution

Let us focus on PCSTP

- **Assume a root node r is given**
- let T_p be the set of potential terminals: only those with revenues $p_i > 0$ such that at least one adjacent edge is strictly cheaper than p_i (only they among nodes not in T_r can be potential leaves).

$$T_p = \{v \in V \setminus \{r\} \mid \exists \{u, v\} \text{ s.t. } c_{uv} < p_v\}.$$

  Recall: T_r is the set of **required terminals**. Together $T = T_r \cup T_p$.
- Transform instance into directed instance $G = (V, A)$ by creating two arcs (i, j), (j, i) for every edge $\{i, j\} \in E$
- Incorporate node-weights into arc costs:

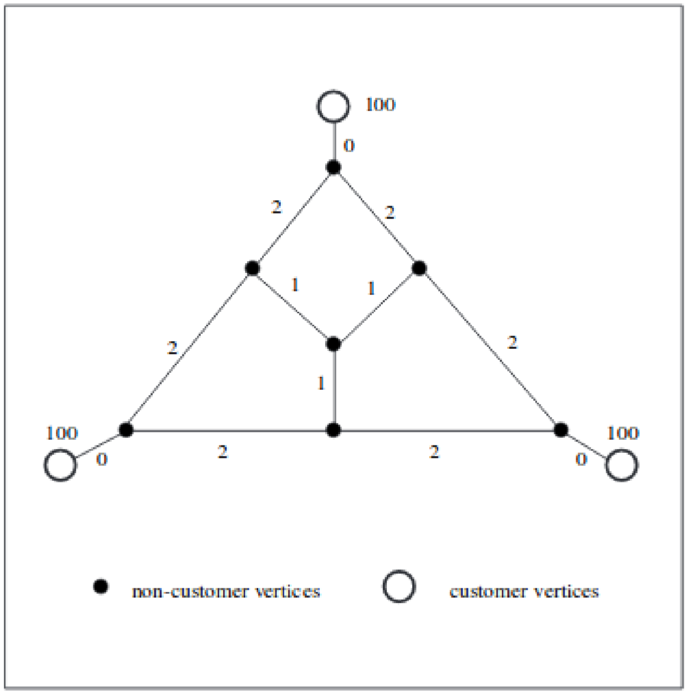
$$c'_{ij} := c_{ij} - p_j$$

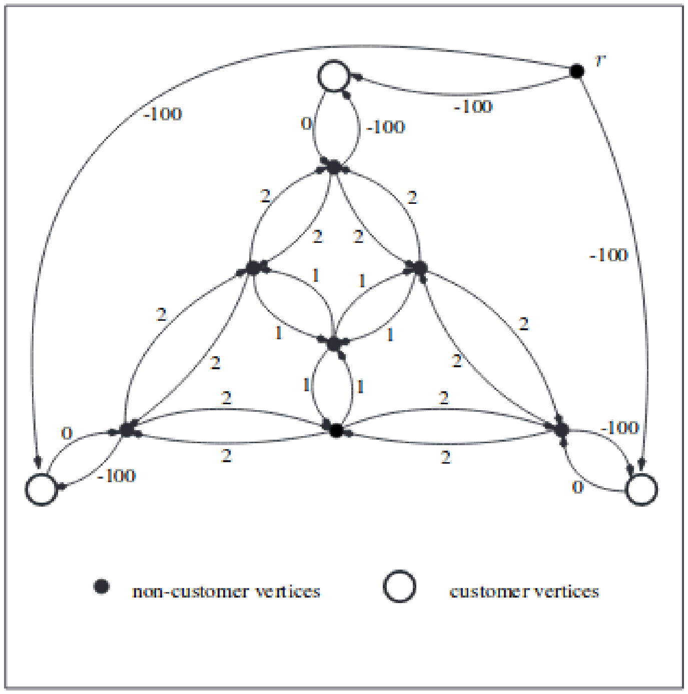
- Wlog: remove arcs entering the root.

Min-Cost Steiner Arborescence

After the transformation:

Every feasible solution is a rooted Steiner arborescence, i.e., from the root r to any node i in the solution, there exists a directed r-i path and the in-degree of each node is at most one.





ILP Models for PCSTP

Decision variables:

$$x_{ij} = \begin{cases} 1, & \text{iff arc } (i,j) \text{ is in solution} \\ 0. & \text{otherwise} \end{cases} \qquad \forall (i,j) \in A$$

$$y_i = \begin{cases} 1, & \text{iff node } i \text{ is in solution} \\ 0. & \text{otherwise} \end{cases} \qquad \forall i \in T$$

To model connectivity:

- flow models (single-commodity, multi-commodity, common-flow, etc)
- MTZ-like constraints,
- generalized subtour elimination constraints, or
- cut-set inequalities.

(x, y)-Model for PCSTP

Directed Cut Model:

$$
\begin{aligned}
\min \sum_{ij \in A} c'_{ij} x_{ij} + \sum_{i \in V} p_i & & \\
\text{s.t. } x(\delta^-(W)) \geq y_i & \quad \forall W \subset V, r \notin W, \forall i \in W \cap T & (1) \\
x(\delta^-(i)) = y_i & \quad \forall i \in T & \\
y_i = 1 & \quad \forall i \in T_r & \\
y_i \in \{0, 1\} & \quad \forall i \in T_p & \\
x_{ij} \in \{0, 1\} & \quad \forall (i, j) \in A &
\end{aligned}
$$

- *incoming cut-set* $\delta^-(W) = \{(i, j) \in A \mid i \notin W, j \in W\}$
- (1): directed Steiner cuts
- separate them in a cutting-plane fashion using max-flow
- Branch-and-cut from Ljubić et al. (2006) has been state-of-the-art for PCSTP until DIMACS (integrated in bioinformatics packages: SteinerNet, HEINZ...)

A node-based model for (almost) uniform edge-costs (DIMACS Results)

Why is PCSTP with uniform edge-costs relevant?

PCSTP with Uniform Edge-Costs

In instances from bioinformatics and machine learning, edges represent a relation between nodes, i.e., they either exist or not, there are no different edge weights. So we have

$$c_{ij} = c, \quad \forall (i,j) \in A.$$

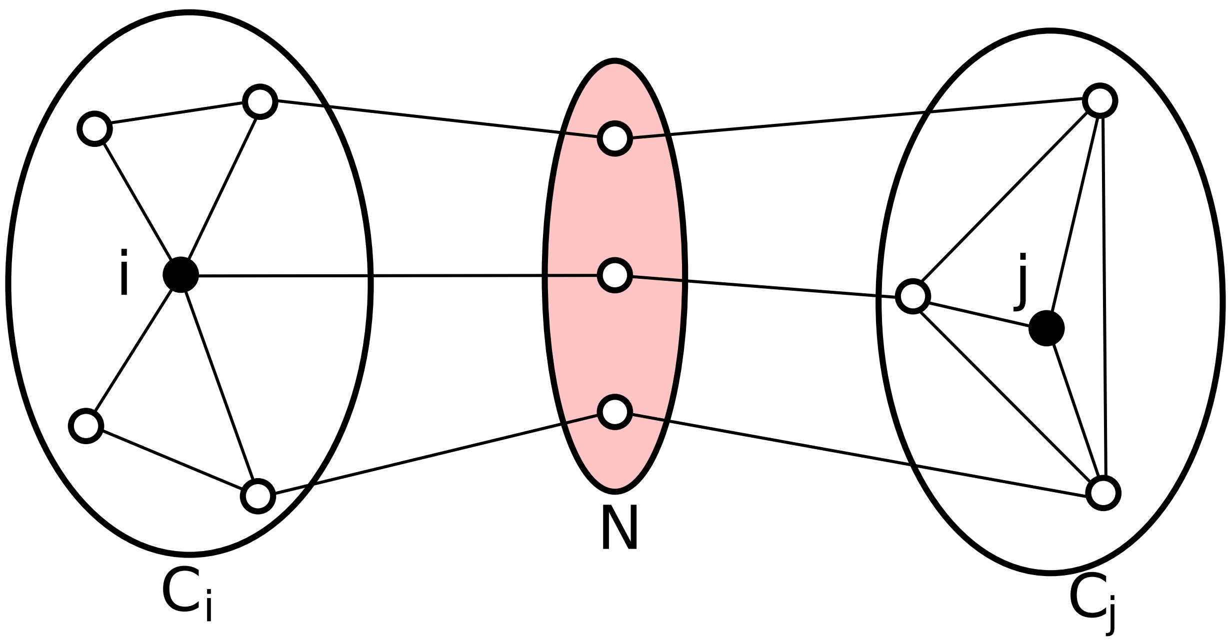
- Can we explot this fact in a different way?
- Can we "thin-out" the existing models in order to approach more challenging instances?
- Besides, among the most challenging DIMACS instances, most of them are with uniform edge-costs (PUC instances).

Outline

Node-based MIP model - Node separators

Definition (Node Separators)

For $i, j \in V$, a subset $N \subseteq V \setminus \{i, j\}$ is called (i, j) **node separator** iff after eliminating N from V there is no (i, j) path in G.
N is a **minimal node separator** if $N \setminus \{i\}$ is not a (i, j) separator, for any $i \in N$. Let $\mathcal{N}(i, j)$ denote the family of all (i, j) separators.



Node-based MIP model

Shift uniform edge costs c into node revenue:

$$\tilde{c}_v = c - p_v, \qquad \forall v \in V$$

Let

$$T = T_r \cup T_p \qquad\qquad P = \sum_{v \in V} p_v$$

$$
\begin{aligned}
\min \quad & \sum_{v \in V} \tilde{c}_v y_v + (P - c) & & (2)\\
\text{s.t.} \quad & y(N) \geq y_i + y_j - 1 & \forall i, j \in T, i \neq j, \forall N \in \mathcal{N}(i,j) \quad & (3)\\
& y_v = 1 & \forall v \in T_r \quad & (4)\\
& y_v \in \{0,1\} & \forall v \in V \setminus T_r \quad & (5)
\end{aligned}
$$

where $y(N) = \sum_{v \in N} y_v$.

Node-based MIP model - Lazy-Cut Separation

Algorithm

Data: **infeasible solution** defined by a vector $\tilde{y} \in \{0,1\}^n$ with $\tilde{y}_i = \tilde{y}_j = 1$, C_i being the connected component of $G_{\tilde{y}}$ containing i, and $j \notin C_i$. Let $Neigh(C_i)$ be neighboring nodes of C_i.

Result: **minimal node separator** N that violates inequality (3) with respect to i, j.

Delete all edges in $E[C_i \cup Neigh(C_i)]$ from G
Find the set R_j of nodes that can be reached from j
Return $N = Neigh(C_i) \cap R_j$

This separation runs in linear time. To separate fractional points, one would need to calculate max-flows in a transformed graph.

Node-based MIP model - Valid inequalities

- Node-degree inequalities:

$$y(A_i) \geq \begin{cases} y_i, & \text{if } i \in T \\ 2y_i, & \text{otherwise} \end{cases}$$

- 2-Cycle inequalities:

$$y_i \leq y_j \quad i \in V, j \in T_p, c_{ij} < p_j$$

Outline

Benders-like (set covering) heuristic

- node-based model can be interpreted as **set covering problem**
- connectivity constraints for pure Steiner tree problem ($T = T_r$) take the following form:

$$y(N) \geq 1, \quad \forall N \in \mathcal{N}$$

where $\mathcal{N}$ is the family of all node separators between arbitrary real terminal pairs.

$\rightarrow$ exploit this property by using a set covering heuristic to generate high-quality solutions

Benders-like (set covering) heuristic

Heuristic

1. Extract set covering relaxation of the current model
2. Solve relaxation heuristically
3. Repair: fix the nodes from the solution and solve the ILP model
4. Refine the model through generated node-separator cuts and repeat

- We employed set covering heuristic from Caprara et al. (1996)

Benders-like (set covering) heuristic

- Cutpool:
  - Add cuts also to set cover relaxation
  - Allows iteration to generate better solutions
- Diversification:
  - random shuffle of rows and columns
  - choose randomly only 80% of variables to fix
- Application to non-uniform instances:
  - shift edge non-uniform costs into node revenue:
  - "Blurred" version of the original problem

$$p_i = \frac{1}{|\delta(i)|} \sum_{e \in \delta(i)} c_e \qquad \forall i \in V \setminus T$$

Outline

Overall Algorithmic Framework

Data: input graph G, instance of the STP/PCSTP/DCSTP/MWCS, iteration and time limits.
Result: (sub)-optimal solution *Sol*.
$\mathcal{S}_{\text{init}}$ =**InitializationHeuristics**()
$k = 1$, $CutPool = \emptyset$
Choose *Sol* from the solution pool $\mathcal{S}_{\text{init}}$.
while *($k \leq maxLBiter$) and (time limit not exceeded)* **do**
    $(Sol, CutPool)$ = **LocalBranching**$(Sol, CutPool, seed)$
    $k = k + 1$
    Choose *Sol* from the solution pool $\mathcal{S}_{\text{init}}$. Change *seed*.
end
Sol = **BranchAndCut**$(CutPool, Sol, TimeLim)$
return *Sol*

Overall Algorithmic Framework

- **Branch & Cut** (B&C)
  - Node-based model (y-model)
  - Classic arc/node-based model ((x, y)-model)
    (Koch and Martin, 1998; Ljubić et al., 2006)
- B&C used as **black-box solver** in various heuristics
  - Benders-like heuristic
  - Local branching (Fischetti and Lodi, 2003)
  - Partitioning-based construction heuristic (Leitner et al., 2014)
- State-of-the-art **dual & primal heuristics**
  - Shortest path construction heuristic
    (de Aragão, Uchoa, and Werneck, 2001)
  - Local search: Keypath-exchange, Keynode-removal, Node-insertion
    (Uchoa and Werneck, 2010)
  - Dual ascent heuristic
    (Wong, 1984)

Local branching

- large-neighborhood exploration using **B&C as black-box solver**
- neighborhood defined by **local branching constraint**
- Given solution Sol, let $W_1 = \{v \in V \mid v \in Sol\}$ and $W_0 = V \setminus W_1$.
  - **Symmetric** local branching constraint

$$\sum_{v \in W_0} y_v + \sum_{v \in W_1} (1 - y_v) \leq r$$

  - **Asymmetric** local branching constraint

$$\sum_{v \in W_1} (1 - y_v) \leq r$$

One problem - different flavors!

y OR (x, y) MODEL??

Instance filtering

- goal: solve **hard** instances well, but also still provide **good average performance**
- approx. 1500 (diverse) instances (STP, PCSTP, MWCS, DCSTP)
- method: match algorithmic configuration to instance features

  `uniform`, **`sparse`**, **`dense`**, **`ratioT`**, **`bipartite`**, **`large`**, ...

- involved decisions:
  - model selection (node-based or arc/node-based model)
  - separation of inequalities (deal with tailing-off behavior)
  - estimate when to apply problem-specific heuristics

Filter rules

Model Selection	
`uniform`	$\rightarrow$ y-model
$\neg$`uniform`	$\rightarrow$ (x, y)-model
`uniform` $\wedge$ `sparse` $\wedge$ `ratioT` < 0.1	$\rightarrow$ (x, y)-model

(x, y)-model Settings	
`dense`	$\rightarrow$ use tailing-off bound, high tolerance
`verydense`	$\rightarrow$ use tailing-off bound, low tolerance
`ratioT` < 0.01	$\rightarrow$ add dual ascent connectivity cuts as violated
`ratioT` ≥ 0.01	$\rightarrow$ init with full set of dual ascent c. cuts
`ratioT` < 0.1 $\wedge$ `sparse` $\wedge$ `big`	$\rightarrow$ separate flow-balance, GSECs of size 2

Heuristic Settings & Preprocessing	
`bipartite` $\wedge$ `uniform`	$\rightarrow$ benders-like heuristic
`bipartite` $\wedge$ $\neg$`uniform` $\wedge$ `stp`	$\rightarrow$ benders-like heuristic (blurred)
`hypercube` $\wedge$ $\neg$`uniform` $\wedge$ $\neg$`small` $\wedge$ `pcstp`	$\rightarrow$ benders-like heuristic (blurred)
$\neg$`bipartite`	$\rightarrow$ local branching
`xy-model` $\wedge$ `big` $\wedge$ `sparse`	$\rightarrow$ partition-based heuristic
`weightRange` < 10	$\rightarrow$ allow non-improving moves during local search
`verydense`	$\rightarrow$ preprocessing (special distance test)

Outline

1. Node-based MIP model for uniform instances
2. Benders-like (set covering) heuristic
3. Overall Algorithmic Framework
4. Computational results

Computational Results

- Implementation in C++ and CPLEX 12.6
- Experiments performed in parallel on 4 cores (2.3GHz, 16GB RAM)
- 4 variants submitted at the DIMACS challenge:

"Mozart Duet"

#MozartBalls	exact, single & multi-threaded	STP, (R)PCSTP, MWCS, DCSTP
#StayNerd*	heuristic, single & multi-threaded	STP, PCSTP

#MozartDuet	multi-threaded 1 thread exact, others heuristic	STP, PCSTP
#HedgeKiller	multi-threaded 50% exact – 50% heuristic	STP, PCSTP

#MozartDuet	multi-threaded 1 thread exact, others heuristic	STP, PCSTP

Exact results for STP and PCSTP

Instance	$\|V\|$	$\|E\|$	$\|T\|$	y-model OPT	y-model Time (s.)	(x, y)-model ((*) out-of-memory) UB	LB	Gap	Time (s.)
s1	64	192	32	**10**	0.03	10	10	0.0%	0.01
s2	106	399	50	**73**	0.04	73	73	0.0%	1.36
s3	743	2947	344	**514**	0.15	514	505	1.78%	1090.61*
s4	5202	20783	2402	**3601**	1.31	3601	3523	2.21%	3444.81*
s5	36415	145635	16808	**25210**	22.28	25210	24056	4.80%	7200.00

