Building combinatorial optimization solvers with SCIP: Steiner tree, QUBO, and MaxCut

Daniel Rehfeldt

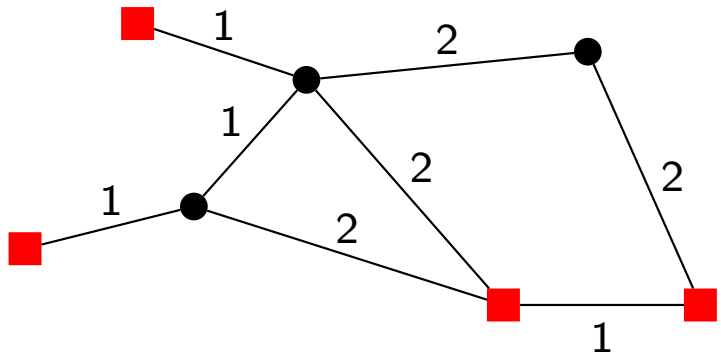
ZIB/I^{2}DAMO GmbH

November 4, 2022

The Steiner tree problem in graphs

Given:

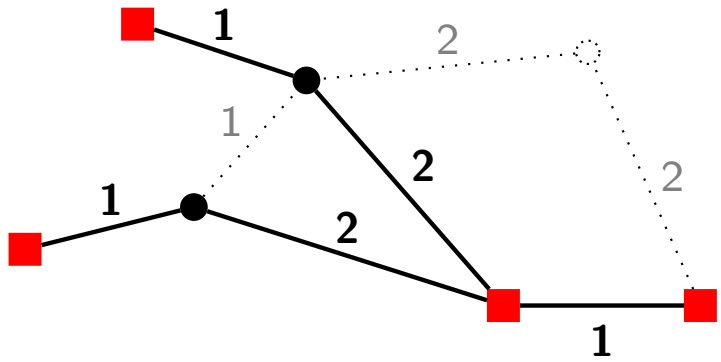
- $G = (V, E)$: undirected graph
- $T \subseteq V$: subset of vertices
- $c \in \mathbb{R}^E_{\geqslant 0}$: edge costs



The Steiner tree problem in graphs

Given:

- $G = (V, E)$: undirected graph
- $T \subseteq V$: subset of vertices
- $c \in \mathbb{R}^E_{\geqslant 0}$: edge costs

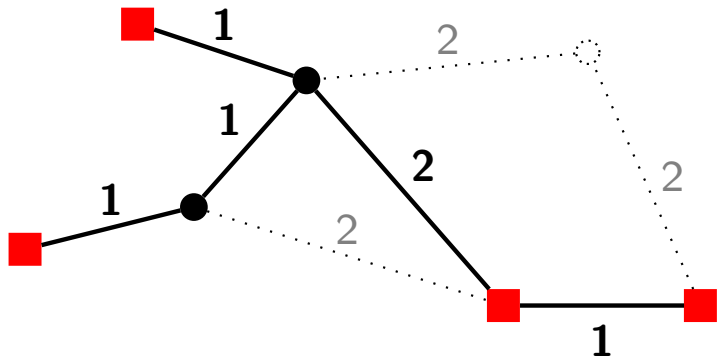


A tree $S \subseteq G$ is called Steiner tree in (G, T, c) if $T \subseteq V(S)$.

The Steiner tree problem in graphs

Given:

- $G = (V, E)$: undirected graph
- $T \subseteq V$: subset of vertices
- $c \in \mathbb{R}^E_{\geqslant 0}$: edge costs



A tree $S \subseteq G$ is called Steiner tree in (G, T, c) if $T \subseteq V(S)$.

Steiner tree problem in graphs (SPG)

Find a Steiner tree S in (G, T, c) with minimum edge costs $\sum_{e \in E(S)} c(e)$

Classic NP-hard problem (Karp '72).

- SPG is among the most studied problems in combinatorial optimization
- and has seen impressive theoretical progress in last 10 years, e.g. concerning approximability and complexity

- SPG is among the most studied problems in combinatorial optimization
- and has seen impressive theoretical progress in last 10 years, e.g. concerning approximability and complexity

...however:

- state of the art in practical exact SPG solution mostly unchallenged for 20 years
  (established by Polzin and Vahdati Daneshmand)
- International Steiner tree competitions DIMACS 2014 and PACE 2018 brought some progress, but Polzin and Vahdati still drastically stronger

- SPG is among the most studied problems in combinatorial optimization
- and has seen impressive theoretical progress in last 10 years, e.g. concerning approximability and complexity

...however:

- state of the art in practical exact SPG solution mostly unchallenged for 20 years
  (established by Polzin and Vahdati Daneshmand)
- International Steiner tree competitions DIMACS 2014 and PACE 2018 brought some progress, but Polzin and Vahdati still drastically stronger

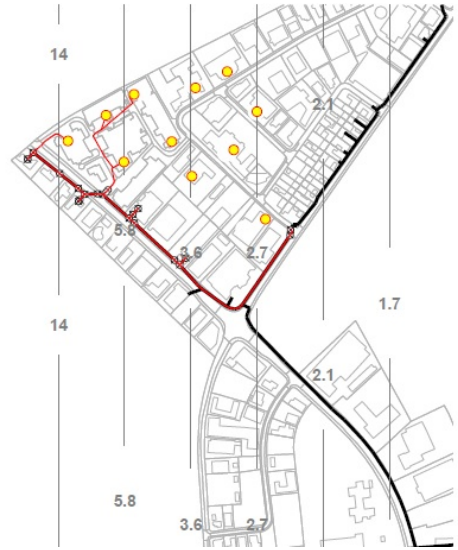
We developed new solver SCIP-Jack, aiming to

- once again advance state of the art in exact SPG solution
- extend SPG results to related problems, which are more common in practice ...

Applications

Some real-world applications of Steiner trees:

- design of fiber optic networks
- prediction of cancer tumor evolution
- Quantum computer design
- design of district heating networks
- computational biology
- ...



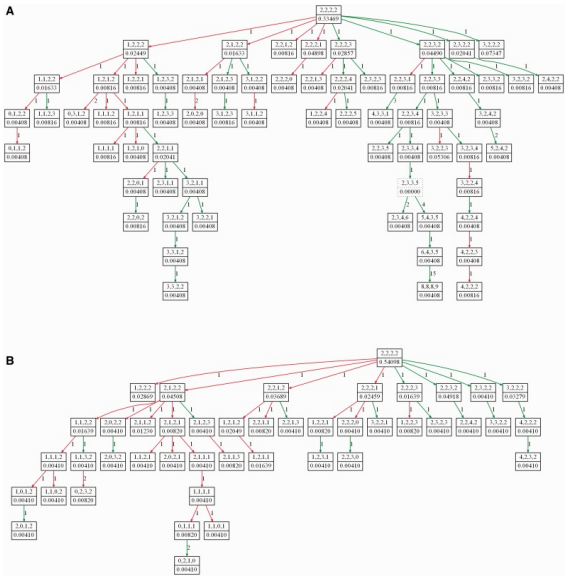
Rooted prize-collecting Steiner tree problem

E.g. *An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem* (Ljubic et al., 2006)

Applications

Some real-world applications of Steiner trees:

- design of fiber optic networks
- prediction of cancer tumor evolution
- Quantum computer design
- design of district heating networks
- computational biology
- ...



Rectilinear Steiner minimum tree problem

E.g. *Phylogenetic analysis of multiprobe fluorescence in situ hybridization data from tumor cell populations* (Chowdhury et al., 2013)

Applications

Some real-world applications of Steiner trees:

- design of fiber optic networks
- prediction of cancer tumor evolution
- Quantum computer design
- design of district heating networks
- computational biology
- ...

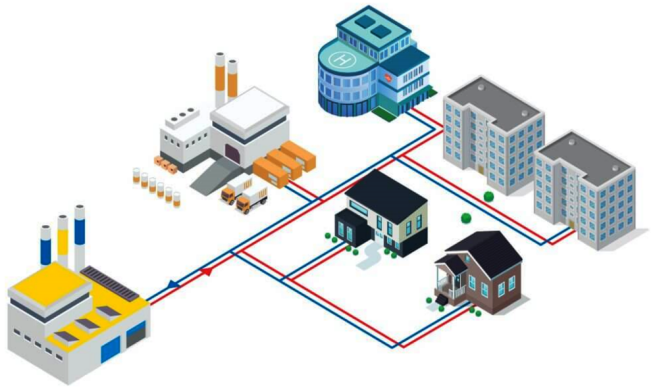
Classic Steiner tree problem

E.g. *CNOT circuit extraction for topologically-constrained quantum memories* (Kissinger et. al., 2020)

Applications

Some real-world applications of Steiner trees:

- design of fiber optic networks
- prediction of cancer tumor evolution
- Quantum computer design
- design of district heating networks
- computational biology
- ...



