Lifting and Separation Procedures for the Cut Polytope

Thorsten Bonato

Research Group Discrete and Combinatorial Optimization
University of Heidelberg

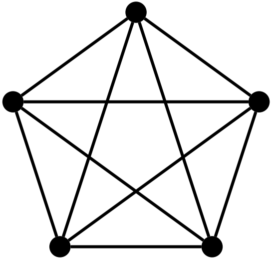
6th Conference of PhD Students in Computer Science
Szeged, July 3, 2008

Outline

Outline

Definition

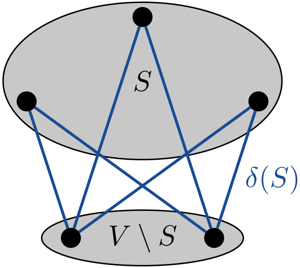
Let $G = (V, E, c)$ be an undirected weighted graph.



Definition

Let $G = (V, E, c)$ be an undirected weighted graph.

Any $S \subseteq V$ induces a set of edges $\delta(S)$ with exactly one endpoint in S. $\delta(S)$ is called a cut of G with shores S and $V \setminus S$.

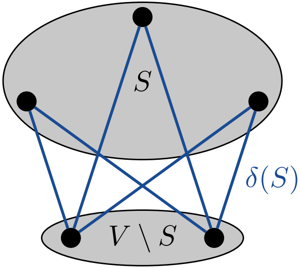


Definition

Let $G = (V, E, c)$ be an undirected weighted graph.

Any $S \subseteq V$ induces a set of edges $\delta(S)$ with exactly one endpoint in S. $\delta(S)$ is called a cut of G with shores S and $V \setminus S$.

Finding a cut with maximum aggregate edge weight is known as max-cut problem.

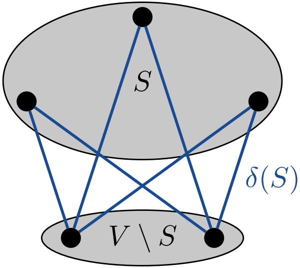


Definition

Let $G = (V, E, c)$ be an undirected weighted graph.

Any $S \subseteq V$ induces a set of edges $\delta(S)$ with exactly one endpoint in S. $\delta(S)$ is called a cut of G with shores S and $V \setminus S$.

Finding a cut with maximum aggregate edge weight is known as max-cut problem.

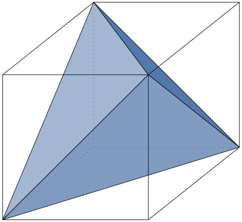


Applications

- quadratic +/–1 resp. 0/1 optimization,
- determining ground states of Ising spin glasses.

Cut polytope CUT(G)

Convex hull of the incidence vectors of all cuts of G.



$\mathrm{CUT}(K_3)$

Cycle polytope M(G)

Relaxation of the cut polytope with linear description

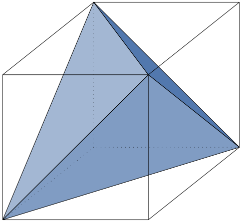
$$\begin{aligned} x(F) - x(C \setminus F) &\leq |F| - 1, && \text{for all } F \subseteq C, |F| \text{ odd}, \\ & && \text{for each cycle } C \text{ of } G, \\ x_e &\in [0,1], && e \in E. \end{aligned}$$

Cut polytope CUT(G)

Convex hull of the incidence vectors of all cuts of G.

Cycle polytope M(G)

Relaxation of the cut polytope with linear description



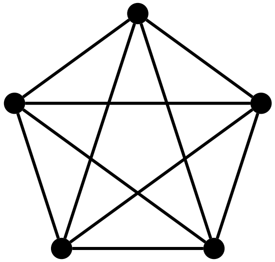
CUT(K_3)

$$\begin{array}{rcll} x(F) - x(C \setminus F) & \leq & |F| - 1, & \text{for all } F \subseteq C, |F| \text{ odd}, \\ & & & \text{for each cycle } C \text{ of } G, \\ x_e & \in & [0,1], & e \in E. \end{array}$$

CUT(G) and M(G) have exactly the same integral points.

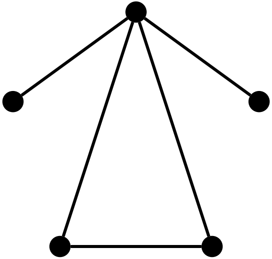
Algorithms

- Branch&Cut (possibly combined with Semidefinite Programming) as exact method,
- most techniques are associated with dense/complete graphs.