Instance	$\|V\|$	$\|E\|$	$\|T\|$	OPT	y-model Time (s.) BEST	AVG	STD	(x, y)-model Time (s.) BEST	AVG	STD
w13c29	783	2262	406	**507** (508)	0.31	0.87	0.46	14.46	38.28	30.04
w23c23	1081	3174	552	**689** (694)	43.91	132.59	59.96	183.93	2600.15	1362.61

Instance	$\|V\|$	$\|E\|$	$\|T\|$	OPT	y-model Time (s.)	y-model Gap	(x, y)-model Time (s.)	(x, y)-model Gap
drosophila001	5226	93394	5226	8273.98263	7.98	**0.00**	86.12	**0.00**
drosophila005	5226	93394	5226	8121.313578	9.48	**0.00**	76.32	**0.00**
drosophila0075	5226	93394	5226	8039.859460	7.45	**0.00**	68.48	**0.00**
HCMV	3863	29293	3863	7371.536373	0.96	**0.00**	6.11	**0.00**
lymphoma	2034	7756	2034	3341.890237	0.28	**0.00**	1.24	**0.00**
metabol_expr_mice_1	3523	4345	3523	11346.927189	5965.76	**0.00**	1.08	**0.00**
metabol_expr_mice_2	3514	4332	3514	16250.235191	1.21	**0.00**	1.57	**0.00**
metabol_expr_mice_3	2853	3335	2853	16919.620407	4.00	**0.00**	0.89	**0.00**

Heuristic results for unsolved STP instances (SteinLib)

				BEST		AVG		STD		
Instance	$\|V\|$	$\|E\|$	$\|T\|$	UB	Time	UB	Time	UB	Time	Impr.*
bip52u	2200	7997	200	**233**	1390.10	233.80	287.94	0.42	597.96	1
bip62u	1200	10002	200	**219**	6.21	219.00	12.28	0.00	5.04	1
bipa2p	3300	18073	300	**35355**	547.18	35360.90	1342.88	4.38	879.59	24
bipa2u	3300	18073	300	**337**	185.06	337.00	310.89	0.00	215.22	4
hc10p	1024	5120	512	**59981**	267.51	60041.30	1013.51	33.38	816.95	513
hc10u	1024	5120	512	**575**	11.17	575.00	86.97	0.00	85.92	6
hc11p	2048	11264	1024	**119500**	3327.76	119533.00	1708.94	35.11	1129.07	279
hc11u	2048	11264	1024	**1145**	663.27	1145.40	1319.21	0.52	873.14	9
hc12p	4096	24576	2048	**236267**	2782.93	236347.10	2514.01	55.44	565.26	682
hc12u	4096	24576	2048	**2261**	2756.85	2262.50	2805.22	1.27	747.01	14
cc10-2p	1024	5120	135	**35257**	875.45	35353.20	704.89	75.12	705.21	122
cc11-2p	2048	11263	244	**63680**	744.33	63895.70	976.37	103.40	726.59	146
cc3-10p	1000	13500	50	**12784**	3471.19	12826.20	1801.62	43.46	1139.72	76
cc3-11p	1331	19965	61	**15599**	458.95	15633.30	812.14	35.44	965.08	10
cc3-12u	1728	28512	74	**185**	59.70	185.00	900.54	0.00	985.39	1
cc6-3p	729	4368	76	**20340**	1266.76	20395.90	1543.97	46.02	983.95	116
cc7-3p	2187	15308	222	**57080**	1385.54	57328.70	1197.71	153.94	888.00	8
cc7-3u	2187	15308	222	**551**	383.80	554.10	1267.21	1.52	1078.48	1
cc9-2p	512	2304	64	**17202**	1603.44	17274.40	1579.81	28.51	984.36	94
i640-312	640	4135	160	**35768**	1410.35	35793.20	1478.45	25.38	1104.32	3
i640-314	640	4135	160	**35533**	1610.03	35547.00	1673.70	12.53	679.53	5
i640-315	640	4135	160	**35720**	156.24	35733.50	866.76	21.87	695.92	21

(*) improved with respect to previously known best objective values

Conclusions

- Our work:
  - explored a **node-based model** for Steiner tree problems
  - exploited **symmetries** to our advantage
  - provided an **algorithmic framework** with local branching and **Benders-like heuristics**
  - handled both easy and hard instances
  - solved previously unsolved uniform instances **within seconds**
- At the end of the challenge, many new ideas and algorithms emerged (see forthcoming articles in Mathematical Programming Computation)
- The idea of thinning-out MIP models has been later successfully applied to Steiner trees with hop-constraints Sinnl and Ljubić (2016) or facility location problems Fischetti et al. (2016, 2017)

Literature I

A. Caprara, M. Fischetti, and P. Toth. A heuristic algorithm for the set covering problem. In *Integer Programming and Combinatorial Optimization*, pages 72–84. Springer Berlin Heidelberg, 1996.

M. P. de Aragão, E. Uchoa, and R. F. F. Werneck. Dual heuristics on the exact solution of large Steiner problems. *Electronic Notes in Discrete Mathematics*, 7:150–153, 2001.

M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98(1-3):23–47, 2003.

M. Fischetti, I. Ljubić, and M. Sinnl. Benders decomposition without separability: A computational study for capacitated facility location problems. *European Journal of Operational Research*, 253(3):557–569, 2016.

M. Fischetti, I. Ljubić, and M. Sinnl. Redesigning Benders Decomposition for Large Scale Facility Location. *Management Science*, 2017. to appear.

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

Literature II

M. Leitner, I. Ljubić, M. Luipersbeck, and M. Resch. A Partition-Based Heuristic for the Steiner Tree Problem in Large Graphs. In M. J. Blesa, C. Blum, and S. Voß, editors, *Hybrid Metaheuristics - Proceedings*, volume 8457 of *Lecture Notes in Computer Science*, pages 56–70. Springer, 2014.

I. Ljubić, R. Weiskircher, U. Pferschy, G. W. Klau, P. Mutzel, and M. Fischetti. An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem. *Mathematical Programming*, 105(2-3):427–449, 2006.

M. Sinnl and I. Ljubić. A node-based layered graph approach for the Steiner tree problem with revenues, budget and hop-constraints. *Math. Program. Comput.*, 8(4): 461–490, 2016.

E. Uchoa and R. F. Werneck. Fast local search for Steiner trees in graphs. In G. E. Blelloch and D. Halperin, editors, *ALENEX*, pages 1–10. SIAM, 2010.

R. T. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming*, 28(3):271–287, 1984. ISSN 0025-5610.

A dual-ascent-based branch-and-bound framework for PCSTP and related problems

Markus Leitner[1] Ivana Ljubić[2] Martin Luipersbeck[1] Markus Sinnl[1]

[1] University of Vienna, Department of Statistics and Operations Research, Vienna, Austria

[2] ESSEC Business School, Paris, France

April 21st, COMEX Workshop 2017

Outline

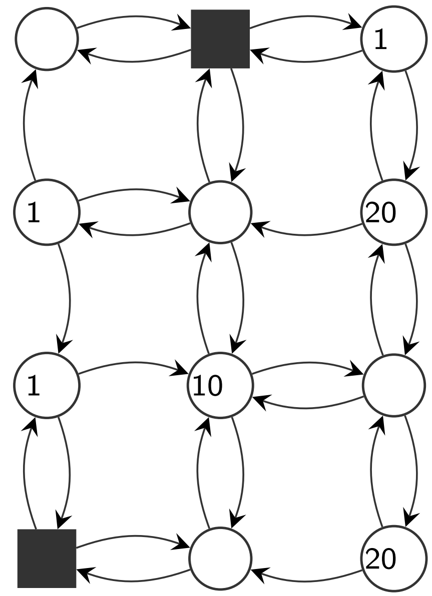
Asymmetric prize-collecting Steiner tree problem (APCSTP)

Definition

Given: digraph $G = (V, A)$, costs $c : A \mapsto \mathbb{R}_{\geq 0}$, prizes $p : V \mapsto \mathbb{R}_{\geq 0}$, fixed terminals $T_f \subset V$

Goal: find arborescence $S = (V_S, A_S) \subseteq G$ with $T_f \subseteq V_S$ and which minimizes

$$c(S) = \sum_{(i,j) \in A_S} c_{ij} + \sum_{i \notin V_S} p_i$$



$c_{ij} = 6 \quad \forall (i,j) \in A$

Potential terminals $T_p = \{i \in V \setminus T_f : p_i > 0\}$

Terminals $T = T_p \cup T_f$

Rooted APCSTP: fixed root $r \in T_f$

Generalizes several network design problems (directed *and* undirected)

Steiner tree/arborescence (STP/SAP), maximum-weight connected subgraph (MWCS), node-weighted Steiner tree (NWSTP), prize-collecting Steiner tree (PCSTP)

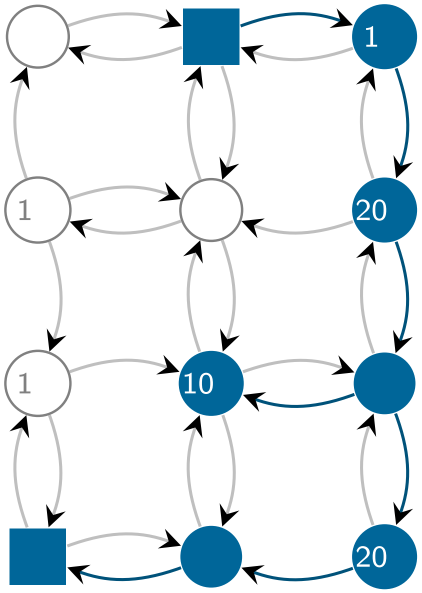
Asymmetric prize-collecting Steiner tree problem (APCSTP)

Definition

Given: digraph $G = (V, A)$, costs $c : A \mapsto \mathbb{R}_{\geq 0}$, prizes $p : V \mapsto \mathbb{R}_{\geq 0}$, fixed terminals $T_f \subset V$

Goal: find arborescence $S = (V_S, A_S) \subseteq G$ with $T_f \subseteq V_S$ and which minimizes

$$c(S) = \sum_{(i,j) \in A_S} c_{ij} + \sum_{i \notin V_S} p_i$$



$c_{ij} = 6 \quad \forall (i, j) \in A$

Potential terminals $T_p = \{i \in V \setminus T_f : p_i > 0\}$

Terminals $T = T_p \cup T_f$

Rooted APCSTP: fixed root $r \in T_f$

Generalizes several network design problems (directed *and* undirected)

Steiner tree/arborescence (STP/SAP), maximum-weight connected subgraph (MWCS), node-weighted Steiner tree (NWSTP), prize-collecting Steiner tree (PCSTP)

Dual ascent

- Solves the dual of an LP relaxation heuristically (usually very fast)
- Follows simple greedy strategy
- Outcome: a valid lower bound and a heuristic solution derived from the subgraph
  - update dual variables such that lower bound increases monotonically
  - preserve dual feasibility at each step

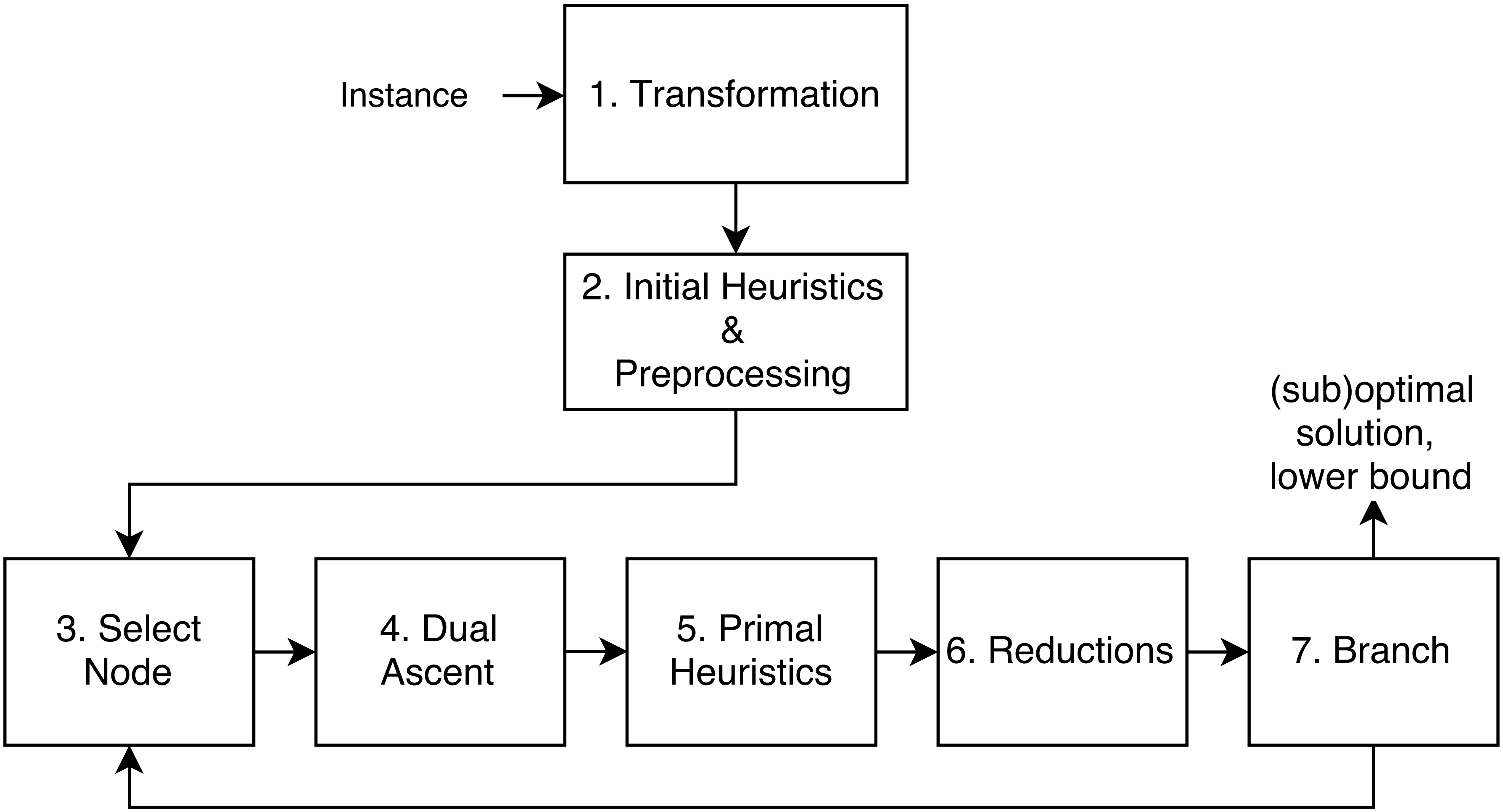
Previous & related works

- Dual ascent algorithm for the SAP (Wong, 1984)
- Used in various B&B frameworks for the STP (Polzin and Daneshmand, 2001; Pajor et al., 2014)

For the first time, dual ascent for APCSTP

- Generalizes Wong's dual ascent for the SAP

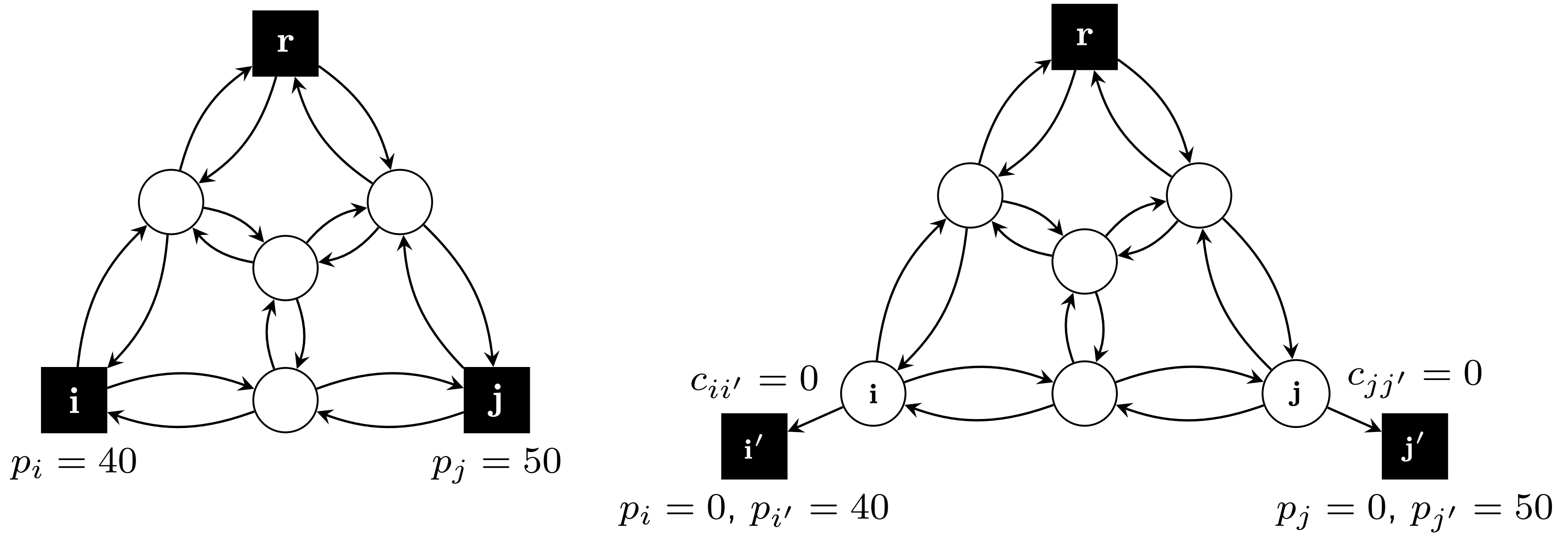
B&B framework - General structure (no MIP solver employed!)