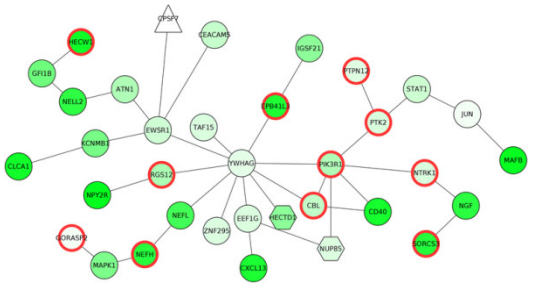
Prize-collecting Steiner tree problem

Energy distribution

Applications

Some real-world applications of Steiner trees:

- design of fiber optic networks
- prediction of cancer tumor evolution
- Quantum computer design
- design of district heating networks
- computational biology
- ...



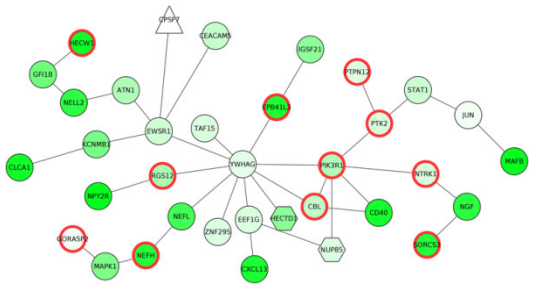
Maximum-weight connected subgraph problem

E.g. *Solving Generalized Maximum-Weight Connected Subgraph Problems for Network Enrichment Analysis* (Loboda et al., 2016)

Applications

Some real-world applications of Steiner trees:

- design of fiber optic networks
- prediction of cancer tumor evolution
- Quantum computer design
- design of district heating networks
- computational biology
- ...



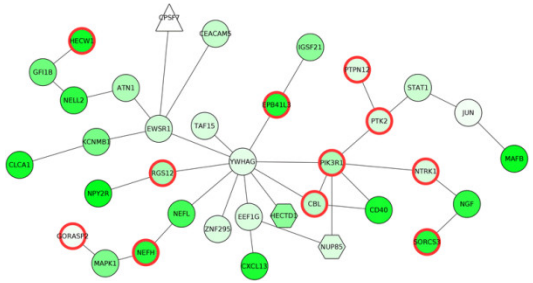
Maximum-weight connected subgraph problem

Real-world applications usually require variations of SPG

Applications

Some real-world applications of Steiner trees:

- design of fiber optic networks
- prediction of cancer tumor evolution
- Quantum computer design
- design of district heating networks
- computational biology
- ...



Maximum-weight connected subgraph problem

Real-world applications usually require variations of SPG

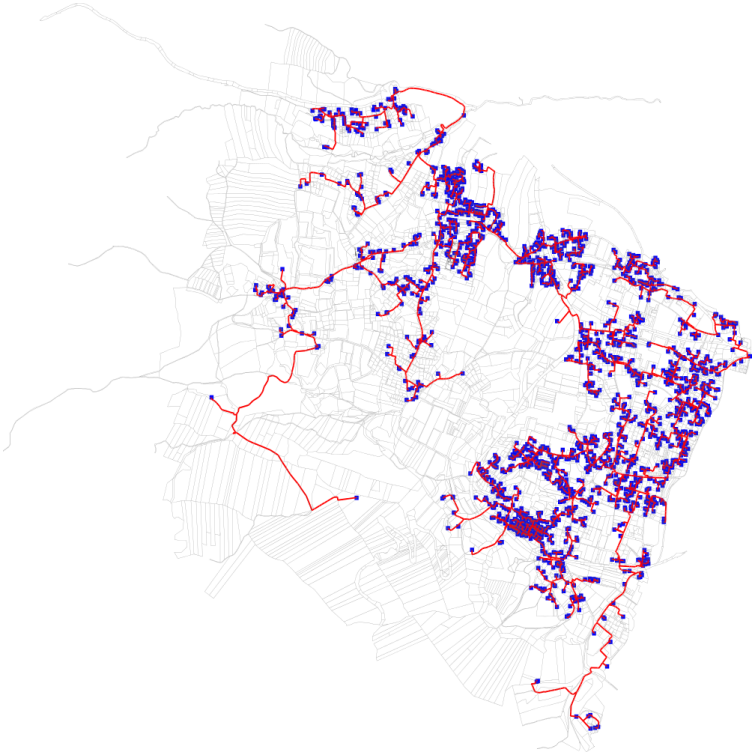
SCIP-Jack is used for all above applications
(both in industry and academia)

Why not using a general MIP solver?

Consider exemplary (rather small-scale) network design instance with:

$$|V| = 12\,715$$
$$|E| = 20\,632$$
$$|T| = 475$$

- CPLEX 12.10, Gurobi 9.5:
  No optimal solution after one week



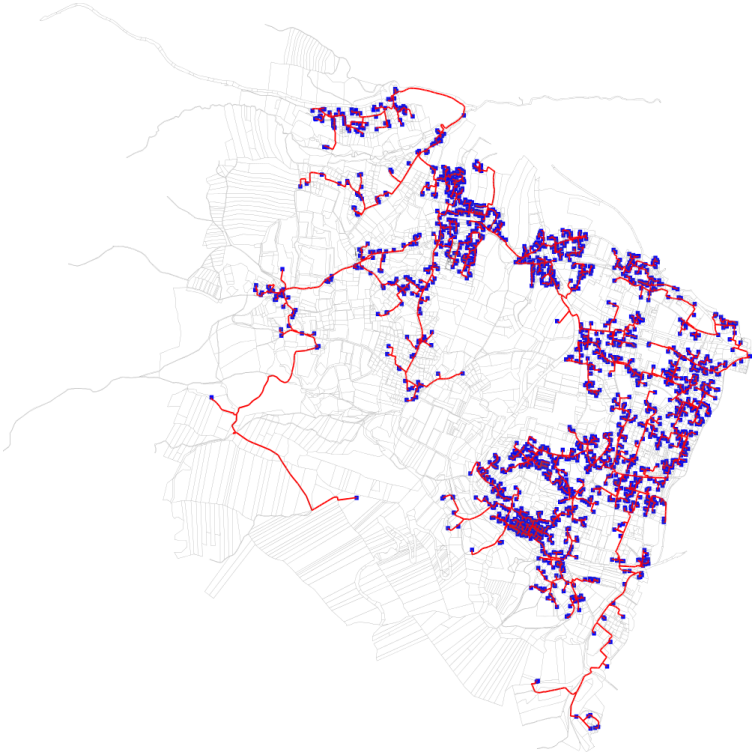
Instance from network telecommunication design for Austrian cities, see *New Real-world Instances for the Steiner Tree Problem in Graphs* (Leitner et al., 2014)

Why not using a general MIP solver?

Consider exemplary (rather small-scale) network design instance with:

$$|V| = 12\,715$$
$$|E| = 20\,632$$
$$|T| = 475$$

- CPLEX 12.10, Gurobi 9.5:
  No optimal solution after one week
- SCIP-Jack:
  Solves to optimality in 0.2 seconds



Instance from network telecommunication design for Austrian cities, see *New Real-world Instances for the Steiner Tree Problem in Graphs* (Leitner et al., 2014)

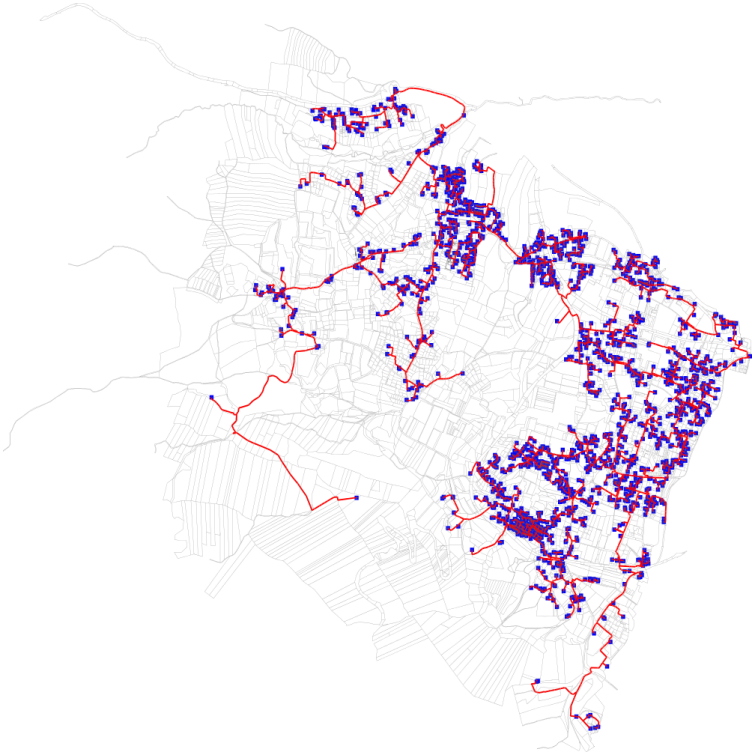
Why not using a general MIP solver?