Algorithms

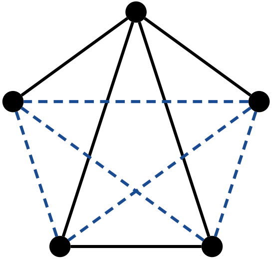
- Branch&Cut (possibly combined with Semidefinite Programming) as exact method,
- most techniques are associated with dense/complete graphs.



Handling sparse graphs

Algorithms

- Branch&Cut (possibly combined with Semidefinite Programming) as exact method,
- most techniques are associated with dense/complete graphs.



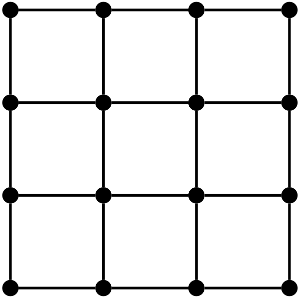
Handling sparse graphs

- **trivial approach**:
  artificial completion using edges with weight zero,
- **major drawback**:
  increase in number of variables/computational difficulty.

Outline

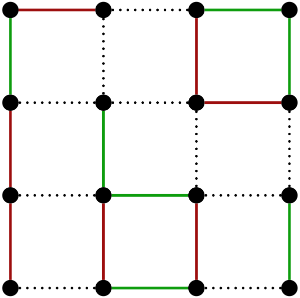
An example

(4×4)-grid with 16 nodes and 24 edges.



(4×4)-grid with 16 nodes and 24 edges.

W.r.t. a vector $z \in \mathsf{M}(G) \setminus \mathsf{CUT}(G)$ the edge set decomposes into:

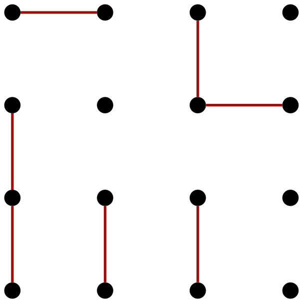


An example

(4×4)-grid with 16 nodes and 24 edges.

W.r.t. a vector $z \in \mathsf{M}(G) \setminus \mathsf{CUT}(G)$ the edge set decomposes into:

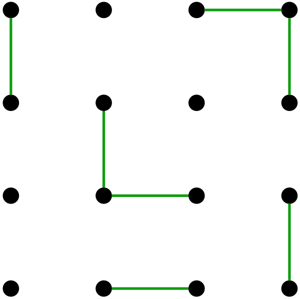
- 0-edges,



(4×4)-grid with 16 nodes and 24 edges.

W.r.t. a vector $z \in \mathsf{M}(G) \setminus \mathsf{CUT}(G)$ the edge set decomposes into:

- 0-edges,
- 1-edges,

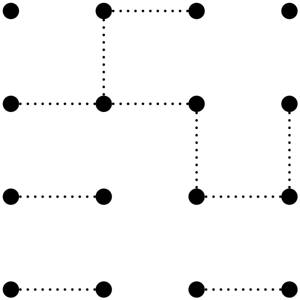


An example

(4×4)-grid with 16 nodes and 24 edges.

W.r.t. a vector $z \in \mathsf{M}(G) \setminus \mathsf{CUT}(G)$ the edge set decomposes into:

- 0-edges,
- 1-edges,
- fractional edges.

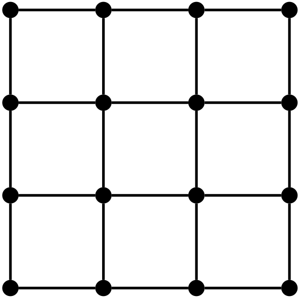


An example

(4×4)-grid with 16 nodes and 24 edges.