Dual Ascent

Dual ascent - Transformation

Add artificial arcs and nodes, make each potential terminal a leaf node



(a) Original instance (b) Transformed instance

Dual ascent - LP relaxation

The following cut-based ILP formulation:

$$
\begin{aligned}
\text{(CUT)} \quad \min \quad & \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{i'\in T_p} (1 - x_{ii'})p_{i'} && && (1)\\
\text{s.t.} \quad & x(\delta^-(W)) \geq 1 && \forall W \in \mathcal{W}_f \quad (\beta_W) && (2)\\
& x(\delta^-(W)) \geq x_{ii'} && \forall i' \in W \cap T_p, W \in \mathcal{W}_p \quad (\beta'_W) && (3)\\
& x_{ii'} \leq 1 && \forall i' \in T_p \quad (\pi_{i'}) && (4)\\
& x_{ij} \geq 0 && \forall (i,j) \in A && (5)
\end{aligned}
$$

Node sets inducing Steiner cuts:

$$
\begin{aligned}
\mathcal{W}_f &= \{W \subset V : r \notin W, |W \cap T_p| = 0, |W \cap T_f| \geq 1\}\\
\mathcal{W}_p &= \{W \subset V : r \notin W, |W \cap T_p| = 1\}
\end{aligned}
$$

(2) ensure connectivity to each fixed terminal $i \in T_f$

(3) ensure connectivity to each potential terminal $i \in T_p$ if prize is collected

Dual ascent - Algorithm

$$
\text{(CUT-D)} \max \sum_{i \in T_p} (p_i - \pi_i) + \sum_{W \in \mathcal{W}_f} \beta_W \quad (6)
$$

$$
\text{s.t.} \sum_{\substack{W \in \mathcal{W}_p: \\ (i,j) \in \delta^-(W)}} \beta'_W + \sum_{\substack{W \in \mathcal{W}_f: \\ (i,j) \in \delta^-(W)}} \beta_W \leq c_{ij} \quad \forall (i,j) \in A, j \notin T_p \quad (7)
$$

$$
\pi_i + \sum_{\substack{W \in \mathcal{W}_p: \\ i \in W}} \beta'_W \geq p_i \quad \forall i \in T_p \quad (8)
$$

$$
(\boldsymbol{\beta}, \boldsymbol{\beta'}, \boldsymbol{\pi}) \in \mathbb{R}_{\geq 0}^{|\mathcal{W}_f| + |\mathcal{W}_p| + |T_p|} \quad (9)
$$

Ascent strategy:

Start with $\boldsymbol{\beta} = \boldsymbol{\beta'} = \mathbf{0}$, $\boldsymbol{\pi} = \boldsymbol{p}$.

Heuristically *choose* W and increase β_W or β'_W.

If β'_W is increased, decrease π_i by the same amount.

Repeat until *no increase possible*.

Dual ascent - Algorithm

Question: How should we choose W?

Reduced cost $\tilde{\mathbf{c}}$ for constraints (7)

$$\tilde{c}_{ij} = c_{ij} - \sum_{\substack{W \in \mathcal{W}_p: \\ (i,j) \in \delta^-(W)}} \beta'_W - \sum_{\substack{W \in \mathcal{W}_f: \\ (i,j) \in \delta^-(W)}} \beta_W \qquad \forall (i,j) \in A, j \notin T_p$$

Saturation graph G_S induced by $\{(i,j) \in A : \tilde{c}_{ij} = 0 \vee j \in T_p\}$

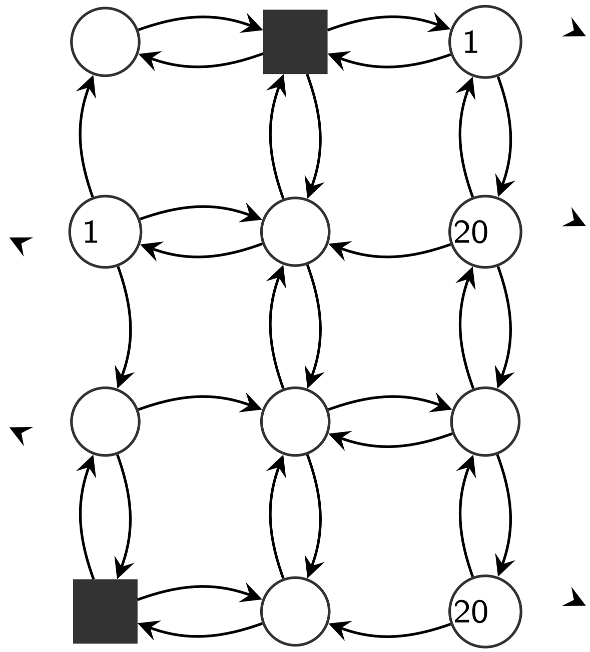
Active terminals are those not connected to the root in G_S and with $\pi_k \neq 0$:

$$T_a := \{k \in T \setminus \{r\} : \nexists P_{G_S}(r,k)\} \setminus \{k \in T_p : \pi_k = 0\}$$

Active component wrt to k contains all nodes reachable from k in G_S:

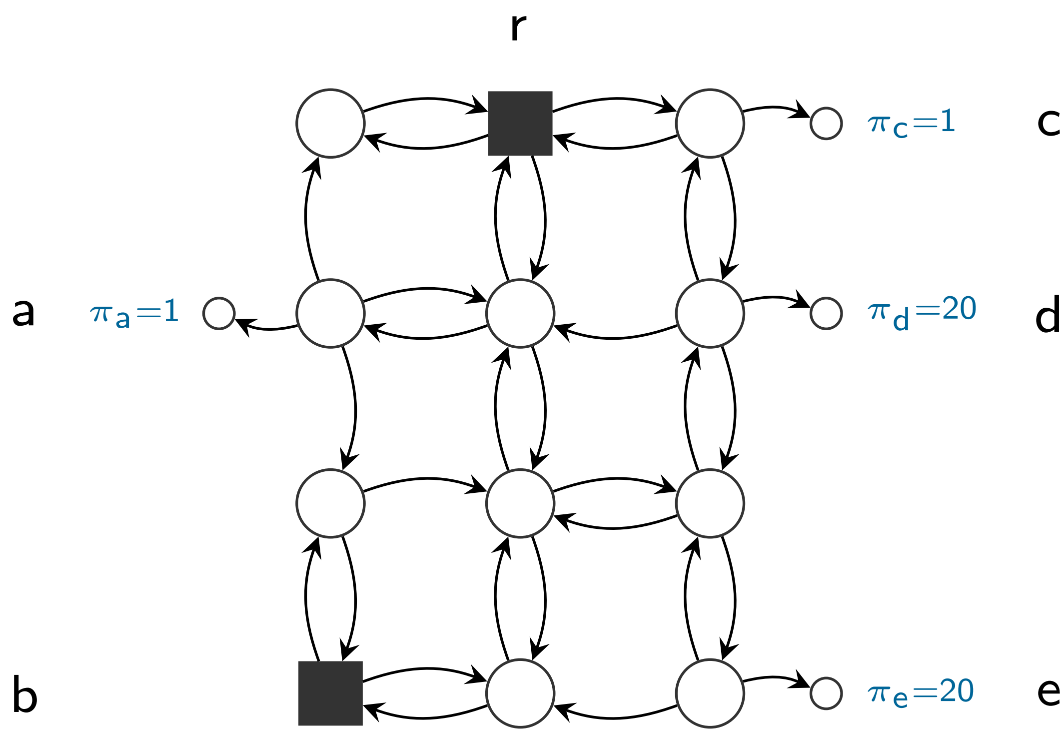
$$W(k) := \{i \in V : \exists P_{G_S}(i,k)\}$$