Consider exemplary (rather small-scale) network design instance with:

$$|V| = 12\,715$$
$$|E| = 20\,632$$
$$|T| = 475$$

- CPLEX 12.10, Gurobi 9.5:
  No optimal solution after one week
- SCIP-Jack:
  Solves to optimality in 0.2 seconds

SCIP-Jack is routinely employed for instances with millions of edges in industry.



Instance from network telecommunication design for Austrian cities, see *New Real-world Instances for the Steiner Tree Problem in Graphs* (Leitner et al., 2014)

Steiner tree problem in graphs

SCIP-Jack combines variety of techniques and algorithms in branch-and-cut framework, e.g.

Steiner tree problem in graphs

SCIP-Jack combines variety of techniques and algorithms in branch-and-cut framework, e.g.

- preprocessing routines
- decomposition methods
- cutting-plane separation for LP-relaxation
- dynamic-programming algorithms
- primal heuristics
- domain propagation
- customized branching
- ...

Implemented as an extension of SCIP. Further contributers: G. Gamrath, T. Koch, S. Maher, M. Winkler.

Steiner tree problem in graphs

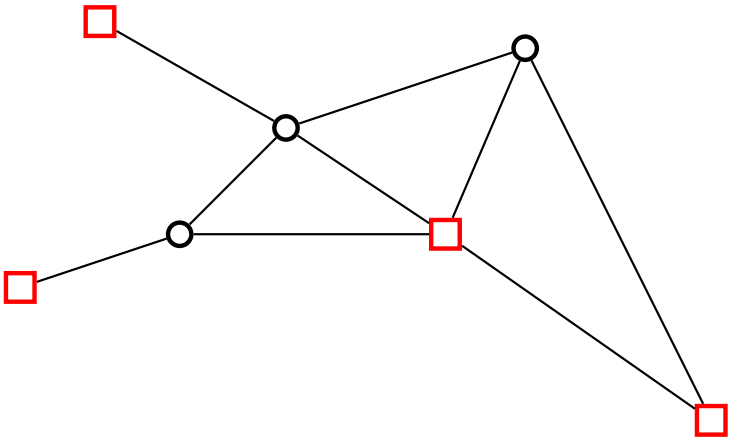
SCIP-Jack combines variety of techniques and algorithms in branch-and-cut framework, e.g.

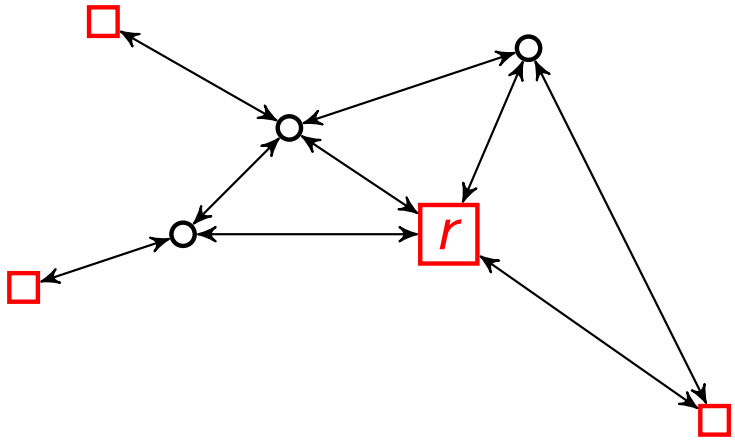
- preprocessing routines
- decomposition methods
- cutting-plane separation for LP-relaxation
- dynamic-programming algorithms
- primal heuristics
- domain propagation
- customized branching
- ...

Implemented as an extension of SCIP. Further contributers: G. Gamrath, T. Koch, S. Maher, M. Winkler.

Solving by integer programming

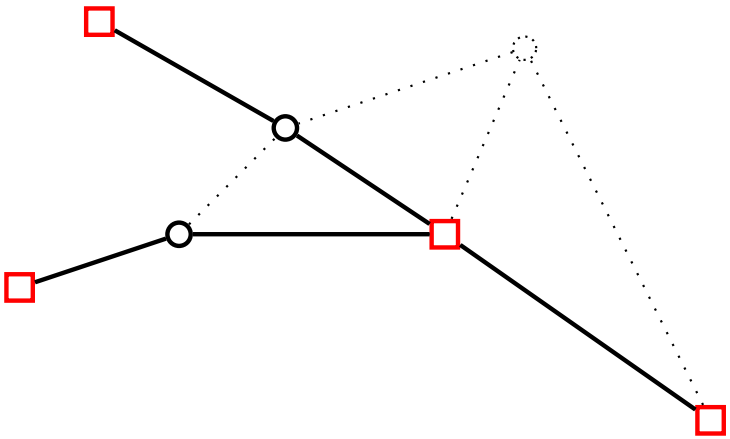
First, transform SPG instance to equivalent (bi-)directed Steiner tree problem.

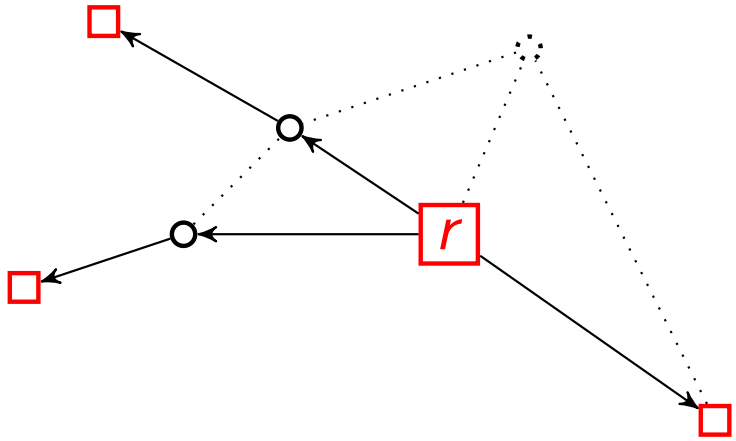




Solving by integer programming

First, transform SPG instance to equivalent (bi-)directed Steiner tree problem.





Integer programming formulation

Use well-known Flow-Balance Directed-Cut formulation ($BDCut_{FB}$):

Formulation (Wong, 1984; Duin, 1993)

$$
\begin{aligned}
\min\ & c^T y & & & \\
y(\delta^+(W)) &\geqslant 1 && \text{for all } W \subset V, r \in W, T \nsubseteq W \\
y(\delta^-(v)) &\leqslant y(\delta^+(v)) && \text{for all } v \in V \setminus T \\
y(a) &\in \{0,1\} && \text{for all } a \in A
\end{aligned}
$$

Integer programming formulation

Use well-known Flow-Balance Directed-Cut formulation ($BDCut_{FB}$):

Formulation (Wong, 1984; Duin, 1993)

$$
\begin{array}{rcll}
\min c^T y & & & \\
y(\delta^+(W)) & \geqslant & 1 & \text{for all } W \subset V, r \in W, T \nsubseteq W \\
y(\delta^-(v)) & \leqslant & y(\delta^+(v)) & \text{for all } v \in V \setminus T \\
y(a) & \in & \{0,1\} & \text{for all } a \in A
\end{array}
$$

We solve the LP-relaxation of $BDCut_{FB}$ with a specialized max-flow based separation algorithm.

New theoretical results (R., Franz, Koch; Networks, '22)

Theorem

If $|T| < 4$, then $BDCut_{FB}$ has tight LP relaxation.

For unweighted Steiner tree problem ($c \equiv 1$):

Theorem

If no four independent vertices exist, then BDCut has tight LP relaxation.

Proposition

BDCut is strictly stronger than node-separator formulation by Fischetti et al. (2017)—relative gap between LP relaxations up to 2.

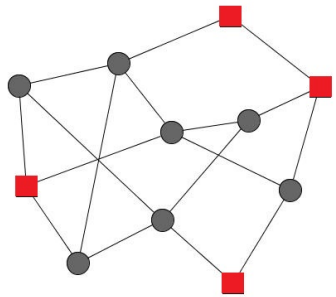
Reduction techniques

- aim to transform an instance to an equivalent smaller one (e.g. by deleting edges or vertices)
- can often significantly reduce problem size
- SCIP-Jack includes several new, provably stronger, reduction techniques
  (R., Koch, SIAM J. Opt. '18; R., Koch, Math. Prog '21)

Reduction techniques

- aim to transform an instance to an equivalent smaller one (e.g. by deleting edges or vertices)
- can often significantly reduce problem size
- SCIP-Jack includes several new, provably stronger, reduction techniques
  (R., Koch, SIAM J. Opt. '18; R., Koch, Math. Prog '21)

original instance (5000 edges)



reduced instance (17 edges)

Computational results: PACE Challenge 2018

Results of SCIP-Jack with Soplex as LP-solver on exact tracks of PACE Challenge 2018[1] (previous SCIP-Jack version won Track B):

		SCIP-Jack-NEW		SCIP-Jack-PACE		Best other
Track	# instances	solved	avg. time [s]	solved	avg. time [s]	solved
A	100	**99**	**38**	94	111	95
B	100	**99**	**25**	92	132	77

Table: Updated computational results for PACE 2018 instances.