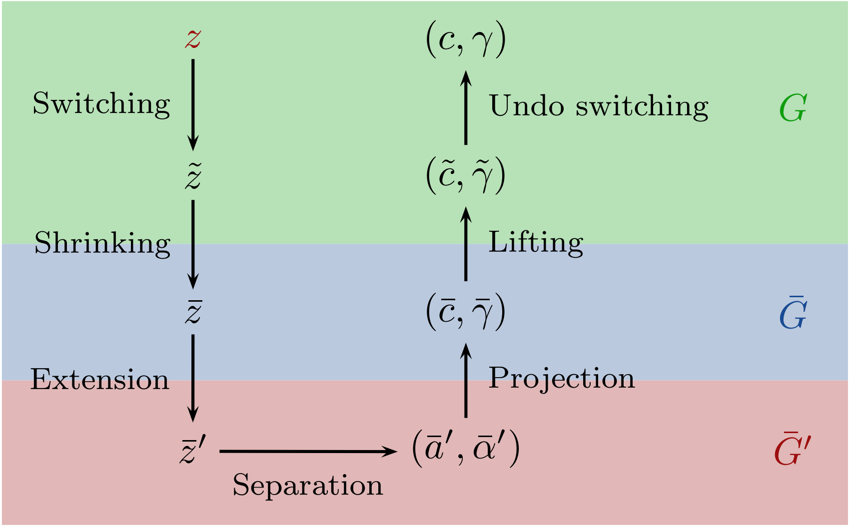
W.r.t. a vector $z \in \mathsf{M}(G) \setminus \mathsf{CUT}(G)$ the edge set decomposes into:

- 0-edges,
- 1-edges,
- fractional edges.



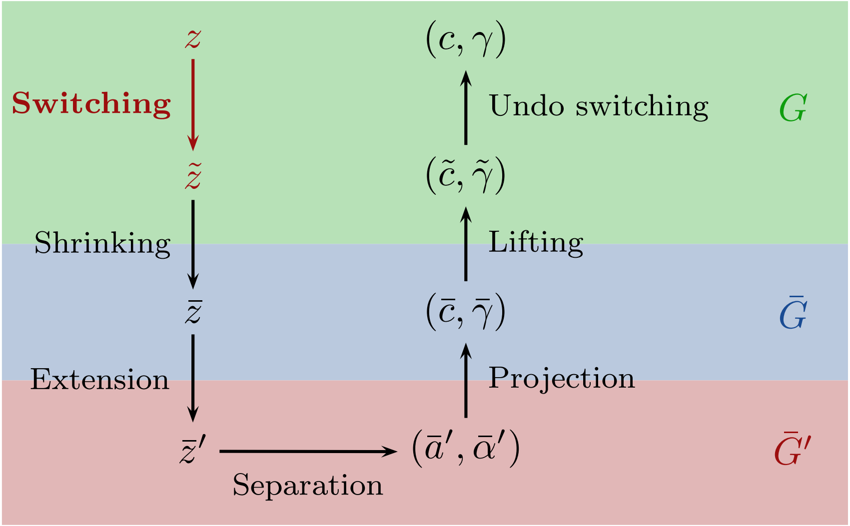
Artificial completion would require 96 additional edges.

Input: vector $z \in \mathsf{M}(G) \setminus \mathsf{CUT}(G)$.



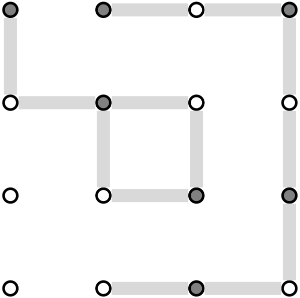
Outline of the shrinking approach

Transform 1-edges into 0-edges without affecting original 0-edges.



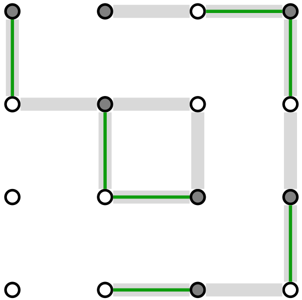
Switching

$z \in \mathrm{M}(G)$ implies existence of a cut that contains

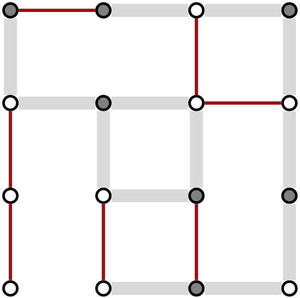


Switching

$z \in \mathrm{M}(G)$ implies existence of a cut that contains all 1-edges



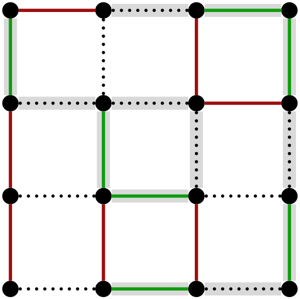
$z \in \mathrm{M}(G)$ implies existence of a cut that contains all 1-edges but no 0-edges.