Dual ascent - Example



$c_{ij} = 6 \ \forall (i, j) \in A$

Dual ascent - Example

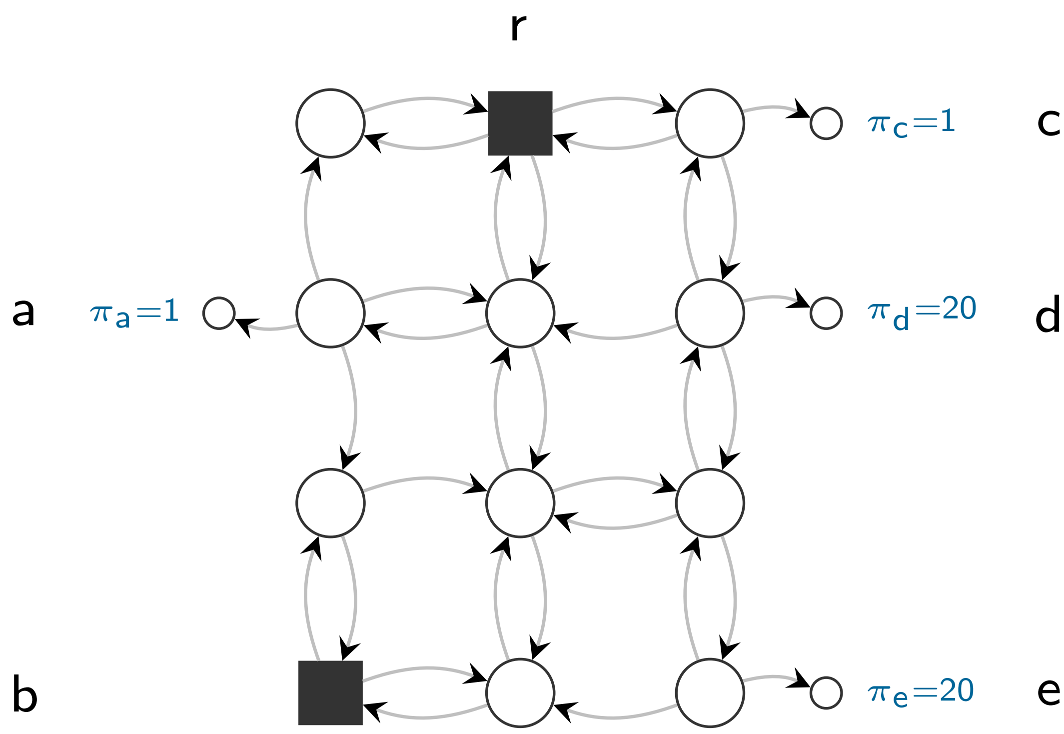


$$T_a = \{a, b, c, d, e\}$$
$$LB = 0,\ T_a = \{a, b, c, d, e\}$$

$c_{ij} = 6\ \forall (i, j) \in A$

Dual ascent - Example

$$T_a = \{a, b, c, d, e\}$$
$$LB = 0,\ T_a = \{a, b, c, d, e\}$$

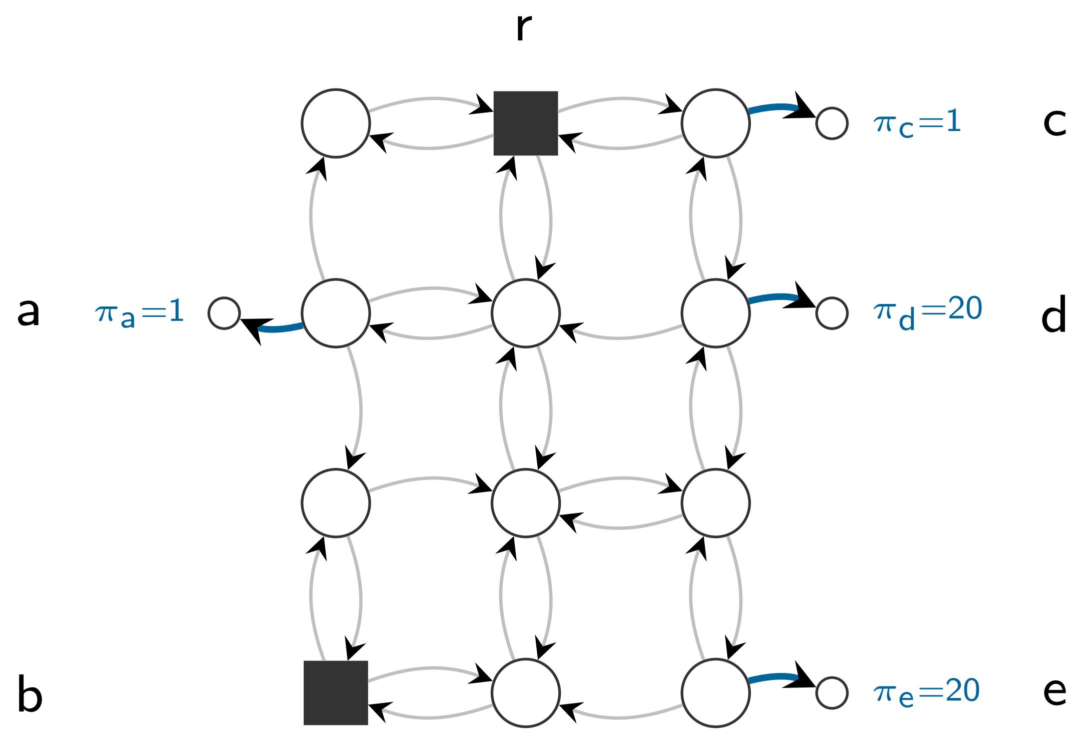


$c_{ij} = 6\ \forall (i, j) \in A$

Dual ascent - Example

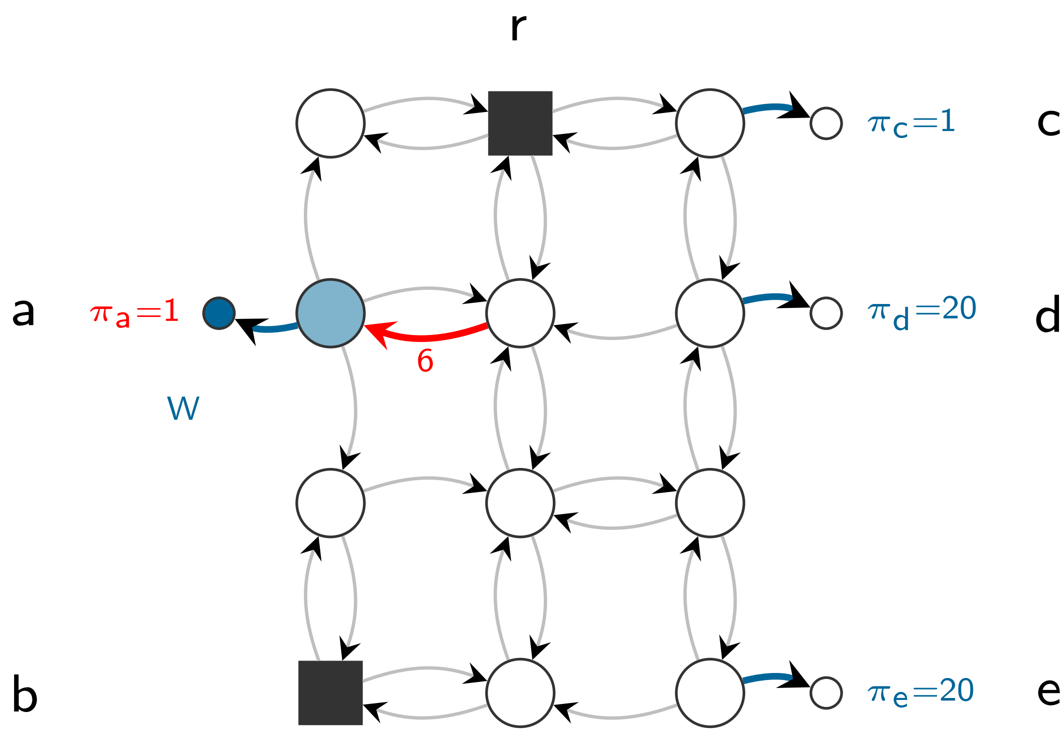
$T_a = \{a, b, c, d, e\}$

$LB = 0$, $T_a = \{a, b, c, d, e\}$



$c_{ij} = 6 \ \forall (i, j) \in A$

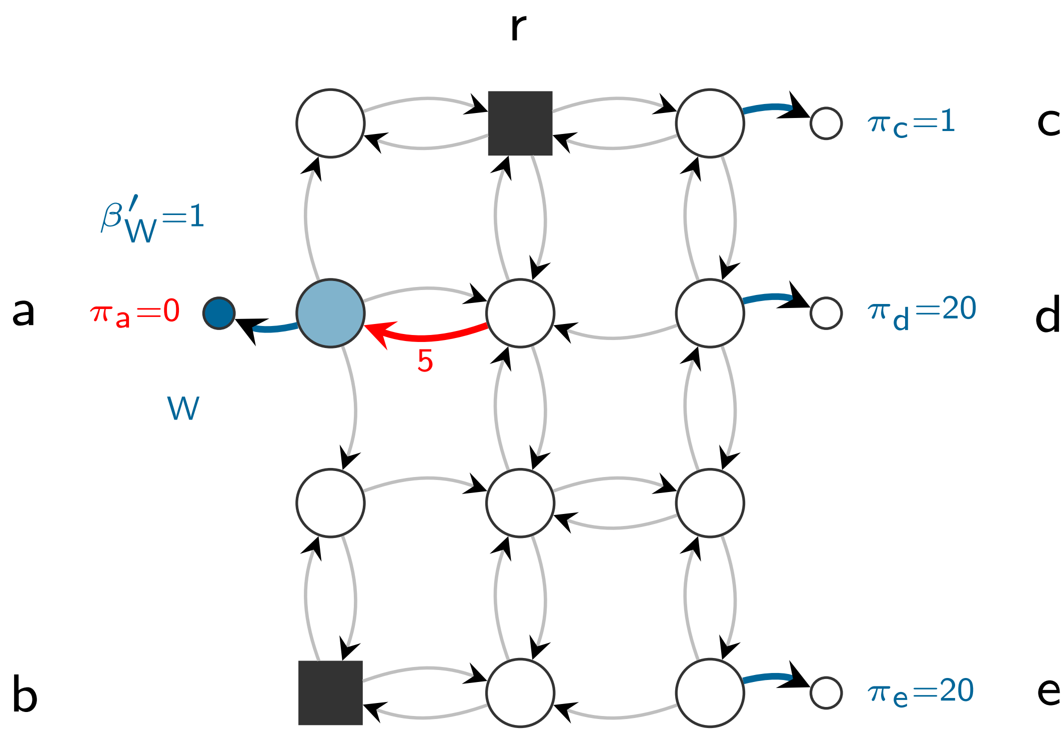
Dual ascent - Example



$$T_a = \{a, b, c, d, e\}$$
$$LB = 0,\ T_a = \{a, b, c, d, e\}$$

$c_{ij} = 6\ \forall (i, j) \in A$

Dual ascent - Example

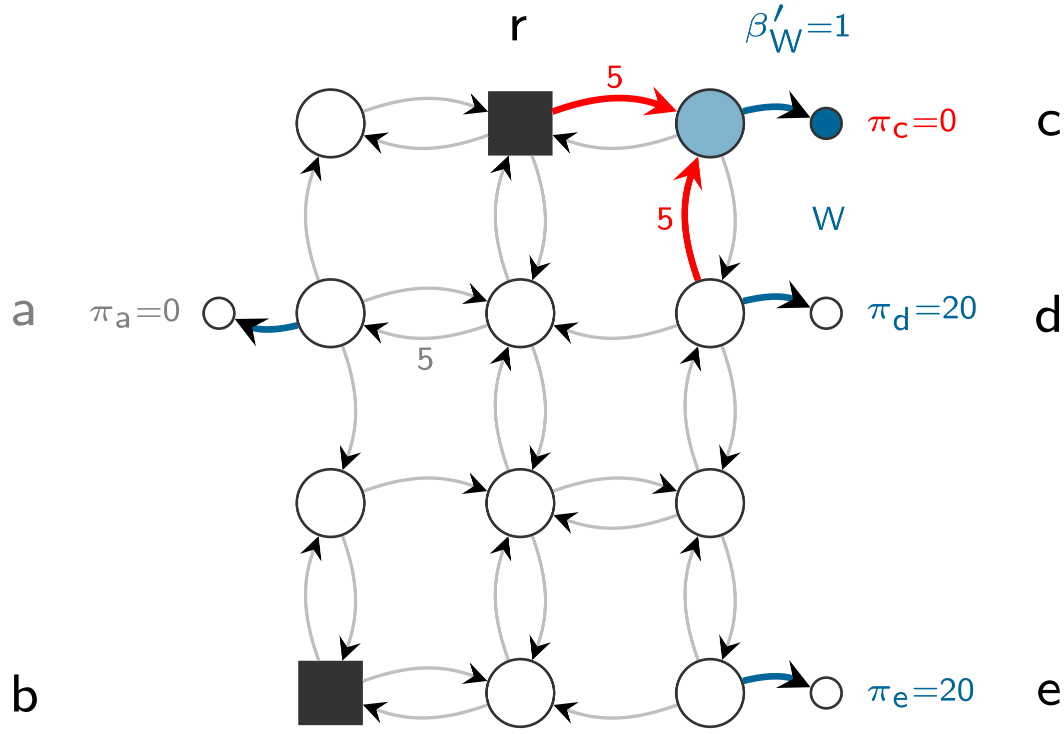


$$T_a = \{a, b, c, d, e\}$$
$$LB = 0,\ T_a = \{a, b, c, d, e\}$$
$$LB = 1,\ T_a = \{b, c, d, e\}$$

$$c_{ij} = 6\ \forall (i, j) \in A$$

Dual ascent - Example

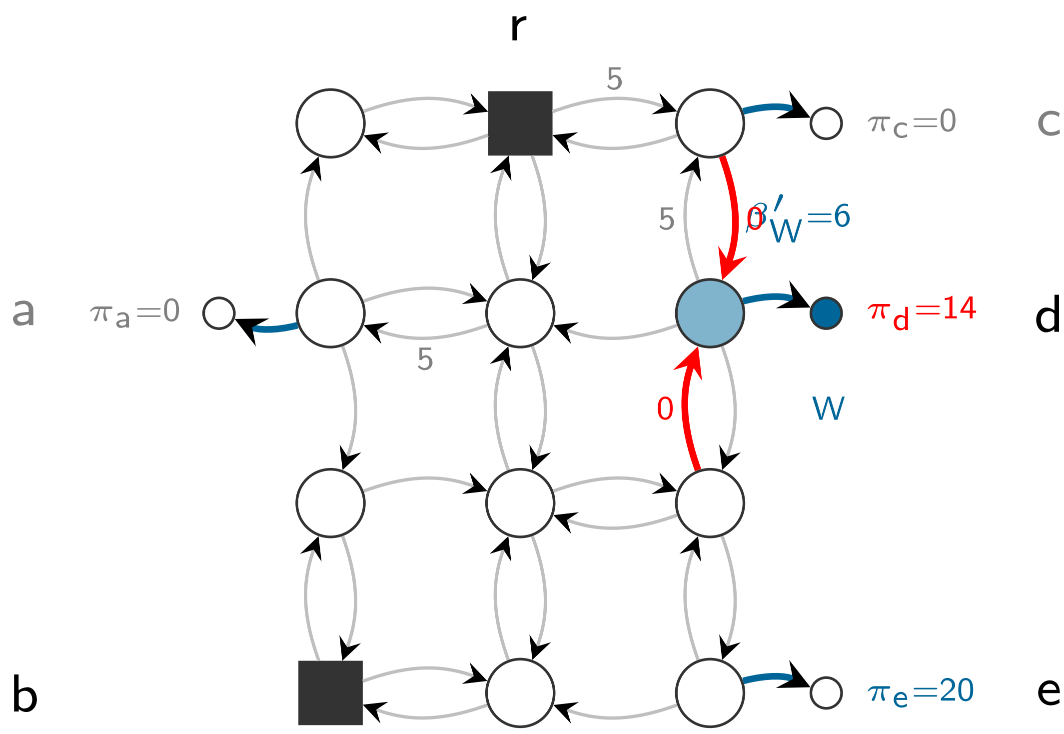
$$T_a = \{a, b, c, d, e\}$$
$$LB = 0,\ T_a = \{a, b, c, d, e\}$$
$$LB = 1,\ T_a = \{b, c, d, e\}$$
$$LB = 2,\ T_a = \{b, d, e\}$$



$$c_{ij} = 6\ \forall (i, j) \in A$$

Dual ascent - Example

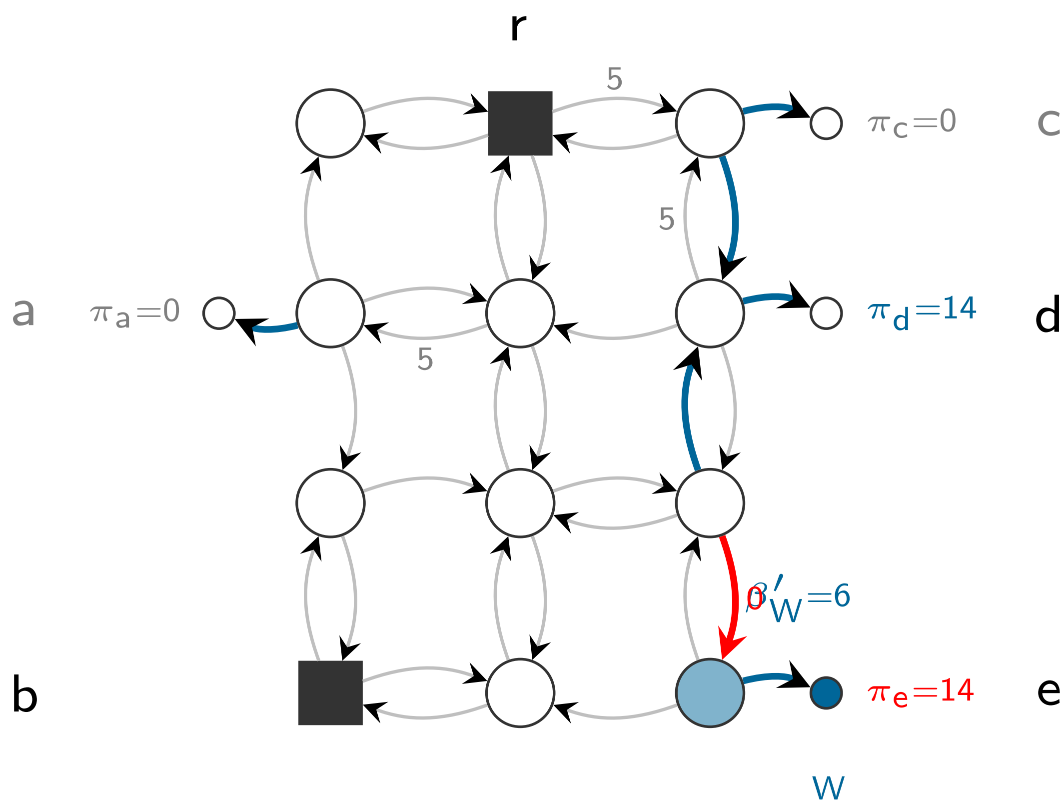
$$T_a = \{a, b, c, d, e\}$$
$$LB = 0,\ T_a = \{a, b, c, d, e\}$$
$$LB = 1,\ T_a = \{b, c, d, e\}$$
$$LB = 2,\ T_a = \{b, d, e\}$$
$$LB = 8,\ T_a = \{b, d, e\}$$



$c_{ij} = 6\ \forall (i,j) \in A$

Dual ascent - Example

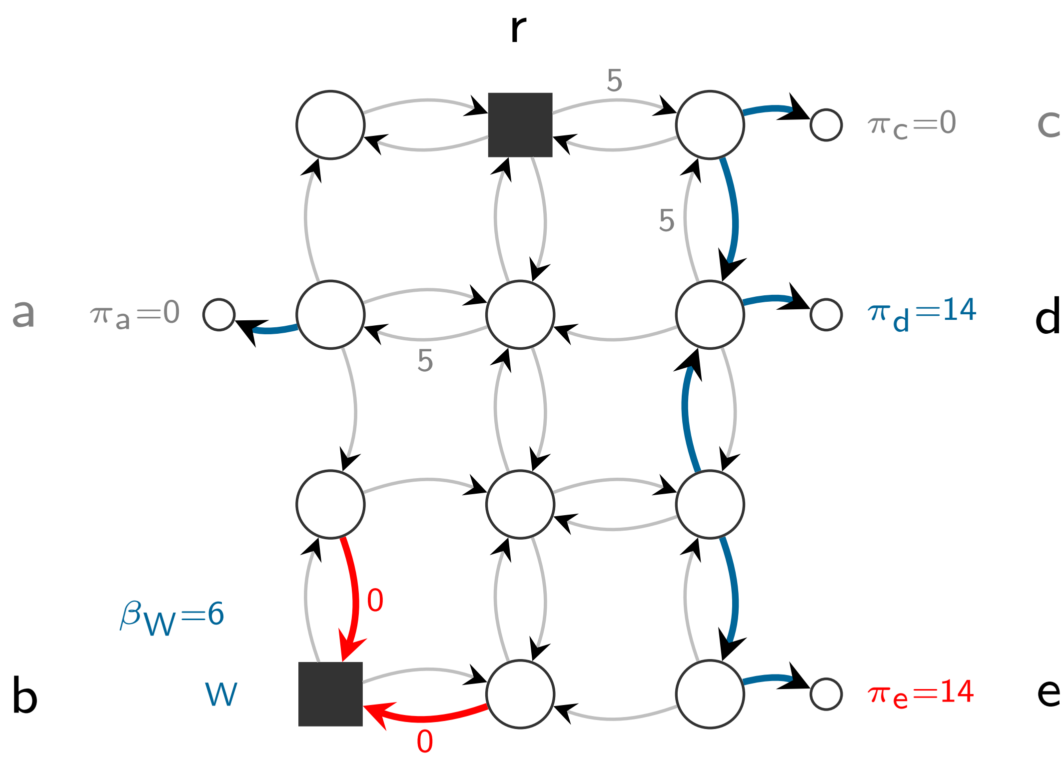
$$T_a = \{a, b, c, d, e\}$$
$$LB = 0,\ T_a = \{a, b, c, d, e\}$$
$$LB = 1,\ T_a = \{b, c, d, e\}$$
$$LB = 2,\ T_a = \{b, d, e\}$$
$$LB = 8,\ T_a = \{b, d, e\}$$
$$LB = 14,\ T_a = \{b, d, e\}$$



$c_{ij} = 6\ \forall (i, j) \in A$

Dual ascent - Example

$$T_a = \{a, b, c, d, e\}$$
$$LB = 0,\ T_a = \{a, b, c, d, e\}$$
$$LB = 1,\ T_a = \{b, c, d, e\}$$
$$LB = 2,\ T_a = \{b, d, e\}$$
$$LB = 8,\ T_a = \{b, d, e\}$$
$$LB = 14,\ T_a = \{b, d, e\}$$
$$LB = 20,\ T_a = \{b, d, e\}$$