PACE competition criteria:

1. number of solved instances within 30 minutes
2. arithmetic mean time (in case of ties)

[1] dedicated to fixed-parameter tractable SPG instances, 40 participating teams from 16 countries

Computational results (2)

14 benchmark test-sets from the literature (almost all non-trivial sets from DIMACS Challenge, SteinLib library). 24 hours time limit.

SCIP-Jack performs better on 13 of the 14 sets.

		# solved		mean time (sh. geo. mean)			maximum time		
Test-set	#	P.&V.	S.-J.	P.&V. [s]	S.-J. [s]	speedup	P.&V. [s]	S.-J. [s]	speedup
TSPFST	76	76	76	1.5	1.1	**1.36**	1161.4	326.9	**3.55**
VLSI	116	116	116	0.5	0.8	0.63	53.9	81.9	0.66
WRP4	62	62	62	3.2	2.4	**1.33**	106.1	95.2	**1.11**
2R	27	27	27	5.0	2.6	**1.92**	43.9	12.1	**3.63**
vienna-a	85	85⋆	85	7.2	5.0	**1.44**	441.3	57.1	**7.73**
vienna-s	85	85⋆	85	7.8	5.9	**1.32**	623.5	57.7	**10.81**
WRP3	63	63	63	22.8	13.4	**1.70**	6073.2	4568.1	**1.33**
GEO-a	23	23⋆	23	158.7	55.8	**2.84**	6476.5	852.4	**7.60**
GEO-org	23	23⋆	23	145.6	59.4	**2.45**	4385.0	834.4	**5.26**
ES10000	1	1	1	138.0	80.9	**1.71**	138.0	80.9	**1.71**
Cophag14	21	20⋆	**21**	27.7	14.8	**1.87**	>86400	4182.7	>**20.66**
SP	8	6	**8**	159.4	30.2	**5.28**	>86400	1892.1	>**45.66**
LIN	37	35	**36**	31.3	15.3	**2.05**	>86400	>86400	1.00
PUC	50	17⋆	**18**	14964.9	11568.3	**1.29**	>86400	>86400	1.00

Shift $s = 10$ used for shifted geometric mean. Results of (non-public) solver by Polzin, Vahdati are scaled.
Times marked by a ⋆ were obtained by P.&V. with (specialized) non-default settings.

Consider aggregated results, according to hardness of instances: Group $\geqslant Ks$ contains all instances solved by at least one solver in at least K seconds and by one solver in at most 24 hours.

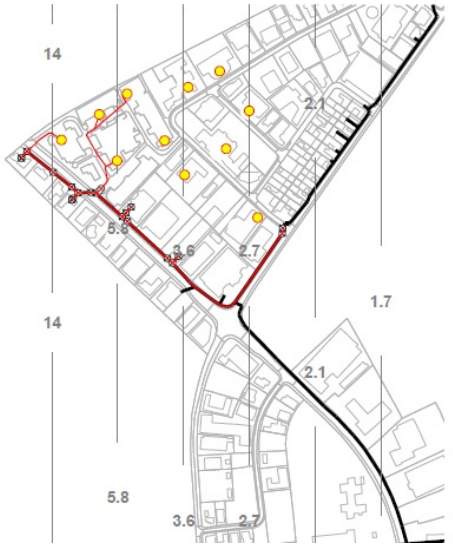
		shifted geometric mean time			arithmetic mean time		
Group	#	P.&V. [s]	S.-J. [s]	speedup	P.&V. [s]	S.-J. [s]	speedup
$\geqslant$0s	644	12.2	7.9	**1.54**	1235.5	229.0	**5.40**
$\geqslant$1s	342	34.5	19.5	**1.77**	2326.4	431.2	**5.40**
$\geqslant$10s	178	125.4	52.6	**2.38**	4466.6	825.5	**5.41**
$\geqslant$100s	66	1403.2	295.0	**4.76**	11999.0	2197.6	**5.46**
$\geqslant$1000s	30	8035.8	1099.0	**7.31**	25923.1	4653.8	**5.57**

Table: Computational comparison of our solver (*S.-J.*) and solver by Polzin, Vahdati (*P.&V.*).

Chapter 4: A generalization: The prize-collecting Steiner tree problem

Given:

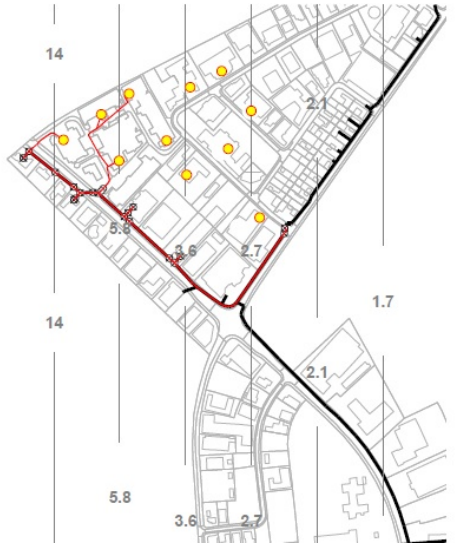
- undirected graph (V, E)
- $p \in \mathbb{R}^V_{\geqslant 0}$: vertex weights (prizes)
- $c \in \mathbb{R}^E_{\geqslant 0}$: edge costs



Chapter 4: A generalization: The prize-collecting Steiner tree problem

Given:

- undirected graph (V, E)
- $p \in \mathbb{R}^V_{\geqslant 0}$: vertex weights (prizes)
- $c \in \mathbb{R}^E_{\geqslant 0}$: edge costs



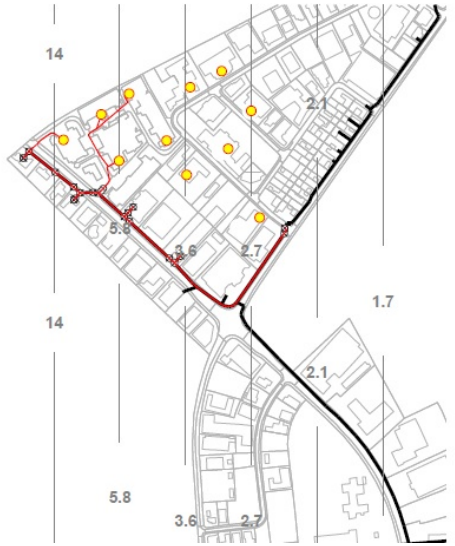
Prize-collecting Steiner tree problem

Find tree S that maximizes $\sum_{v \in V(S)} p(v) - \sum_{e \in E(S)} c(e)$.

Chapter 4: A generalization: The prize-collecting Steiner tree problem

Given:

- undirected graph (V, E)
- $p \in \mathbb{R}^V_{\geqslant 0}$: vertex weights (prizes)
- $c \in \mathbb{R}^E_{\geqslant 0}$: edge costs



Prize-collecting Steiner tree problem

Find tree S that maximizes $\sum_{v \in V(S)} p(v) - \sum_{e \in E(S)} c(e)$.

- various real-world applications (e.g., in computational biology, electricity planning, machine learning)
- many new algorithms/solvers introduced in the last years

New complexity results

Let $T_p^+ := \{t \in V : p(t) > \min_{e \in \delta(t)} c(e)\}$[2], called *proper potential terminals*.

Theorem (R., Koch; INFORMS J. Comp. '21)

Any prize-collecting Steiner tree instance (V, E, p, c) *can be solved in time*

$$O(3^{s^+} n + 2^{s^+} n^2 + n^2 \log n + mn),$$

where $m := |E|, n := |V|, s^+ := |T_p^+|$.

Roughly speaking:
If the number of high-profit vertices is limited, then the prize-collecting Steiner tree problem can be solved in polynomial time.

[2] see, e.g., Fischetti et al., Math. Prog. Comp. '17

New complexity results

Let $T_p^+ := \{t \in V : p(t) > \min_{e \in \delta(t)} c(e)\}$[2], called *proper potential terminals*.

Theorem (R., Koch; INFORMS J. Comp. '21)

Any prize-collecting Steiner tree instance (V, E, p, c) *can be solved in time*

$$O(3^{s^+} n + 2^{s^+} n^2 + n^2 \log n + mn),$$

where $m := |E|, n := |V|, s^+ := |T_p^+|$.

Roughly speaking:
If the number of high-profit vertices is limited, then the prize-collecting Steiner tree problem can be solved in polynomial time.