$z \in \mathrm{M}(G)$ implies existence of a cut that contains all 1-edges but no 0-edges.

Switching z alongside this cut

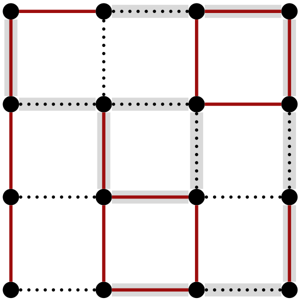
- only affects cut edges,



$z \in \mathrm{M}(G)$ implies existence of a cut that contains all 1-edges but no 0-edges.

Switching z alongside this cut

- only affects cut edges,
- transforms all 1-edges into 0-edges,

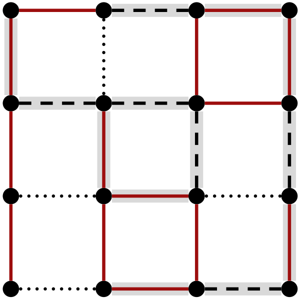


Switching

$z \in \mathrm{M}(G)$ implies existence of a cut that contains all 1-edges but no 0-edges.

Switching z alongside this cut

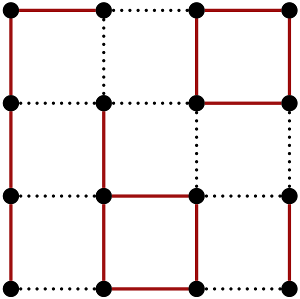
- only affects cut edges,
- transforms all 1-edges into 0-edges,
- may alter values of fractional edges.



$z \in \mathrm{M}(G)$ implies existence of a cut that contains all 1-edges but no 0-edges.

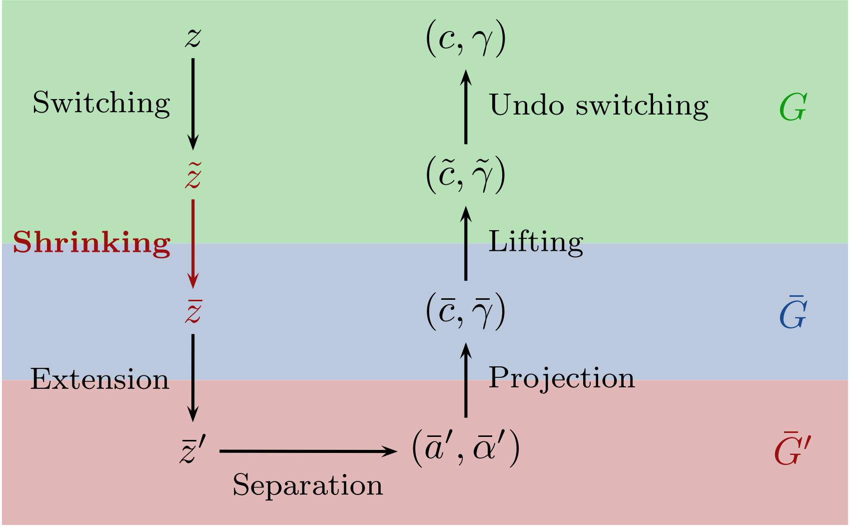
Switching z alongside this cut

- only affects cut edges,
- transforms all 1-edges into 0-edges,
- may alter values of fractional edges.

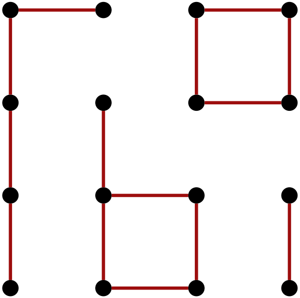


Switched vector $\tilde{z}$ has only fractional and 0-edges.

Shrink 0-edges.

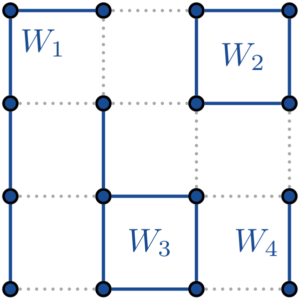


Consider the graph G_0 induced by the 0-edges of the switched vector $\tilde{z}$.



Consider the graph G_0 induced by the 0-edges of the switched vector $\tilde{z}$.