$$c_{ij} = 6\ \forall (i, j) \in A$$

Dual ascent - Example

$$T_a = \{a, b, c, d, e\}$$

$LB = 0,\ T_a = \{a, b, c, d, e\}$

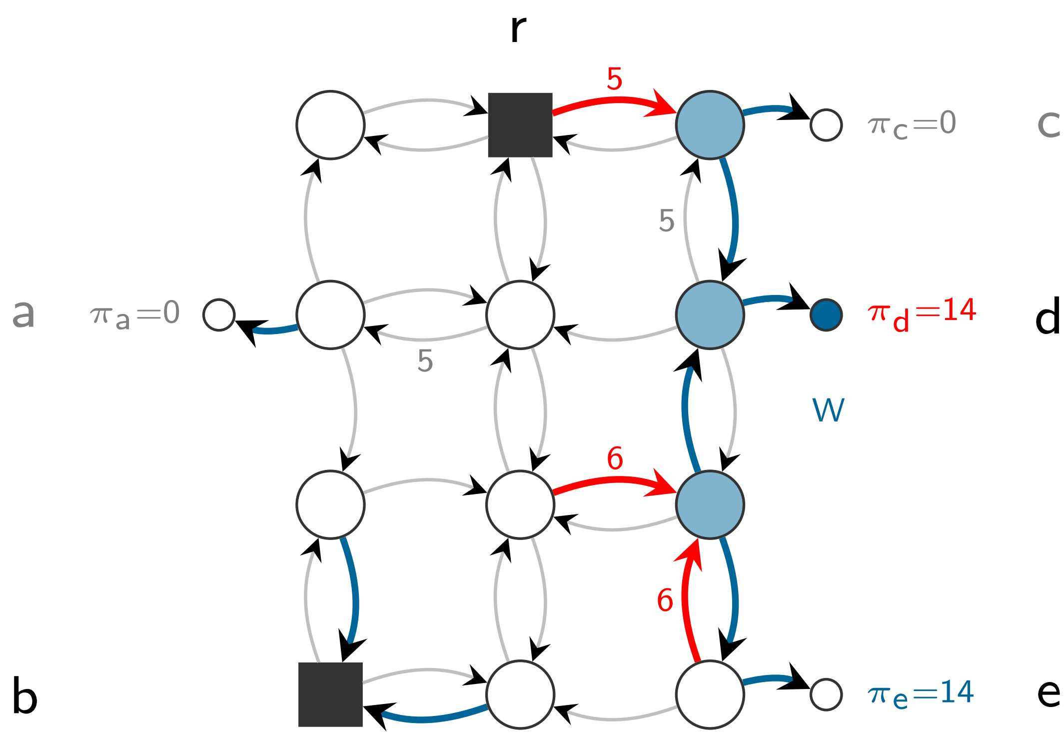
$LB = 1,\ T_a = \{b, c, d, e\}$

$LB = 2,\ T_a = \{b, d, e\}$

$LB = 8,\ T_a = \{b, d, e\}$

$LB = 14,\ T_a = \{b, d, e\}$

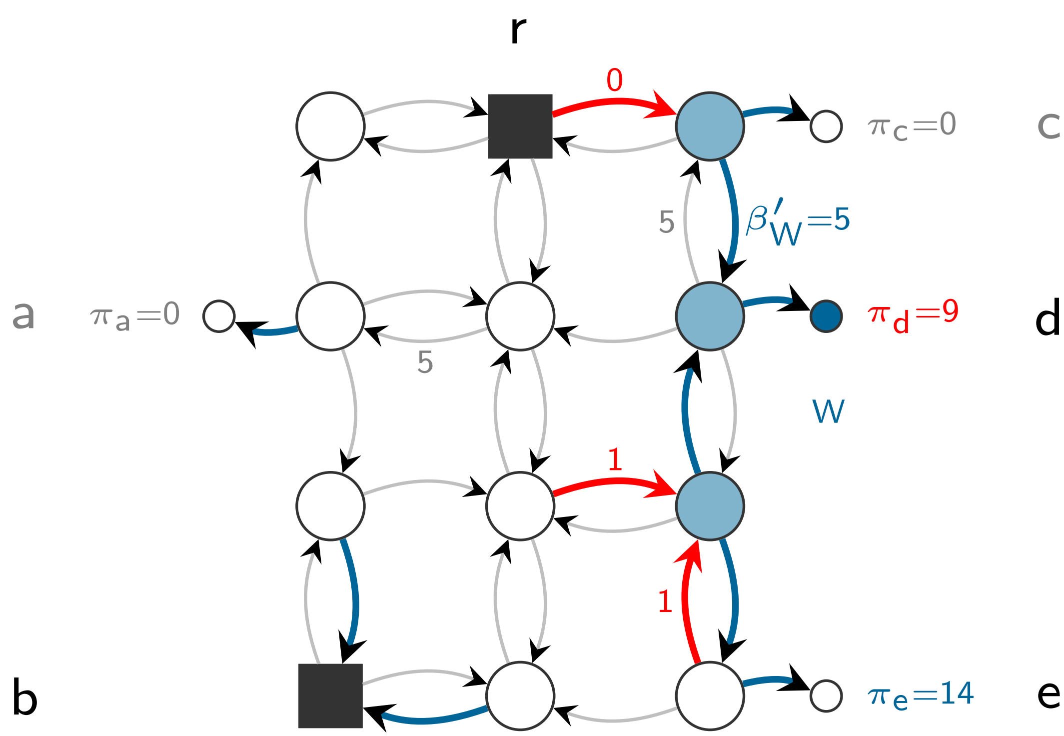
$LB = 20,\ T_a = \{b, d, e\}$



$$c_{ij} = 6\ \forall (i, j) \in A$$

Dual ascent - Example

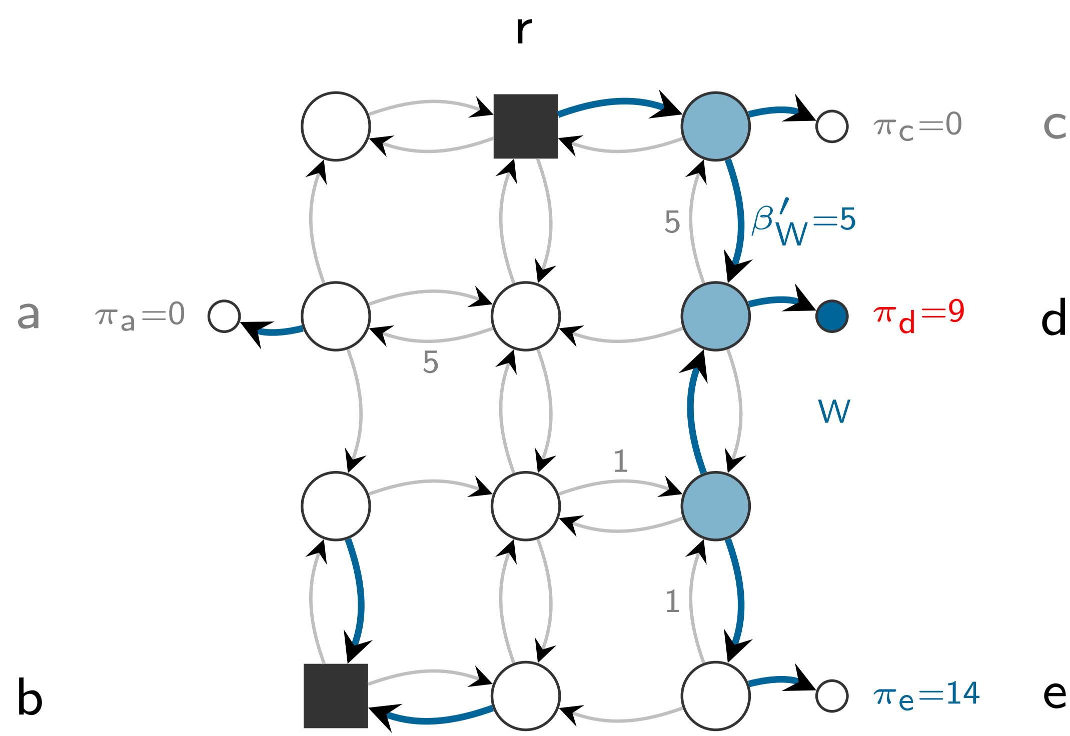
$$T_a = \{a, b, c, d, e\}$$
$$LB = 0,\ T_a = \{a, b, c, d, e\}$$
$$LB = 1,\ T_a = \{b, c, d, e\}$$
$$LB = 2,\ T_a = \{b, d, e\}$$
$$LB = 8,\ T_a = \{b, d, e\}$$
$$LB = 14,\ T_a = \{b, d, e\}$$
$$LB = 20,\ T_a = \{b, d, e\}$$
$$LB = 25,\ T_a = \{b, e\}$$



$c_{ij} = 6\ \forall (i,j) \in A$

Dual ascent - Example

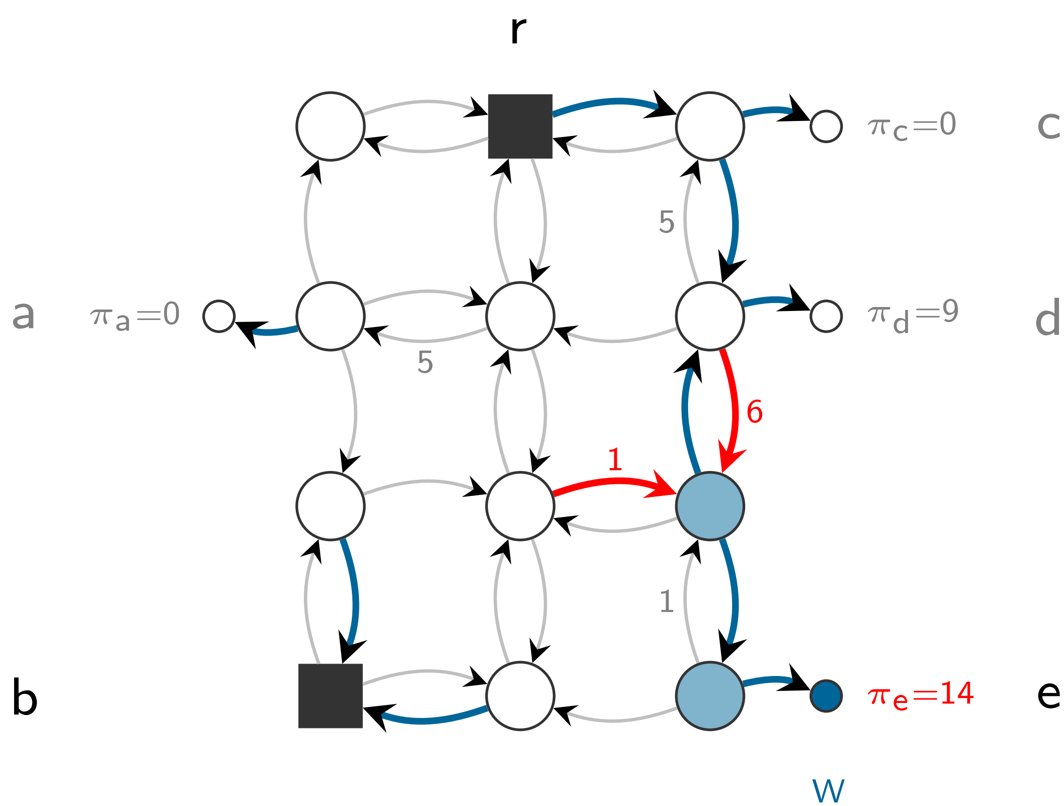
$$T_a = \{a, b, c, d, e\}$$
$$LB = 0,\ T_a = \{a, b, c, d, e\}$$
$$LB = 1,\ T_a = \{b, c, d, e\}$$
$$LB = 2,\ T_a = \{b, d, e\}$$
$$LB = 8,\ T_a = \{b, d, e\}$$
$$LB = 14,\ T_a = \{b, d, e\}$$
$$LB = 20,\ T_a = \{b, d, e\}$$
$$LB = 25,\ T_a = \{b, e\}$$



$c_{ij} = 6\ \forall (i, j) \in A$

Dual ascent - Example

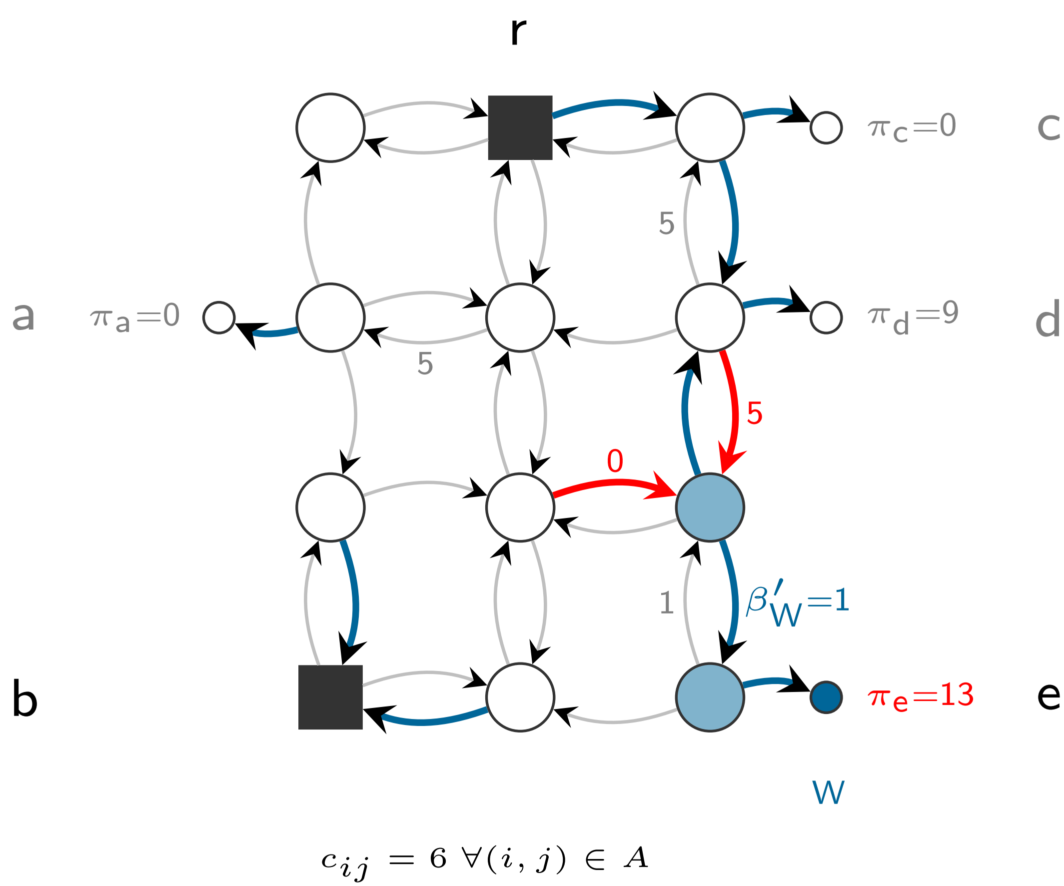
$$T_a = \{a, b, c, d, e\}$$
$$LB = 0,\ T_a = \{a, b, c, d, e\}$$
$$LB = 1,\ T_a = \{b, c, d, e\}$$
$$LB = 2,\ T_a = \{b, d, e\}$$
$$LB = 8,\ T_a = \{b, d, e\}$$
$$LB = 14,\ T_a = \{b, d, e\}$$
$$LB = 20,\ T_a = \{b, d, e\}$$
$$LB = 25,\ T_a = \{b, e\}$$



$$c_{ij} = 6\ \forall (i, j) \in A$$

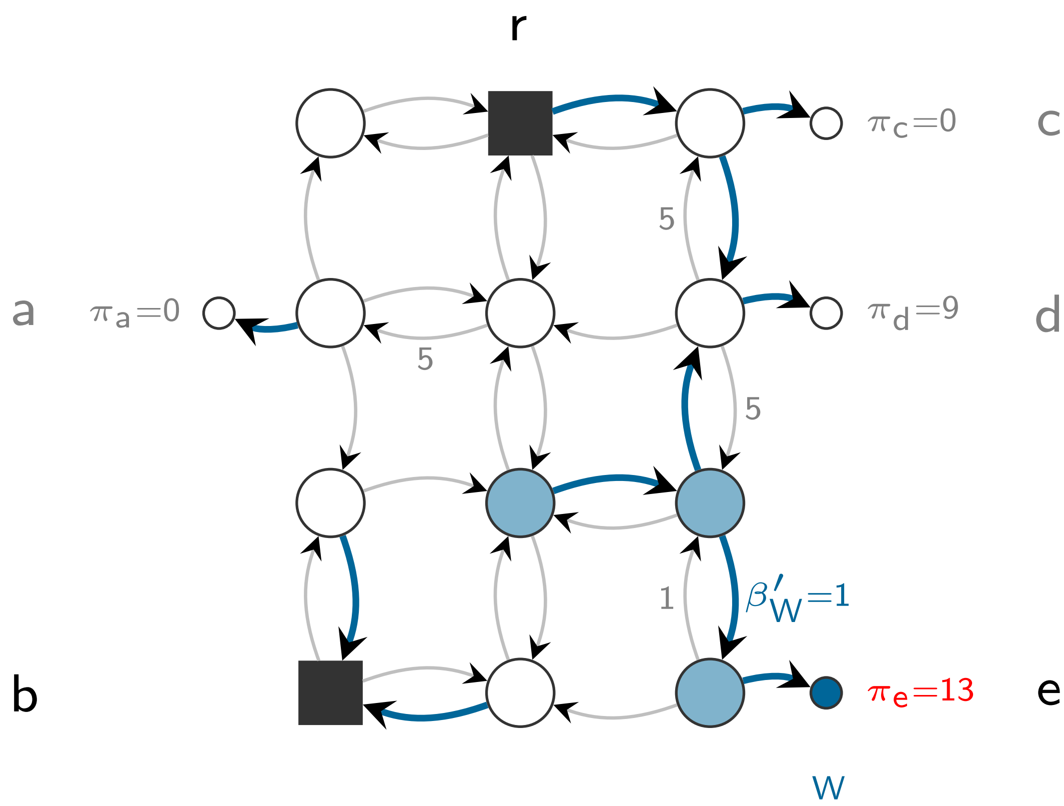
Dual ascent - Example

$T_a = \{a, b, c, d, e\}$
$LB = 0,\ T_a = \{a, b, c, d, e\}$
$LB = 1,\ T_a = \{b, c, d, e\}$
$LB = 2,\ T_a = \{b, d, e\}$
$LB = 8,\ T_a = \{b, d, e\}$
$LB = 14,\ T_a = \{b, d, e\}$
$LB = 20,\ T_a = \{b, d, e\}$
$LB = 25,\ T_a = \{b, e\}$
$LB = 26,\ T_a = \{b, e\}$



Dual ascent - Example

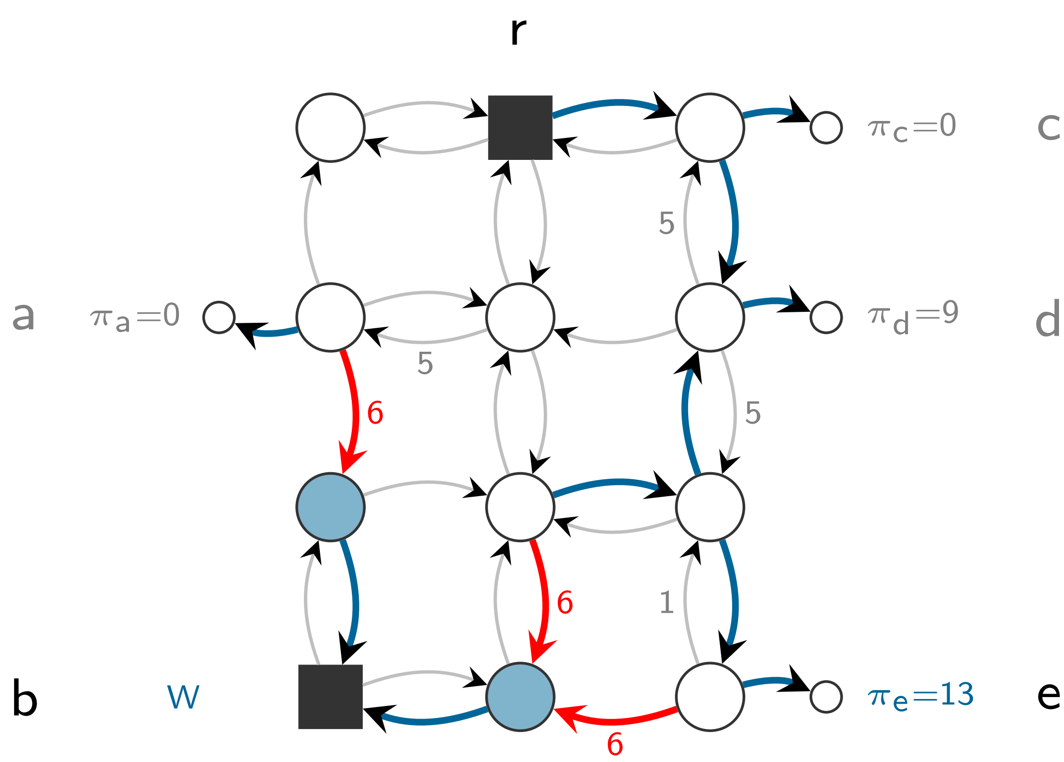
$T_a = \{a, b, c, d, e\}$
$LB = 0,\ T_a = \{a, b, c, d, e\}$
$LB = 1,\ T_a = \{b, c, d, e\}$
$LB = 2,\ T_a = \{b, d, e\}$
$LB = 8,\ T_a = \{b, d, e\}$
$LB = 14,\ T_a = \{b, d, e\}$
$LB = 20,\ T_a = \{b, d, e\}$
$LB = 25,\ T_a = \{b, e\}$
$LB = 26,\ T_a = \{b, e\}$