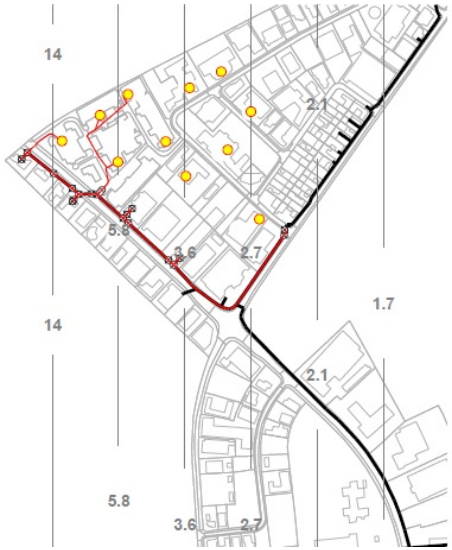
For practical solving we use a branch-and-cut framework again.

[2] see, e.g., Fischetti et al., Math. Prog. Comp. '17

Computational results

Max. runtime on fiber optic network instances (Cologne), up to 20 000 edges:

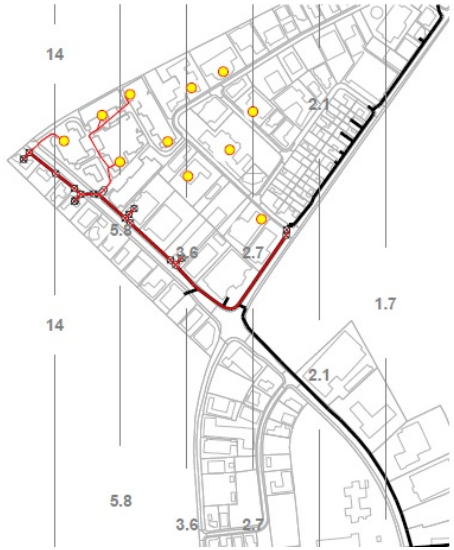
- Original publication (Ljubic et al., Math. Prog. '06): > 24 h
- Best result DIMACS Challenge 2014 (SCIP-Jack, faster machine): 112 sec.
- Latest SCIP-Jack: < 0.5 sec.



Computational results

Max. runtime on fiber optic network instances (Cologne), up to 20 000 edges:

- Original publication (Ljubic et al., Math. Prog. '06): > 24 h
- Best result DIMACS Challenge 2014 (SCIP-Jack, faster machine): 112 sec.
- Latest SCIP-Jack: < 0.5 sec.

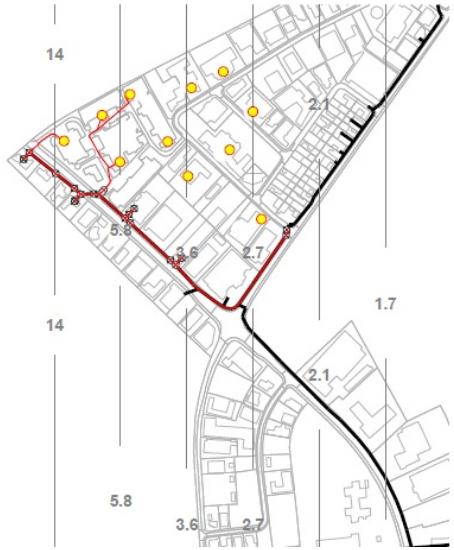


Runtime of current best other solver (Leitner et. al, IJOC '18): < 3 sec.

Computational results

Max. runtime on fiber optic network instances (Cologne), up to 20 000 edges:

- Original publication (Ljubic et al., Math. Prog. '06): > 24 h
- Best result DIMACS Challenge 2014 (SCIP-Jack, faster machine): 112 sec.
- Latest SCIP-Jack: < 0.5 sec.



Runtime of current best other solver (Leitner et. al, IJOC '18): < 3 sec.

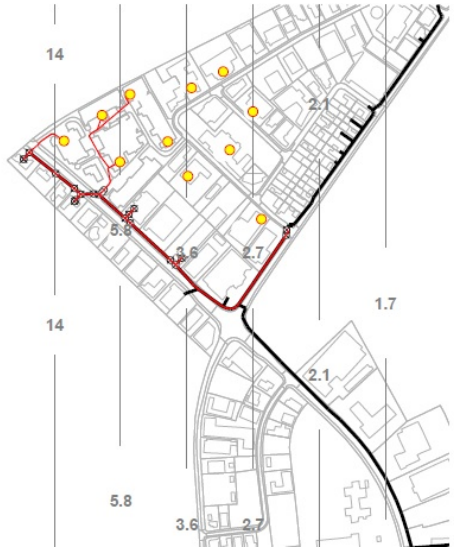
Largest instances from literature (Sun et al, IEEE/ACM TON '19): 10 million edges; 20 instances. In 24 hours:

- Leitner et. al.: None solved

Computational results

Max. runtime on fiber optic network instances (Cologne), up to 20 000 edges:

- Original publication (Ljubic et al., Math. Prog. '06): > 24 h
- Best result DIMACS Challenge 2014 (SCIP-Jack, faster machine): 112 sec.
- Latest SCIP-Jack: < 0.5 sec.



Runtime of current best other solver (Leitner et. al, IJOC '18): < 3 sec.

Largest instances from literature (Sun et al, IEEE/ACM TON '19): 10 million edges; 20 instances. In 24 hours:

- Leitner et. al.: None solved
- SCIP-Jack: 16 solved, sh. geo. mean time: 9 296 sec. (R., Koch, IJOC '21)

Chapter 5: Further related problems

Euclidean Steiner tree problem (Chapter 5.4): Here SPG can be applied instead of commonly used hypergraphic MST concatenation.

Comparison on largest instances from literature with state-of-the-art geometric Steiner tree solver GeoSteiner 5.1 (Juhl et al., 2018):

Chapter 5: Further related problems

Euclidean Steiner tree problem (Chapter 5.4): Here SPG can be applied instead of commonly used hypergraphic MST concatenation.

Comparison on largest instances from literature with state-of-the-art geometric Steiner tree solver GeoSteiner 5.1 (Juhl et al., 2018):

1) 15 instances with 50 000 terminals in the plane:
   - GeoSteiner solves 8 within one week
   - SCIP-Jack solves all 15 in three minutes

Chapter 5: Further related problems

Euclidean Steiner tree problem (Chapter 5.4): Here SPG can be applied instead of commonly used hypergraphic MST concatenation.

Comparison on largest instances from literature with state-of-the-art geometric Steiner tree solver GeoSteiner 5.1 (Juhl et al., 2018):

1) 15 instances with 50 000 terminals in the plane:
   - GeoSteiner solves 8 within one week
   - SCIP-Jack solves all 15 in three minutes

2) 15 instances with 100 000 terminals in the plane:
   - GeoSteiner solves 3 within one week
   - SCIP-Jack solves all 15 in 13 minutes

$\rightarrow$ 19 instances solved for first time to optimality
(R., Koch; Math. Prog. '21)

SCIP-Jack

SCIP-Jack:

- is fastest solver for all 15 problem classes it can handle
- is freely available for academic use:
  `https://scipjack.zib.de`
- has been used by several industry customers, for example to design fiber optic and district heating networks...

Quadratic unconstrained binary optimization (QUBO)

Problem: Given $Q \in \mathbb{R}^{n \times n}$, find optimal solution x to

$$\begin{aligned} \min \ & x^T Q x \\ & x \in \{0,1\}^n \end{aligned}$$

Equivalently: Find maximum-cut in edge-weighted, undirected graph (Max-Cut problem).

Quadratic unconstrained binary optimization (QUBO)

Problem: Given $Q \in \mathbb{R}^{n \times n}$, find optimal solution x to

$$\begin{aligned} \min \ & x^T Q x \\ & x \in \{0,1\}^n \end{aligned}$$

Equivalently: Find maximum-cut in edge-weighted, undirected graph (Max-Cut problem).

Currently, there is great interest in these problem classes (both in academia and industry), since heuristic solution methods for QUBOs on Quantum annealers (e.g., D-Wave) exist.

Building an exact QUBO solver

Together with Thorsten Koch and Yuji Shinano:
Developing fastest exact QUBO/Max-Cut solver (for digital computers). Main focus on sparse instances.

Building an exact QUBO solver

Together with Thorsten Koch and Yuji Shinano:
Developing fastest exact QUBO/Max-Cut solver (for digital computers). Main focus on sparse instances.

Current state: Branch-and-cut solver based on SCIP;
main components:

- Simple presolving
- Simple domain propagation
- Problem-specific cutting planes (optimized implementation)
- Primal heuristics
- Parallelization via UG framework (still experimental)

Building an exact QUBO solver

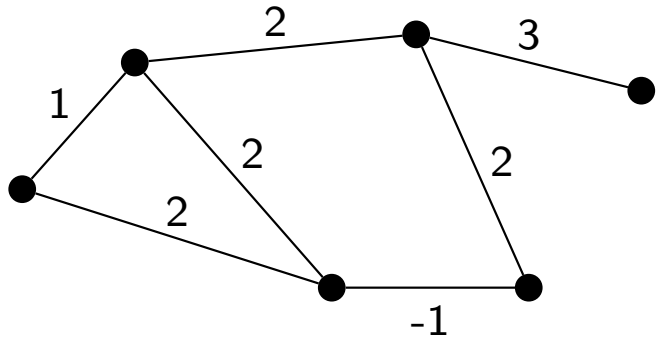
Together with Thorsten Koch and Yuji Shinano:
Developing fastest exact QUBO/Max-Cut solver (for digital computers). Main focus on sparse instances.