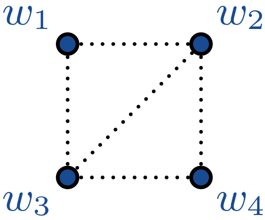
1. determine connected components of G_0,



Shrinking

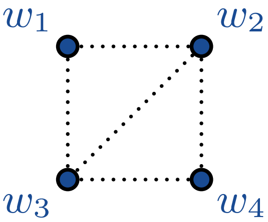
Consider the graph G_0 induced by the 0-edges of the switched vector $\tilde{z}$.

1. determine connected components of G_0,
2. shrink each component to a supernode.



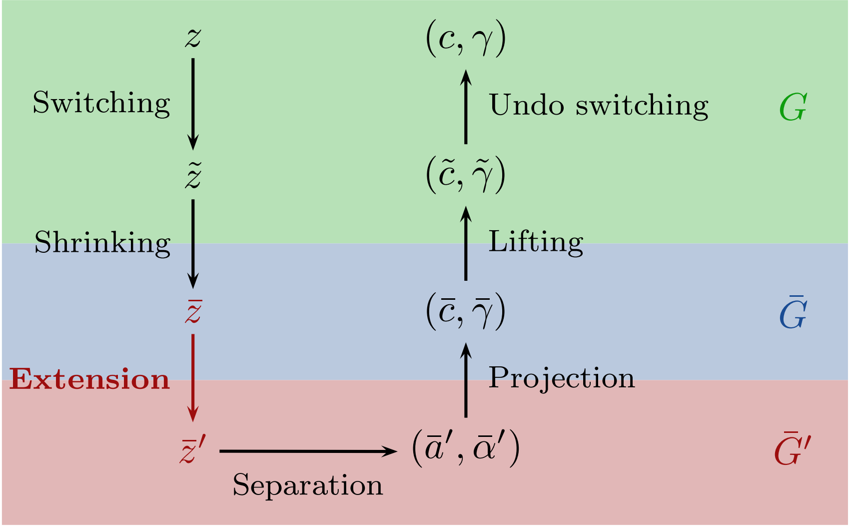
Consider the graph G_0 induced by the 0-edges of the switched vector $\tilde{z}$.

1. determine connected components of G_0,
2. shrink each component to a supernode.

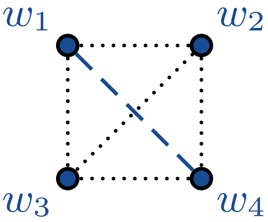


Shrunk vector $\bar{z}$ has only fractional edges. Associated graph $\bar{G}$ may not be complete.

Introduce artificial values for missing edges.



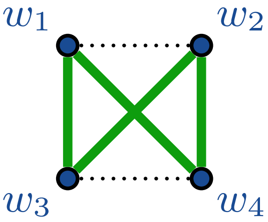
Assign artificial values to missing edges.
Extended vector $\bar{z}'$ shall be in $\mathsf{M}(\bar{G}')$.



Assign artificial values to missing edges.
Extended vector $\bar{z}'$ shall be in $\mathsf{M}(\bar{G}')$.

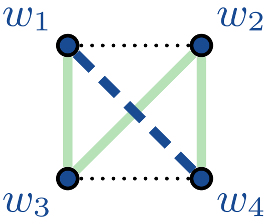
Idea

New cycles in the extended graph consist of



Extension

Assign artificial values to missing edges.
Extended vector $\bar{z}'$ shall be in $\mathsf{M}(\bar{G}')$.



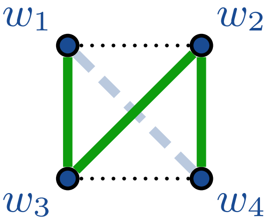
Idea

New cycles in the extended graph consist of an artificial edge

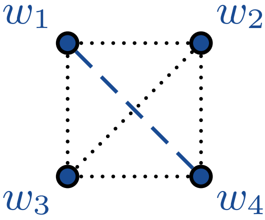
Assign artificial values to missing edges.
Extended vector $\bar{z}'$ shall be in $\mathsf{M}(\bar{G}')$.

Idea

New cycles in the extended graph consist of an artificial edge and a connecting path.



Assign artificial values to missing edges.
Extended vector $\bar{z}'$ shall be in $\mathrm{M}(\bar{G}')$.