$c_{ij} = 6\ \forall (i, j) \in A$

Dual ascent - Example

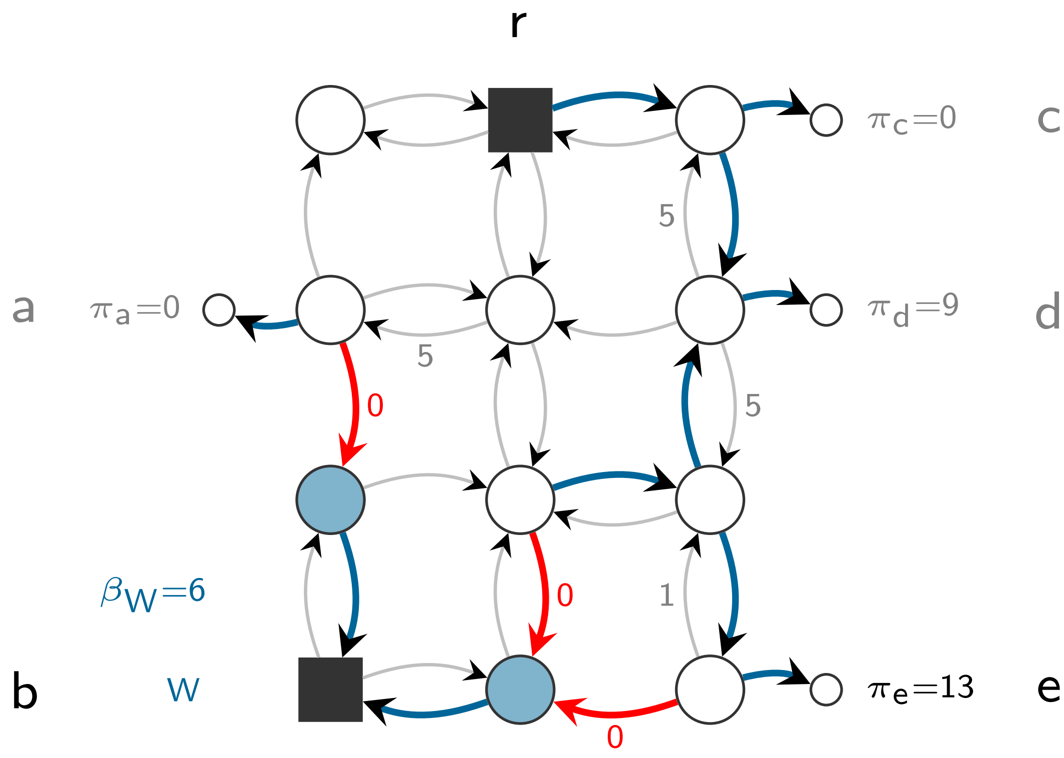
$$T_a = \{a, b, c, d, e\}$$
$$LB = 0,\ T_a = \{a, b, c, d, e\}$$
$$LB = 1,\ T_a = \{b, c, d, e\}$$
$$LB = 2,\ T_a = \{b, d, e\}$$
$$LB = 8,\ T_a = \{b, d, e\}$$
$$LB = 14,\ T_a = \{b, d, e\}$$
$$LB = 20,\ T_a = \{b, d, e\}$$
$$LB = 25,\ T_a = \{b, e\}$$
$$LB = 26,\ T_a = \{b, e\}$$



$c_{ij} = 6\ \forall (i, j) \in A$

Dual ascent - Example

$$T_a = \{a, b, c, d, e\}$$
$LB = 0,\ T_a = \{a, b, c, d, e\}$
$LB = 1,\ T_a = \{b, c, d, e\}$
$LB = 2,\ T_a = \{b, d, e\}$
$LB = 8,\ T_a = \{b, d, e\}$
$LB = 14,\ T_a = \{b, d, e\}$
$LB = 20,\ T_a = \{b, d, e\}$
$LB = 25,\ T_a = \{b, e\}$
$LB = 26,\ T_a = \{b, e\}$
$LB = 32,\ T_a = \{b\}$



$\pi_c=0$, $\pi_a=0$, $\pi_d=9$, $\pi_e=13$, $\beta_W=6$

$c_{ij} = 6\ \forall (i,j) \in A$

Dual ascent - Example

$$T_a = \{a, b, c, d, e\}$$

$LB = 0,\ T_a = \{a, b, c, d, e\}$

$LB = 1,\ T_a = \{b, c, d, e\}$

$LB = 2,\ T_a = \{b, d, e\}$

$LB = 8,\ T_a = \{b, d, e\}$

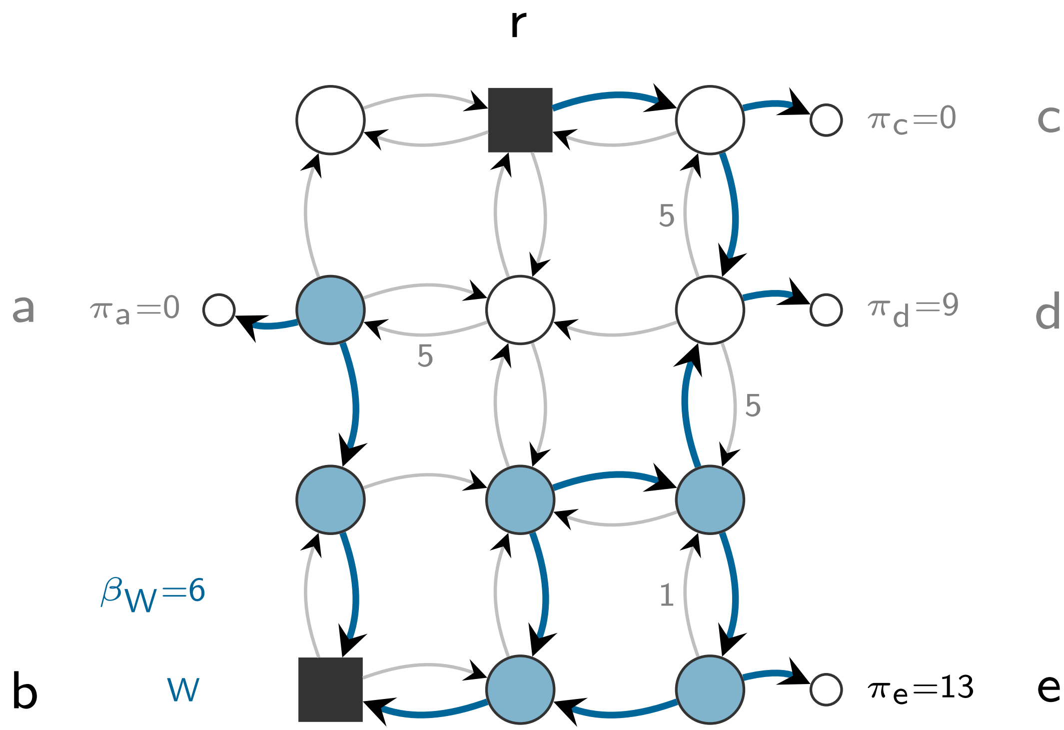
$LB = 14,\ T_a = \{b, d, e\}$

$LB = 20,\ T_a = \{b, d, e\}$

$LB = 25,\ T_a = \{b, e\}$

$LB = 26,\ T_a = \{b, e\}$

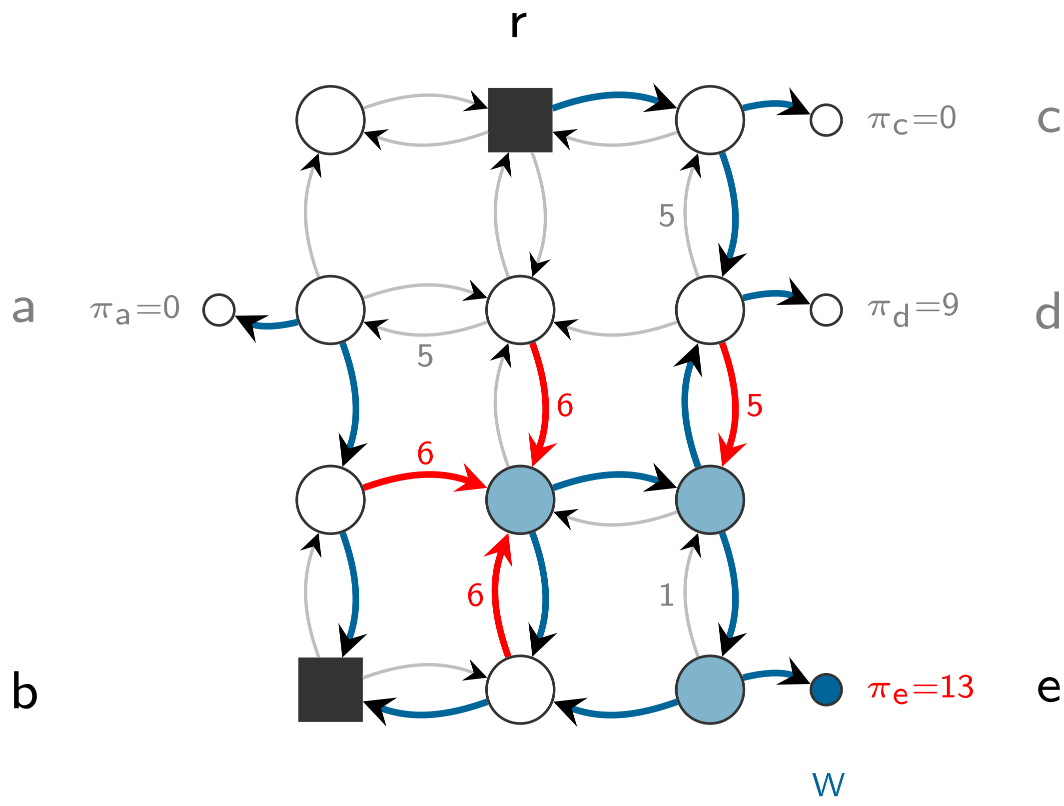
$LB = 32,\ T_a = \{b\}$



$c_{ij} = 6\ \forall (i, j) \in A$

Dual ascent - Example

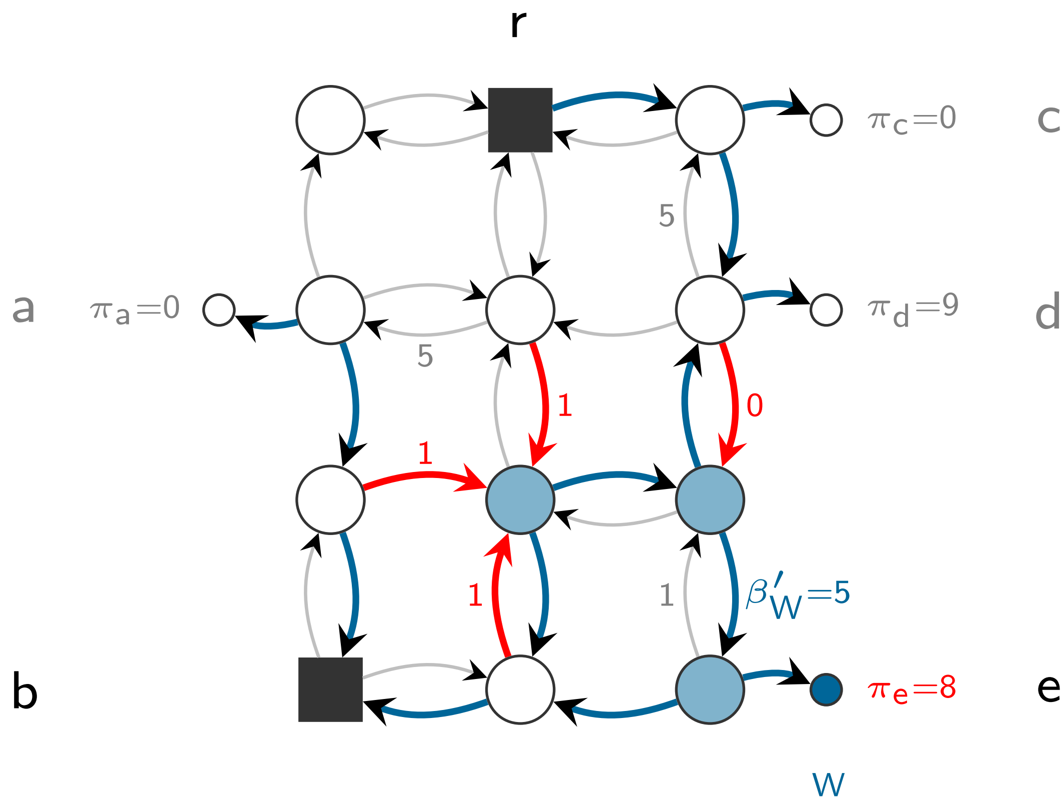
$$T_a = \{a, b, c, d, e\}$$
$$LB = 0,\ T_a = \{a, b, c, d, e\}$$
$$LB = 1,\ T_a = \{b, c, d, e\}$$
$$LB = 2,\ T_a = \{b, d, e\}$$
$$LB = 8,\ T_a = \{b, d, e\}$$
$$LB = 14,\ T_a = \{b, d, e\}$$
$$LB = 20,\ T_a = \{b, d, e\}$$
$$LB = 25,\ T_a = \{b, e\}$$
$$LB = 26,\ T_a = \{b, e\}$$
$$LB = 32,\ T_a = \{b\}$$



$c_{ij} = 6\ \forall (i, j) \in A$

Dual ascent - Example

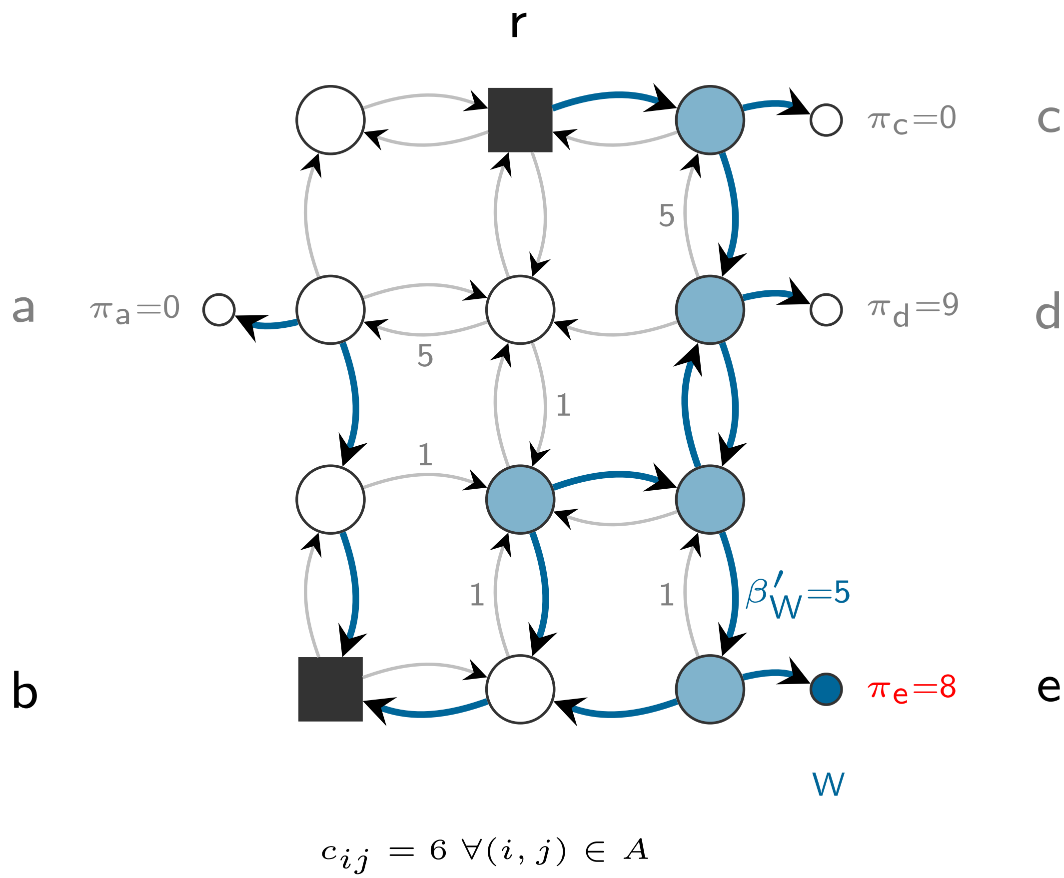
$$T_a = \{a, b, c, d, e\}$$
$$LB = 0,\ T_a = \{a, b, c, d, e\}$$
$$LB = 1,\ T_a = \{b, c, d, e\}$$
$$LB = 2,\ T_a = \{b, d, e\}$$
$$LB = 8,\ T_a = \{b, d, e\}$$
$$LB = 14,\ T_a = \{b, d, e\}$$
$$LB = 20,\ T_a = \{b, d, e\}$$
$$LB = 25,\ T_a = \{b, e\}$$
$$LB = 26,\ T_a = \{b, e\}$$
$$LB = 32,\ T_a = \{b\}$$
$$LB = 37$$