Current state: Branch-and-cut solver based on SCIP;
main components:

- Simple presolving
- Simple domain propagation
- Problem-specific cutting planes (optimized implementation)
- Primal heuristics
- Parallelization via UG framework (still experimental)

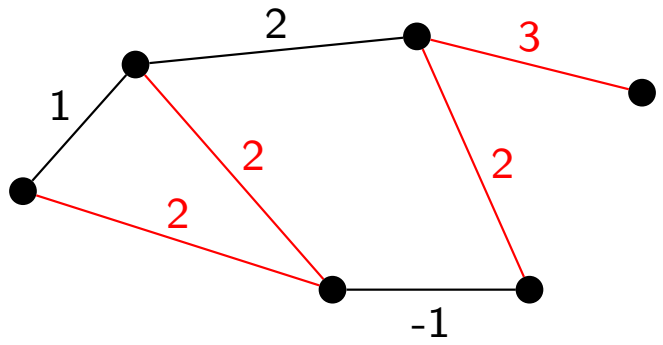
Maximum-cut problem (reminder)

Find maximum-cut in edge-weighted, undirected graph:



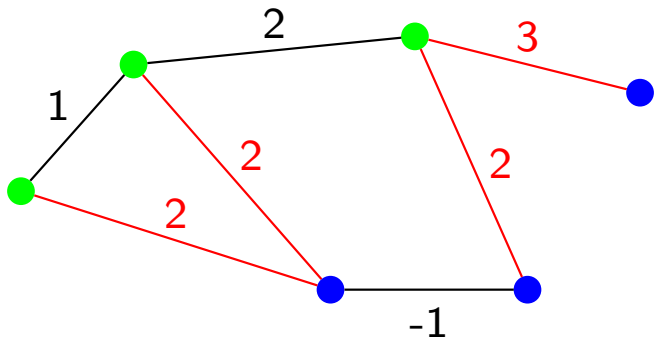
Maximum-cut problem (reminder)

Find maximum-cut in edge-weighted, undirected graph:



Maximum-cut problem (reminder)

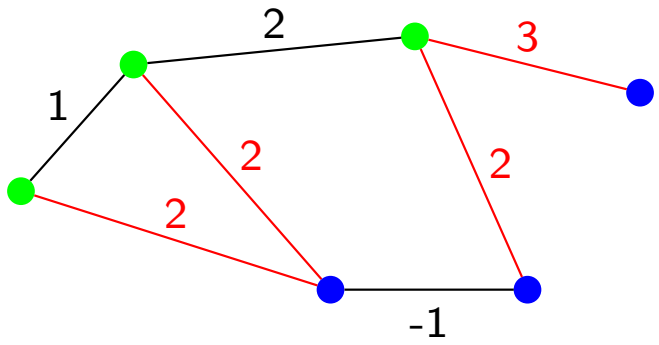
Find maximum-cut in edge-weighted, undirected graph:



QUBO with dimension n is equivalent to Max-Cut problem on undirected graph with $n + 1$ vertices.

Maximum-cut problem (reminder)

Find maximum-cut in edge-weighted, undirected graph:



QUBO with dimension n is equivalent to Max-Cut problem on undirected graph with $n + 1$ vertices.

In the following, we consider mostly the maximum-cut formulation.

Odd-cycle cuts

Observation: For any edge cut $\delta(U)$ and any cycle C, let $F := C \cap \delta(U)$ be their common edges. $|F|$ is even.

Odd-cycle cuts

Observation: For any edge cut $\delta(U)$ and any cycle C, let $F := C \cap \delta(U)$ be their common edges. $|F|$ is even.

Formulation (Barahona and Mahjoub, '89)

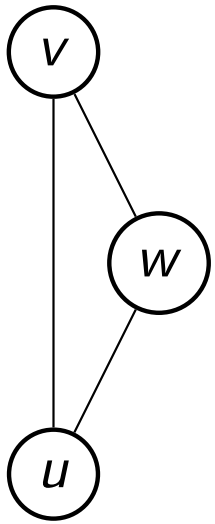
$$
\begin{aligned}
& max \quad w^T x \\
s.t. \quad \sum_{e \in F} x(e) - \sum_{e \in C \setminus F} x(e) \quad & \leqslant \quad |F| - 1 \text{ for all cycles } C, F \subseteq C, |F| \text{ odd} \\
x(e) \quad & \in \quad \{0,1\} \quad \text{for all } e \in E.
\end{aligned}
$$

$\rightarrow$ We cannot have an odd intersection of a cycle and a cut.

Separation of the odd-cycle constraints

Rewrite the odd-cycle constraints as

$$\sum_{e \in F} (1 - x(e)) + \sum_{e \in C \setminus F} x(e) \geq 1 \quad \text{for all cycles } C, F \subseteq C, |F| \text{ odd.}$$



(e) Original graph.

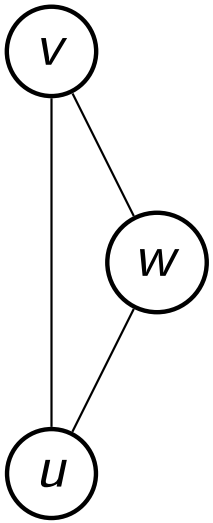
(f) Auxiliary graph.

Separation of the odd-cycle constraints

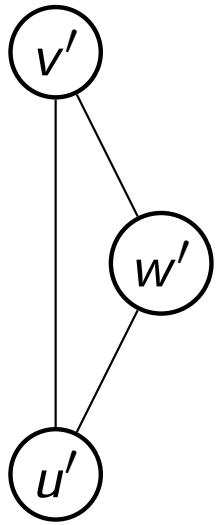
Rewrite the odd-cycle constraints as

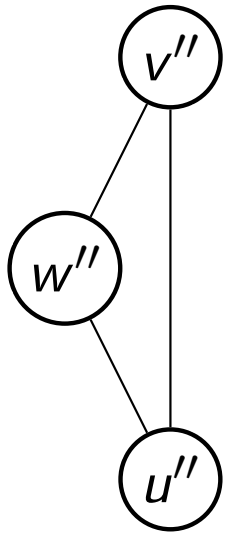
$$\sum_{e \in F} (1 - x(e)) + \sum_{e \in C \setminus F} x(e) \geq 1 \quad \text{for all cycles } C, F \subseteq C, |F| \text{ odd.}$$

1) Add copy of the graph



(g) Original graph.





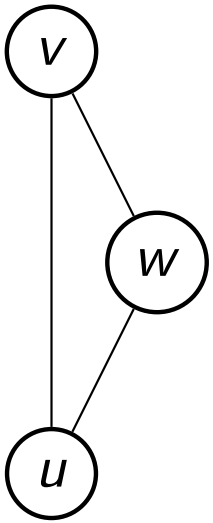
(h) Auxiliary graph.

Separation of the odd-cycle constraints

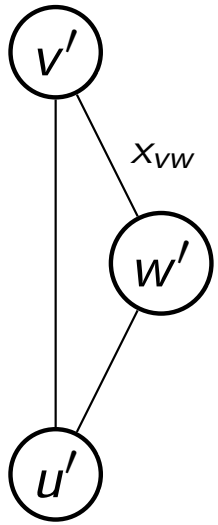
Rewrite the odd-cycle constraints as

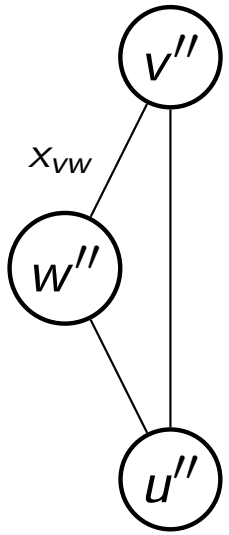
$$\sum_{e \in F} (1 - x(e)) + \sum_{e \in C \setminus F} x(e) \geq 1 \quad \text{for all cycles } C, F \subseteq C, |F| \text{ odd.}$$

1b) Set edge weights to LP value x



(i) Original graph.





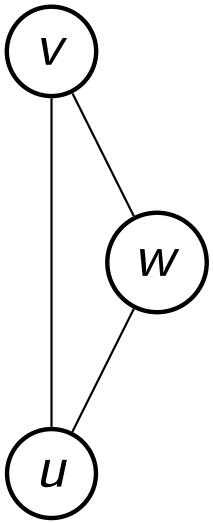
(j) Auxiliary graph.