Idea

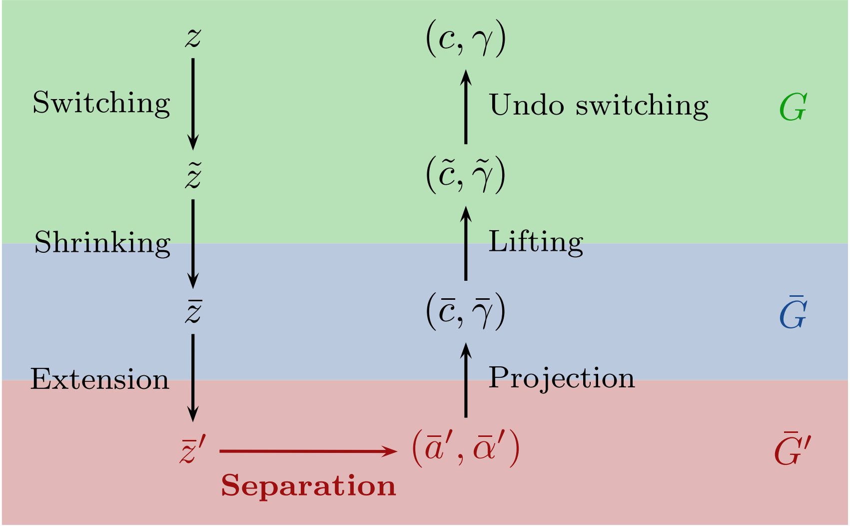
New cycles in the extended graph consist of an artificial edge and a connecting path.

Feasible artificial values

Use shortest-path algorithm to compute range $[\xi_l, \xi_u] \subseteq [0, 1]$ of feasible artificial values for each missing edge.

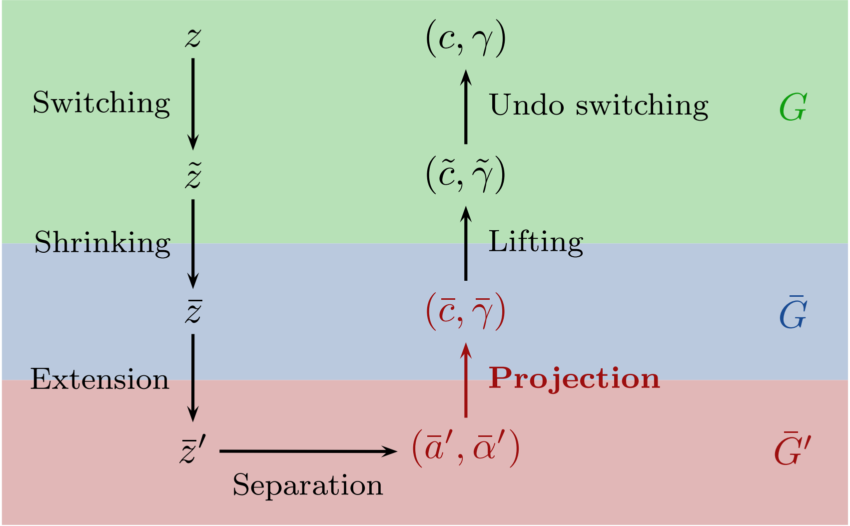
$$\begin{aligned} \xi_l &:= \max \{\bar{z}(F) - \bar{z}(P \setminus F) - |F| + 1 \mid F \subseteq P, |F| \text{ odd}, P \text{ connecting path}\}, \\ \xi_u &:= \min \{-\bar{z}(F) + \bar{z}(P \setminus F) + |F| \mid F \subseteq P, |F| \text{ even}, P \text{ connecting path}\}. \end{aligned}$$

Separate extended vector using techniques for complete graphs.



Outline of the shrinking approach

Project out coefficients related to missing edges.