$c_{ij} = 6\ \forall (i,j) \in A$

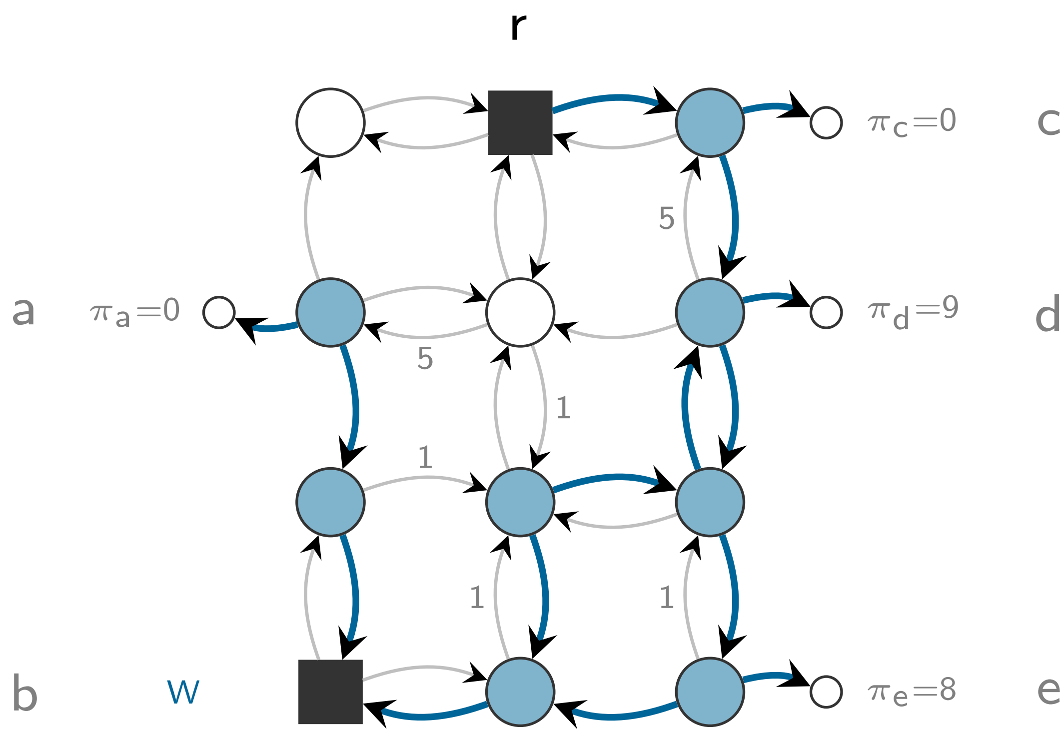
Dual ascent - Example

$$T_a = \{a, b, c, d, e\}$$
$LB = 0$, $T_a = \{a, b, c, d, e\}$
$LB = 1$, $T_a = \{b, c, d, e\}$
$LB = 2$, $T_a = \{b, d, e\}$
$LB = 8$, $T_a = \{b, d, e\}$
$LB = 14$, $T_a = \{b, d, e\}$
$LB = 20$, $T_a = \{b, d, e\}$
$LB = 25$, $T_a = \{b, e\}$
$LB = 26$, $T_a = \{b, e\}$
$LB = 32$, $T_a = \{b\}$
$LB = 37$



Dual ascent - Example

$T_a = \{a, b, c, d, e\}$
$LB = 0,\ T_a = \{a, b, c, d, e\}$
$LB = 1,\ T_a = \{b, c, d, e\}$
$LB = 2,\ T_a = \{b, d, e\}$
$LB = 8,\ T_a = \{b, d, e\}$
$LB = 14,\ T_a = \{b, d, e\}$
$LB = 20,\ T_a = \{b, d, e\}$
$LB = 25,\ T_a = \{b, e\}$
$LB = 26,\ T_a = \{b, e\}$
$LB = 32,\ T_a = \{b\}$
$LB = 37 T_a = \{\}$
$\rightarrow$ Terminate.
$LB = 37$



$c_{ij} = 6\ \forall (i, j) \in A$

Resulting saturated graph G_S is very useful!

Upon termination of DA:

- We have a valid LB
- We have dual information in form of reduced costs on edges
- We can perform reduction tests:
  - Decrease instance size while preserving at least one optimal solution
  - Operations: exclude/fix/merge arcs and nodes
- We can create heuristic solutions from G_S

DA can be applied in every B&B node

Reduction Tests

Reduction tests

Natural extensions of tests known for the STP, PCSTP:

- Bound-based arc/node elimination
  (STP, Duin, 1993; Polzin and Daneshmand, 2001)
- Degree 1/2, least cost, non-reachability
  (STP, Duin, 1993)
- (Asymmetric) minimum adjacency
  (PCSTP, Duin and Volgenant, 1987; Ljubić et al., 2006)
- Bound-based node inclusion

Complementary new tests based on graph connectivity:

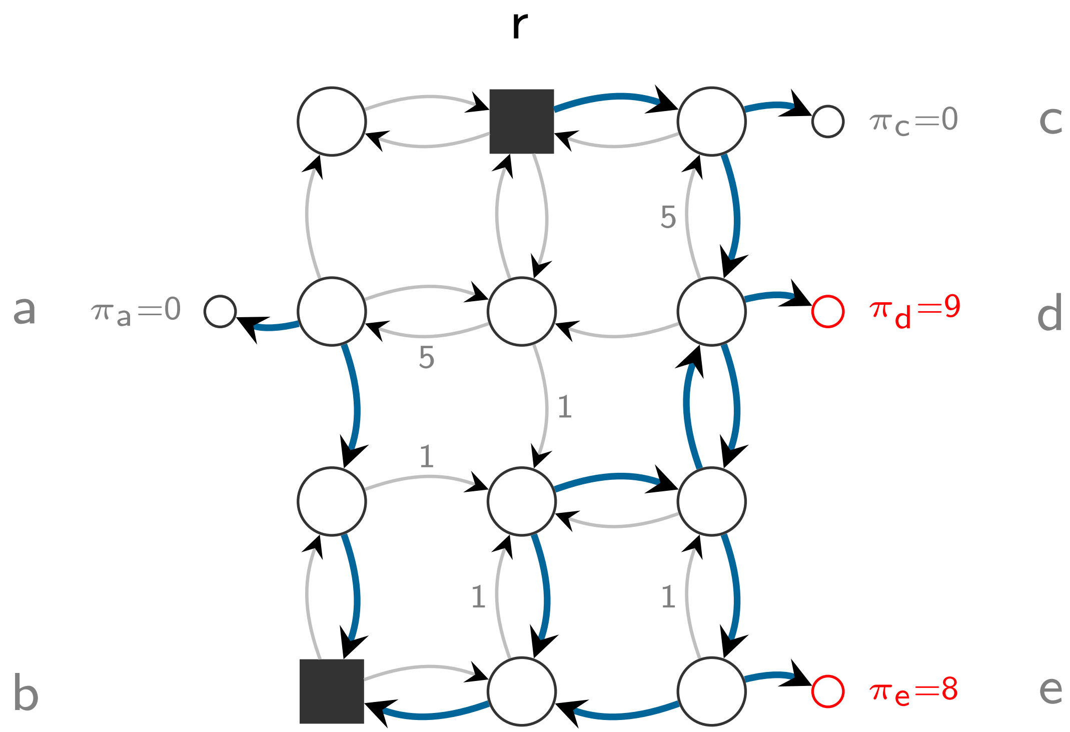
- Single-successor, minimum-successor

Bound-based reductions

Node inclusion: $i \in T_p$ can be added to T_f if

$$LB + \pi_i > UB$$

$LB = 37$, assume $UB = 42$

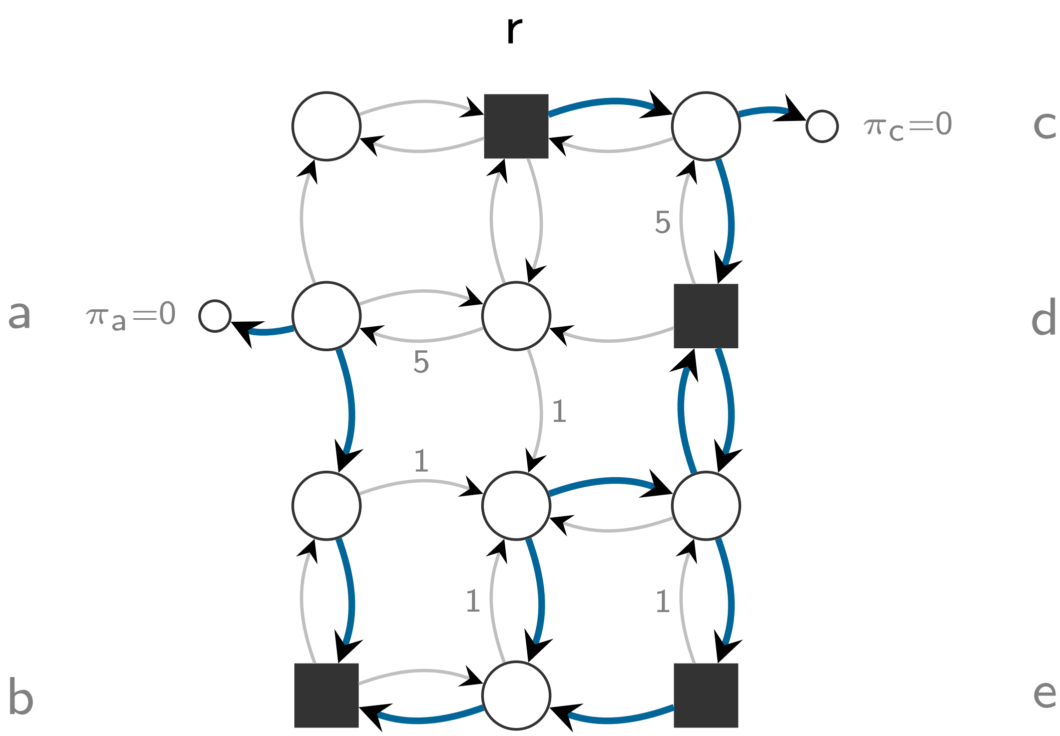


Bound-based reductions

Node inclusion: $i \in T_p$ can be added to T_f if

$$LB + \pi_i > UB$$

$LB = 37$, assume $UB = 42$

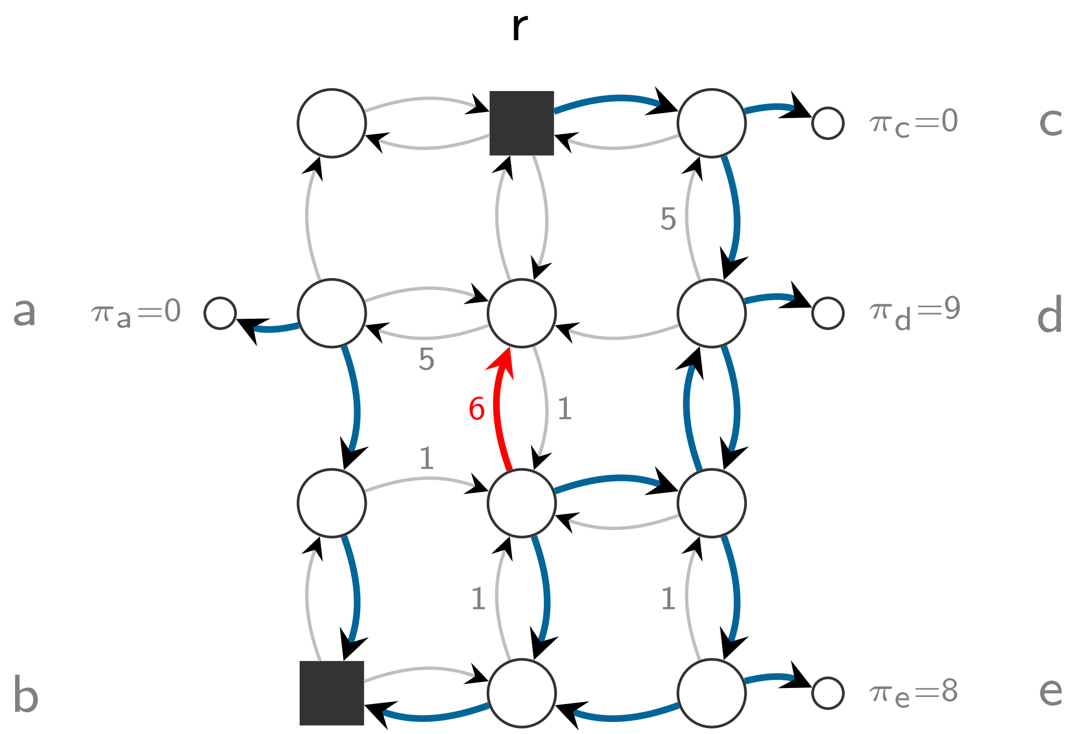


Bound-based reductions

Arc elimination: (i, j) can be removed if

$$LB + \tilde{d}(r, i) + \tilde{c}_{ij} + \min_{t \in T \setminus \{r\}} \tilde{d}(j, t) > UB$$

$LB = 37$, assume $UB = 42$

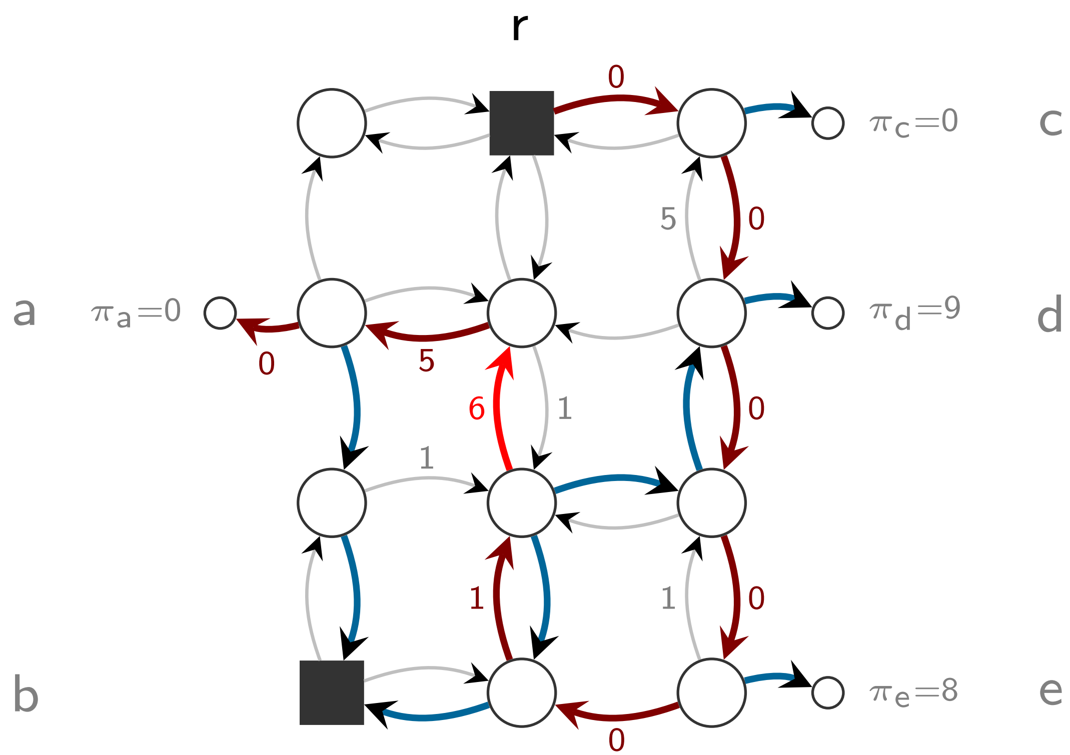


Bound-based reductions

Arc elimination: (i,j) can be removed if

$$LB + \tilde{d}(r,i) + \tilde{c}_{ij} + \min_{t \in T \setminus \{r\}} \tilde{d}(j,t) > UB$$