Separation of the odd-cycle constraints

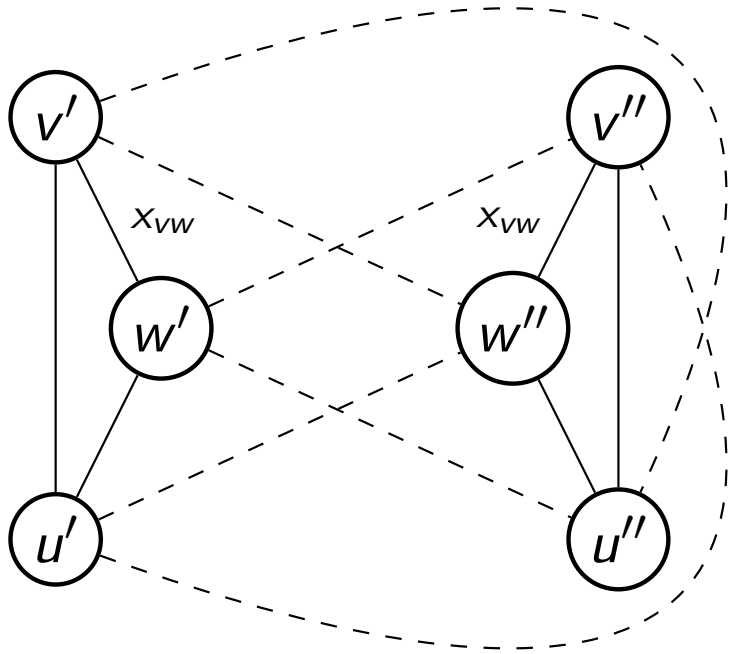
Rewrite the odd-cycle constraints as

$$\sum_{e \in F} (1 - x(e)) + \sum_{e \in C \setminus F} x(e) \geq 1 \quad \text{for all cycles } C, F \subseteq C, |F| \text{ odd.}$$

2) Add cross edges for each original edge



(k) Original graph.



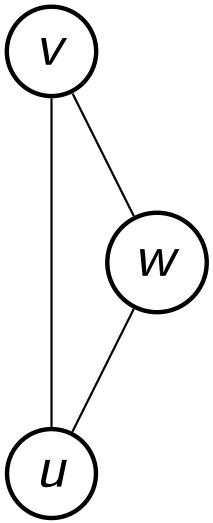
(l) Auxiliary graph.

Separation of the odd-cycle constraints

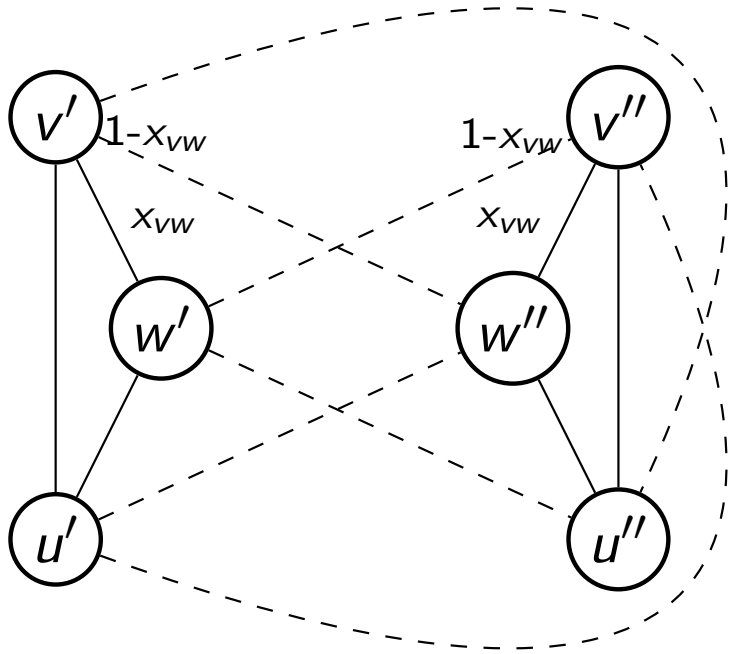
Rewrite the odd-cycle constraints as

$$\sum_{e \in F} (1 - x(e)) + \sum_{e \in C \setminus F} x(e) \geq 1 \quad \text{for all cycles } C, F \subseteq C, |F| \text{ odd.}$$

2b) Set edges weight to $1 - x_{vw}$ for cross edges $\{v', w''\}, \{v'', w'\}$



(m) Original graph.



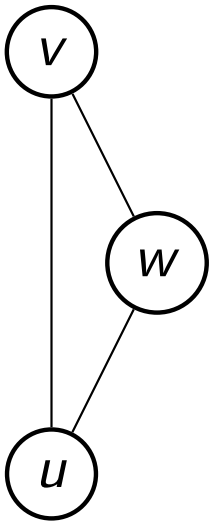
(n) Auxiliary graph.

Separation of the odd-cycle constraints

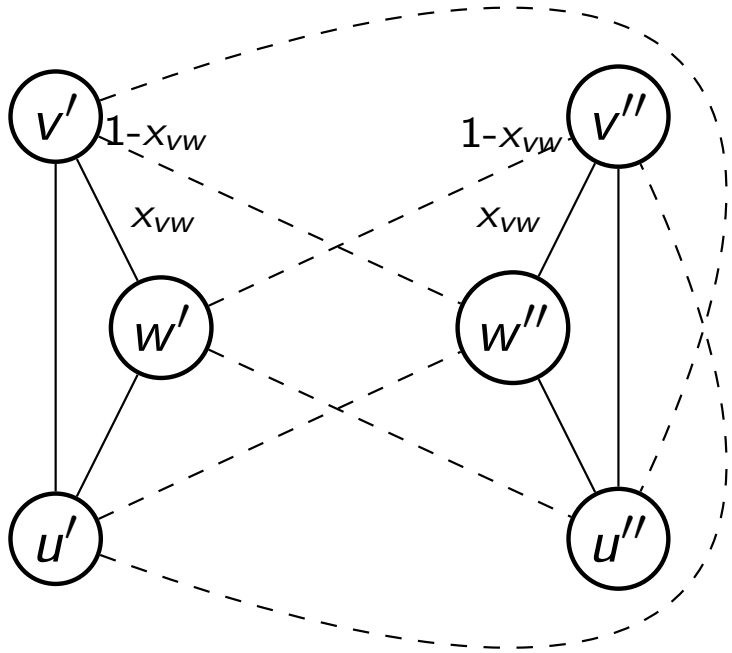
Rewrite the odd-cycle constraints as

$$\sum_{e \in F} (1 - x(e)) + \sum_{e \in C \setminus F} x(e) \geq 1 \quad \text{for all cycles } C, F \subseteq C, |F| \text{ odd.}$$

3) Compute shortest-path between all twin nodes (e.g., v', v''); distance < 1 corresponds to violated constraint



(o) Original graph.



(p) Auxiliary graph.

Speeding-up the shortest-paths computation

Various papers from the literature: *The shortest-paths computation takes too long to be practical.* $\rightarrow$ only heuristic cuts are used

Speeding-up the shortest-paths computation

Various papers from the literature: *The shortest-paths computation takes too long to be practical.* $\rightarrow$ only heuristic cuts are used

Well, it depends...possible speed-ups:

- use optimized shortest-path implementation
- don't add vertices with distance $\geqslant 1$ to priority queue
- don't continue from vertices v' if v'' is labeled and summed distances $\geqslant 1$
- remove edges with weight 1
- contract edges with weight 0

A new reduction method

Proposition (R., Koch, Shinano; 2022)

Assume there is a triangle in G with edges $\{v_1, v_2\}$, $\{v_1, v_3\}$, $\{v_2, v_3\} \in E$ such that $w(\{v_1, v_2\}) > 0$, $w(\{v_1, v_3\}) > 0$, and $w(\{v_2, v_3\}) < 0$. Let $V_1 \subset V$ such that $\{v_1, v_2\}, \{v_1, v_3\} \in \delta(V_1)$ and let $V_2 \subset V$ such that $\{v_1, v_2\}, \{v_2, v_3\} \in \delta(V_2)$. If

$$w(\{v_1, v_2\}) + w(\{v_1, v_3\}) \geqslant \sum_{e \in \delta(V_1) \setminus \{\{v_1, v_2\}, \{v_1, v_3\}\}} |w(e)|,$$

and

$$w(\{v_1, v_2\}) - w(\{v_2, v_3\}) \geqslant \sum_{e \in \delta(V_2) \setminus \{\{v_1, v_2\}, \{v_1, v_3\}\}} |w(e)|,$$

then there is a maximum-cut vector $x \in \{0, 1\}^E$ such that $x(\{v_1, v_2\}) = 1$.

Parallelization (Joint work with Yuji Shinano)

- For parallelization of branch-and-bound search we use the UG framework
- We use dedicated threads (processes) for running primal heuristics
- We extended UG to be able to use idle threads during the branch-and-bound root node computation for running primal heuristics
- Some other tricks...