$LB = 37$, assume $UB = 42$

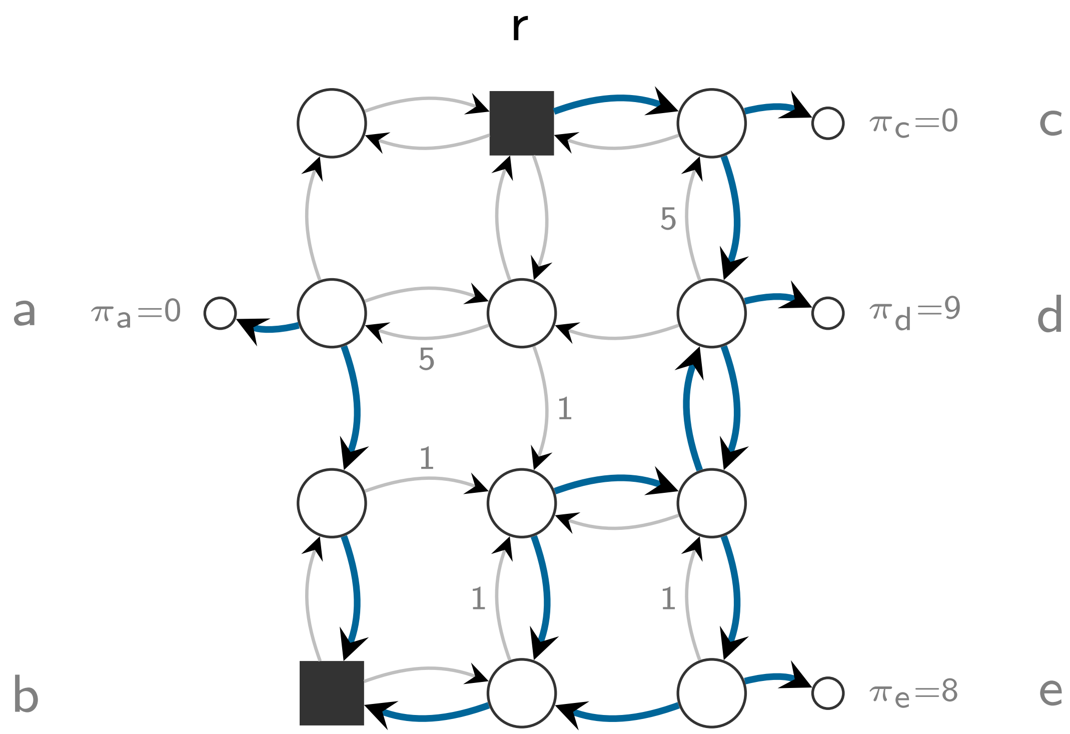


Bound-based reductions

Arc elimination: (i, j) can be removed if

$$LB + \tilde{d}(r, i) + \tilde{c}_{ij} + \min_{t \in T \setminus \{r\}} \tilde{d}(j, t) > UB$$

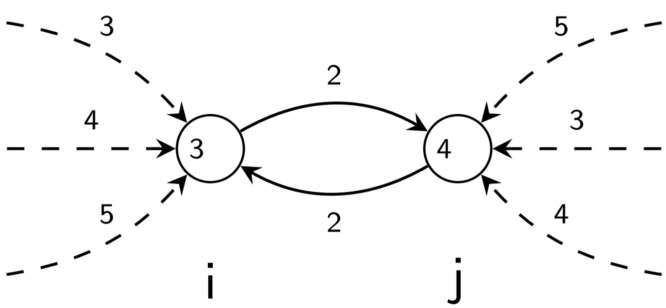
$LB = 37$, assume $UB = 42$



(Asymmetric) minimum adjacency

Minimum adjacency: adjacent nodes i, j can be merged if $c_{ij} = c_{ji} < \min\{p_i, p_j\}$ and

$$c_{ji} = \min_{(k,i)\in\delta^-(i)} c_{ki} \qquad c_{ij} = \min_{(k,j)\in\delta^-(j)} c_{kj}$$

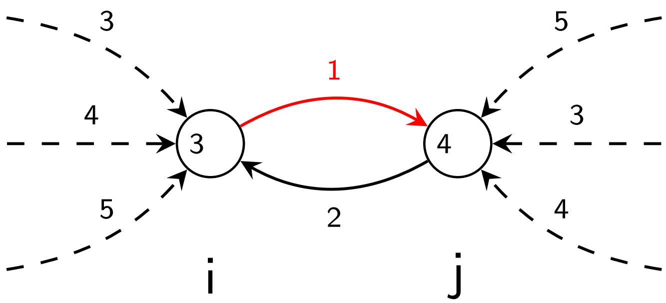


Either none or exactly one of (i, j) and (j, i) will be part of an optimal solution.

(Asymmetric) minimum adjacency

Minimum adjacency: adjacent nodes i, j can be merged if $c_{ij} = c_{ji} < \min\{p_i, p_j\}$ and

$$c_{ji} = \min_{(k,i)\in\delta^-(i)} c_{ki} \qquad c_{ij} = \min_{(k,j)\in\delta^-(j)} c_{kj}$$



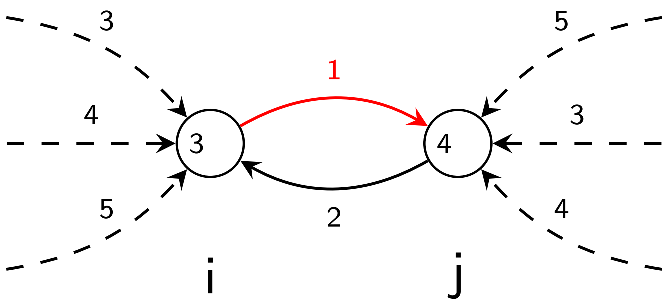
Either none or exactly one of (i, j) and (j, i) will be part of an optimal solution.

Question: What if $c_{ij} \neq c_{ji}$?

(Asymmetric) minimum adjacency

Minimum adjacency: adjacent nodes i, j can be merged if $c_{ij} = c_{ji} < \min\{p_i, p_j\}$ and

$$c_{ji} = \min_{(k,i) \in \delta^-(i)} c_{ki} \qquad c_{ij} = \min_{(k,j) \in \delta^-(j)} c_{kj}$$



Either none or exactly one of (i, j) and (j, i) will be part of an optimal solution.

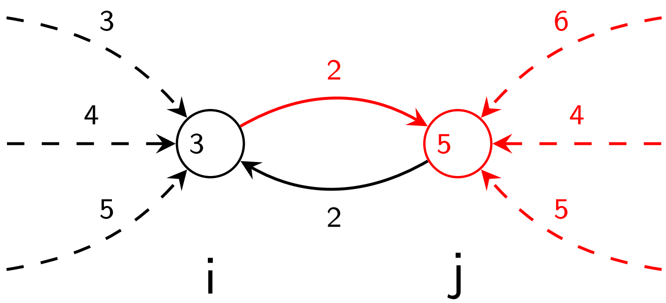
Question: What if $c_{ij} \neq c_{ji}$?

If $i \neq r$ / $j \neq r$, eliminate asymmetry by cost shifting

(Asymmetric) minimum adjacency

Minimum adjacency: adjacent nodes i, j can be merged if $c_{ij} = c_{ji} < \min\{p_i, p_j\}$ and

$$c_{ji} = \min_{(k,i)\in\delta^-(i)} c_{ki} \qquad c_{ij} = \min_{(k,j)\in\delta^-(j)} c_{kj}$$



$$c_{\text{fixed}} := c_{\text{fixed}} - 1$$

Either none or exactly one of (i, j) and (j, i) will be part of an optimal solution.

Question: What if $c_{ij} \neq c_{ji}$?

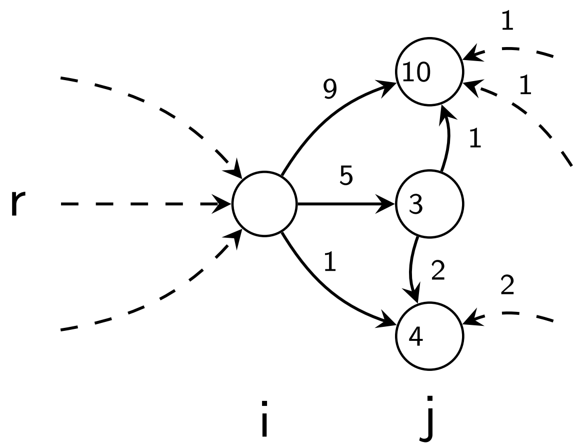
If $i \neq r$ / $j \neq r$, eliminate asymmetry by cost shifting

Single/Minimum successor

Augment local (asymmetric) minimum adjacency test with global (connectivity) information

Minimum successor: (i, j) can be contracted if i separates j from r (cut node) and

$$p_j > c_{ij} = \min_{(k,j) \in \delta^-(j)} c_{kj}$$



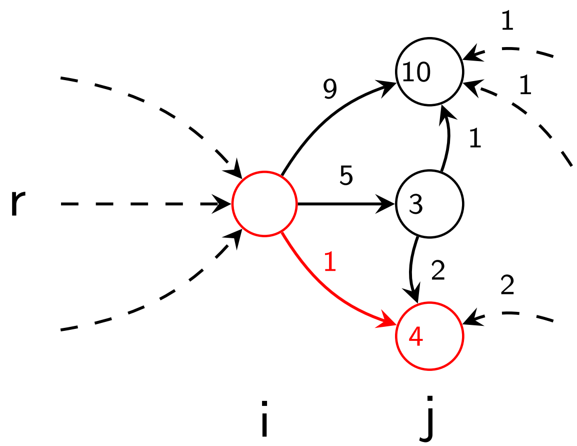
Single successor: (i, j) can be contracted if (i, j) separates j from r (cut arc) and $p_j > c_{ij}$.

Single/Minimum successor

Augment local (asymmetric) minimum adjacency test with global (connectivity) information

Minimum successor: (i, j) can be contracted if i separates j from r (cut node) and

$$p_j > c_{ij} = \min_{(k,j)\in\delta^-(j)} c_{kj}$$



Single successor: (i, j) can be contracted if (i, j) separates j from r (cut arc) and $p_j > c_{ij}$.

Other algorithmic details

Branching strategies

Root-multiway branching
decompose unrooted APCSTP instances into rooted instances

Node-based branching
priority based on highest degree in saturation graph G_S

Primal heuristics

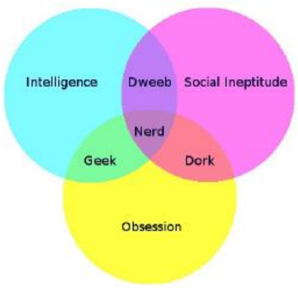
Search for primal solutions on G_S

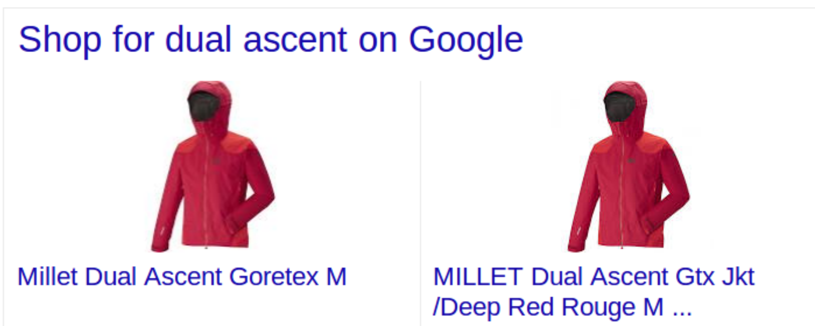
Cost shifting

Shift costs down as far as possible

Supports reduction tests, primal heuristics, dual ascent

Computational Comparison.
Staynerd or Mozartballs or DualAscent??





Computational results

B&B framework implemented in C++

Intel Xeon CPU (2.5 GHz)

414 benchmark instances gathered during the 11^{th} DIMACS Challenge on Steiner tree problems:

(rooted) PCSTP, MWCS, NWSTP

Time limit: 1 hour

Memory limit: 16 GB

Computational results

		#Inst.	avg. size $\|V\|$	$\|A\|$	$\|T\|$	B&B #Nds.	t[s.]	speedup w.r.t CPLEX†
PCSTP	CRR	80	500	12469	140	27	0.4	4
	JMP	34	100	568	46	0	0.1	10
	RANDOM	68	4000	64056	4000	99	4.3	8
	HANDSD	10	39600	157408	19135	2	5.5	228*
	HANDSI	10	42500	168950	19905	81	5.5	94*
	I640-0	25	640	100700	61	1	2.3	12
	I640-1	25	640	100700	61	54	4.6	22
RPCSTP	COLOGNE	29	1294	23435	9	0	0.2	284
MWCS	ACTMOD	8	3933	82311	3595	1	2.0	2
	JMPALMK	72	938	17390	936	0	0.1	2

(*) Data sets contained instances previously unsolved within an hour
(†) State-of-the-art exact ILP-based B&C approach by Fischetti et al. (2016), winner of most categories during the 11th DIMACS Challenge on Steiner tree problems

Computational results: summary on large-scale instances

NWSTP	$\|V\|$	$\|A\|$	$\|T\|$	B&B #Nds.	B&B gap	B&B time	SCIPJACK/CPLEX gap	SCIPJACK/CPLEX time
hiv-1	205717	4932002	54857	4	0.05	*TL*	0.0049	72 (hrs.)†

PCSTP	$\|V\|$	$\|A\|$	$\|T\|$	B&B #Nds.	B&B gap	B&B time	SCIPJACK/CPLEX gap	SCIPJACK/CPLEX time
handbi01	158400	631616	157385	0	0.00	117.2	1.10	*TL*
handbi02	158400	631616	8589	33	0.00	44.3	2.71	*TL*
handbi03	158400	631616	154148	0	0.00	11.3	0.00	1246.2
handbi04	158400	631616	16288	29518	0.06	*TL*	4.22	*TL*
handbi05	158400	631616	155695	0	0.00	12.4	0.00	916.3
i640-241	640	81792	50	1751	0.00	89.2	0.24	*TL*
i640-321	640	408960	160	25615	0.00	2544.1	0.36	*TL*
i640-322	640	408960	160	6583	0.00	2573.7	0.31	*TL*
i640-323	640	408960	160	3163	0.00	1906.2	0.26	*TL*
i640-324	640	408960	160	16955	0.00	1306.1	0.26	*TL*
i640-325	640	408960	160	3195	0.00	818.9	0.29	*TL*

Solved previously unsolved instances: 6 (I640), 13 (HANDBI/BD), 4 (HANDSI/SD)

(†) computed by SCIPJACK, exact ILP-based B&C approach by Gamrath et al. (2016) (on a machine with 386 GB memory)

Conclusions

Presented B&B framework based on a
dual ascent algorithm & reduction tests for the APCSTP

APCSTP generalizes several fundamental network design problems

Extremely good results on large-scale instances

Outperforms state-of-the-art exact ILP solver in most cases

The biggest synthetic PUC instances still unsolved (there `Mozartballs` outperforms `DualAscent`)

Source code publicly available at
`https://github.com/mluipersbeck/dapcstp`

No MIP solvers involved - ideal for applications in bioinformatics

Single-thread so far

Thank you for your attention!

Questions?

Conclusions

- Presented B&B framework based on a dual ascent algorithm & reduction tests for the APCSTP
- APCSTP generalizes several fundamental network design problems

- Extremely good results on large-scale instances
- Outperforms state-of-the-art exact ILP solver in most cases
- The biggest synthetic PUC instances still unsolved (there `Mozartballs` outperforms `DualAscent`)

Source code publicly available at
`https://github.com/mluipersbeck/dapcstp`

- No MIP solvers involved - ideal for applications in bioinformatics
- Single-thread so far

Thank you for your attention!

Questions?

Literature I

C. W. Duin. *Steiner's problem in graphs*. PhD thesis, University of Amsterdam, 1993.

C. W. Duin and A. Volgenant. Some generalizations of the Steiner problem in graphs. *Networks*, 17(3):353–364, 1987. ISSN 1097-0037.

M. Fischetti, M. Leitner, I. Ljubić, M. Luipersbeck, M. Monaci, M. Resch, D. Salvagnin, and M. Sinnl. Thinning out Steiner trees: a node-based model for uniform edge costs. *Mathematical Programming Computation*, pages 1–27, 2016.

G. Gamrath, T. Koch, S. J. Maher, D. Rehfeldt, and Y. Shinano. SCIP-Jack — a solver for STP and variants with parallelization extensions. *Mathematical Programming Computation*, pages 1–66, 2016.

I. Ljubić, R. Weiskircher, U. Pferschy, G. W. Klau, P. Mutzel, and M. Fischetti. An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem. *Mathematical Programming*, 105(2-3):427–449, 2006.

T. Pajor, E. Uchoa, and R. F. Werneck. A robust and scalable algorithm for the Steiner problem in graphs. 2014. 11th DIMACS challenge workshop.

T. Polzin and S. V. Daneshmand. Improved algorithms for the Steiner problem in networks. *Discrete Applied Mathematics*, 112(1):263–300, 2001.

R. T. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming*, 28(3):271–287, 1984. ISSN 0025-5610.

Dual ascent - Algorithm

Data: instance $(G = (V, A), \boldsymbol{c}, \boldsymbol{p}, T_f, r)$
Result: lower bound LB, reduced costs $\tilde{\boldsymbol{c}}$, dual vector $\boldsymbol{\pi}$

$LB \leftarrow 0$
$\tilde{c}_{ij} \leftarrow c_{ij} \qquad \forall\,(i,j) \in A, j \notin T_p$
$\pi_j \leftarrow p_j \qquad \forall\, j \in T_p$
$T_a \leftarrow T_f \cup T_p \setminus \{r\}$
while $T_a \neq \emptyset$ **do**
  $k \leftarrow \texttt{chooseActiveTerminal}(T_a)$
  $W \leftarrow W(k)$
  $\Delta \leftarrow \min_{(i,j) \in \delta^-(W)} \tilde{c}_{ij}$
  if $k \in T_p$ **then**
    $\Delta \leftarrow \texttt{min}\{\Delta, \pi_k\}$
    $\pi_k \leftarrow \pi_k - \Delta$
  end
  $\tilde{c}_{ij} \leftarrow \tilde{c}_{ij} - \Delta \qquad \forall\,(i,j) \in \delta^-(W)$
  $LB \leftarrow LB + \Delta$
  $T_a \leftarrow \texttt{removeInactiveTerminals}(T_a)$
end

Worst-case complexity: $O(|A| \cdot \min\{|T||V|, |A|\})$