Computational results: Separation time

Name	#	Sepa-time [%]	LP-time [%]
BE120.3	10	2.5	79.3
BE250	10	2.8	84.3
BQP100	10	3.0	32.0
BQP250	10	2.6	86.2
$GKA_{a\text{-}d}$	35	6.7	49.1
IsingChain	30	0.0	0.0
K64-chimera	80	10.1	58.3
Kernel	14	1.3	4.8
$PM1s_{100}$	10	13.1	78.3
QPLIB	22	18.1	65.5
Torus	18	16.7	15.5
$W01_{100}$	10	12.0	59.3

Table: Average times spent in separation and (re-) optimization of the LP for Max-Cut and QUBO test-sets. CPLEX 12.10 used as LP solver.

Computational results: Comparison with McSparse

McSparse: Best other sparse Max-Cut/QUBO solver (Charfreitag, Jünger, Mallach, Mutzel; 2022). Results on set of 14 representative instances, selected by Charfreitag et. al.[3]. Max-Cut results:

			# B&B nodes		run time	
Name	|V|	|E|	MS	new	MS [s]	new [s]
pm1s_100.3	100	495	341	737	48.2	48.0
pw01_100.0	100	495	171	179	20.0	8.5
mannino_k487b	487	5391	1	15	167.3	2.9
bio-diseasome	516	1188	1	1	9.5	0.6
ca-netscience	379	914	1	1	0.1	0.0
g000981	110	188	1	1	0.0	0.0
imgseg_138032	12736	23664	1	1	30.5	4.4

[3]McSparse is not publicly available, a faster machine than ours was used to produce the results

Computational results: Comparison with McSparse (2)

QUBO results:

			# B&B nodes		run time	
Name	*n*	*nnz*	MS	new	MS [s]	new [s]
bqp250-3	250	3092	25	15	414.1	85.8
gka2c	50	813	1	1	0.5	0.3
gka4d	100	2010	129	71	219.6	61.9
gka5c	80	721	1	1	0.1	0.1
gka7a	30	241	1	1	0.0	0.0
be120.3.5	120	2248	111	63	257.7	62.4
be250.3	250	3277	107	81	841.0	212.0

Benchmark test-sets from the literature

Name	#	|V|	|E|	Description
DIMACS	4	512 - 3375	1536-10125	Introduced at 7th DIMACS Challenge.
Mannino	4	48 - 487	1128-8511	From a radio frequency assignment problem.
$PM1s_{100}$	10	100	495	Randomly generated.
$W01_{100}$	10	100	495	Randomly generated.
K64-chimera	80	2049	8064	Instances on D-Wave Chimera graphs.
Kernel	14	33 - 2888	91-2981	From various sources.
IsingChain	30	100 - 300	4950-44850	From an application in statistical physics.
Torus	18	100 - 343	200-1029	2D and 3D torus instances from an application in statistical physics.

Name	#	n	nnz	Description
QPLIB	22	120 - 1225	602-34876	All QUBO instances from the QPLIB.
BQP100	10	100	471-528	Randomly generated.
BQP250	10	250	3039-3208	Randomly generated.
BE120.3	10	120	2176-2253	Randomly generated.
BE250	10	250	3268-3388	Randomly generated.
$GKA_{a\text{-}d}$	35	20 - 125	204-7788	Randomly generated instances with different densities.

Computational results: Comparison with Gurobi

Single-thread comparison of Gurobi 9.5 (*Grb*) and our solver (*new*):

		# solved		mean time (sh. geo. mean)			maximum time		
Test-set	#	Grb	new	Grb [s]	new [s]	speedup	Grb [s]	new [s]	speedup
$PM1s_{100}$	10	10	10	192.3	21.0	**9.16**	303.3	48.6	**6.24**
$W01_{100}$	10	10	10	44.1	3.1	**14.23**	97.1	21.4	**4.54**
Kernel	14	14	14	0.7	0.1	**7.00**	14.3	1.1	**13.00**
IsingChain	30	30	30	1.3	<0.05	>**26.00**	41.0	<0.05	>**820.00**
Torus	18	18	18	3.8	0.4	**9.50**	628.0	7.6	**82.63**
KChimera	80	80	80	90.1	1.5	**60.07**	195.4	8.6	**22.72**
QPLIB	22	8	**13**	687.4	173.3	**3.97**	3600	3600	1.00
BQP250	10	0	**7**	3600	654.1	**5.50**	3600	3600	1.00
BE120.3	10	9	**10**	265.6	60.2	**4.41**	3600	820.0	>**4.40**
BE250	10	0	**8**	3600	609.3	**5.91**	3600	3600	1.00
$GKA_{a\text{-}d}$	35	29	**31**	6.5	6.0	**1.08**	3600	3600	1.00

Computational results: Comparison with Gurobi

4-threads comparison of Gurobi 9.5 (*Grb*) and our solver (*new*):

		# solved		mean time (sh. geo. mean)			maximum time		
Test-set	#	Grb	new	Grb [s]	new [s]	speedup	Grb [s]	new [s]	speedup
$PM1s_{100}$	10	10	10	244.6	3.0	**81.53**	527.6	24.3	**21.71**
$W01_{100}$	10	10	10	54.8	2.1	**26.10**	129.3	12.2	**10.60**
Kernel	14	14	14	1.1	0.1	**11.00**	23.1	1.2	**19.25**
IsingChain	30	30	30	2.6	<0.05	>**52.00**	49.8	0.1	**498.00**
Torus	18	18	18	5.3	0.6	**8.83**	239.4	10.0	**23.94**
KChimera	80	80	80	54.6	1.5	**36.40**	328.7	8.6	**38.22**
QPLIB	22	9	**13**	728.6	137.6	**5.30**	3600	3600	1.00
BQP250	10	0	**7**	3600	657.2	>**5.48**	3600	3600	1.00
BE120.3	10	10	10	501.8	8.6	**58.35**	3212.2	17.2	**186.76**
BE250	10	1	**8**	3312.9	606.3	**5.46**	3600	3600	1.00
$GKA_{a\text{-}d}$	35	29	**35**	8.3	2.0	**4.15**	3600	33.8	>**106.51**

Newly solved instances

Name	gap [%]	new primal	previous primal
torusg3-15	**opt**	**286626481**	282534518
torusg3-15⋆	**opt**	**292031950**	-
toruspm3-15-50	1.8	3010	2968
toruspm3-15-50⋆	1.8	3008	-
QPLIB_3693	1.3	-1152	-1148
QPLIB_3850	1.7	-1194	-1192

Table: Improved solutions for Max-Cut (first four) and QUBO (last two) benchmark instances.

For *torusg3-15* and *torusg3-15⋆* we used a machine with 88 cores of Intel Xeon E7-8880 v4 CPUs @ 2.20GHz. We ran the two instances (non-exclusively) for at most 3 days while using 80 threads. The rest is done single-threaded in one hour.

So how about Quantum annealers?

From the article
Quantum Annealing versus Digital Computing: An Experimental Comparison, Jünger et al., ACM J. Exp. Algorithmics, 2021:

"However, we should stress the fact that exact optimization requires a lot of time to prove optimality, and thus it is not fair to compare their times with the heuristic times, but even with this additional burden, the exact algorithms are faster than D-Wave on a large portion of the sample".

So how about Quantum annealers?

From the article
Quantum Annealing versus Digital Computing: An Experimental Comparison, Jünger et al., ACM J. Exp. Algorithmics, 2021:

"However, we should stress the fact that exact optimization requires a lot of time to prove optimality, and thus it is not fair to compare their times with the heuristic times, but even with this additional burden, the exact algorithms are faster than D-Wave on a large portion of the sample".

- McSparse is faster than the (sparse) solver used by Jünger et. al.
- Our solver is faster than McSparse

So how about Quantum annealers?

From the article
Quantum Annealing versus Digital Computing: An Experimental Comparison, Jünger et al., ACM J. Exp. Algorithmics, 2021:

"However, we should stress the fact that exact optimization requires a lot of time to prove optimality, and thus it is not fair to compare their times with the heuristic times, but even with this additional burden, the exact algorithms are faster than D-Wave on a large portion of the sample".

- McSparse is faster than the (sparse) solver used by Jünger et. al.
- Our solver is faster than McSparse

So how about Quantum annealers?

From the article
Quantum Annealing versus Digital Computing: An Experimental Comparison, Jünger et al., ACM J. Exp. Algorithmics, 2021:

"However, we should stress the fact that exact optimization requires a lot of time to prove optimality, and thus it is not fair to compare their times with the heuristic times, but even with this additional burden, the exact algorithms are faster than D-Wave on a large portion of the sample".

- McSparse is faster than the (sparse) solver used by Jünger et. al.
- Our solver is faster than McSparse

$\rightarrow$ If you need to solve a QUBO, better save the money for a D-Wave machine and use our solver instead!

The end

Thank you for your attention!

Faster exact solution of sparse MaxCut and QUBO problems
Daniel Rehfeldt, Thorsten Koch, Yuji Shinano
doi: 10.48550/arXiv.2202.02305