Projection

Separation $\longrightarrow$ violated inequality $\bar{a}'^T \bar{x}' \leq \bar{\alpha}'$ (denoted $(\bar{a}', \bar{\alpha}')$). Missing edges may have non-zero coefficients!

$$(\cdots \quad \bar{a}'_e \quad \cdots \quad \bar{a}'_f \quad \cdots, \bar{\alpha}')$$

Separation $\longrightarrow$ violated inequality $\bar{a}'^T \bar{x}' \leq \bar{\alpha}'$ (denoted $(\bar{a}', \bar{\alpha}')$). Missing edges may have non-zero coefficients!

$$\begin{array}{rl} & (\cdots \;\; \bar{a}'_e \;\; \cdots \;\; \bar{a}'_f \;\; \cdots,\; \bar{\alpha}') \\ + & (\cdots -\bar{a}'_e \;\; \cdots \;\; \cdots \;\; \cdots,\; \bar{\beta}'_1) \\ + & (\cdots \;\; \cdots \;\; \cdots -\bar{a}'_f \;\; \cdots,\; \bar{\beta}'_2) \end{array}$$

Project out artificial non-zero coefficients

- add multiples of suited valid inequalities,
- odd-cycle inequalities defining the bounds ξ_l, ξ_u are possible candidates.

Separation $\longrightarrow$ violated inequality $\bar{a}'^T \bar{x}' \leq \bar{\alpha}'$ (denoted $(\bar{a}', \bar{\alpha}')$). Missing edges may have non-zero coefficients!

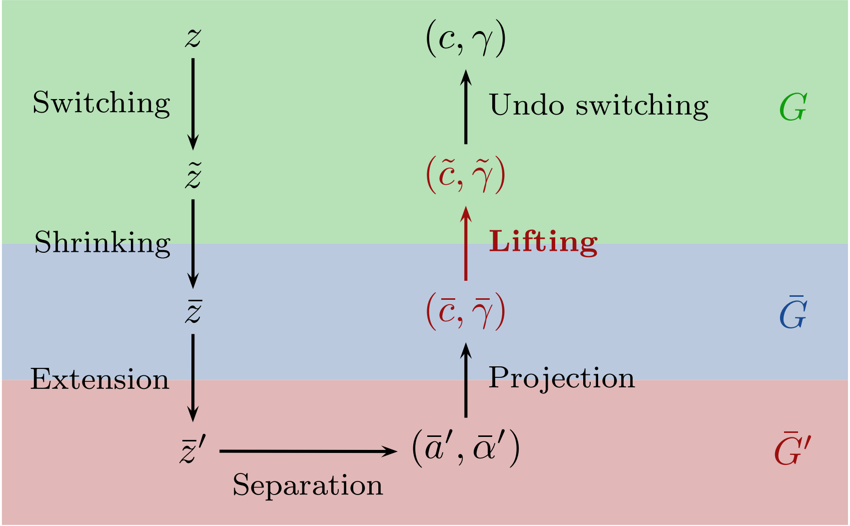
$$
\begin{array}{rl}
 & (\cdots \;\; \bar{a}'_e \;\; \cdots \;\; \bar{a}'_f \;\; \cdots, \; \bar{\alpha}') \\
+ & (\cdots -\bar{a}'_e \;\; \cdots \;\; \cdots \;\; \cdots, \; \bar{\beta}'_1) \\
+ & (\cdots \;\; \cdots \;\; \cdots -\bar{a}'_f \;\; \cdots, \; \bar{\beta}'_2) \\
\hline
= & (\cdots \;\; 0 \;\; \cdots \;\; 0 \;\; \cdots, \; \bar{\gamma})
\end{array}
$$

Project out artificial non-zero coefficients

- add multiples of suited valid inequalities,
- odd-cycle inequalities defining the bounds ξ_l, ξ_u are possible candidates.

In the projected inequality all coefficients of missing edges are zero. Truncation $\longrightarrow$ $(\bar{c}, \bar{\gamma})$.

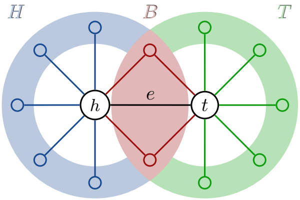
Lift inequality.



Required information

When shrinking edge $e = (h, t)$ store sets:

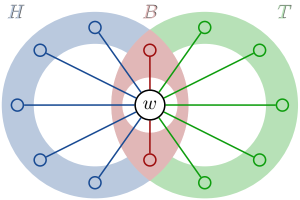
- $H = \{$ exclusive neighbors of $h\}$,
- $T = \{$ exclusive neighbors of $t\}$,
- $B = \{$ common neighbors of h and $t\}$.



Required information

When shrinking edge $e = (h, t)$ store sets:

- $H = \{$ exclusive neighbors of $h \}$,
- $T = \{$ exclusive neighbors of $t \}$,
- $B = \{$ common neighbors of h and $t \}$.



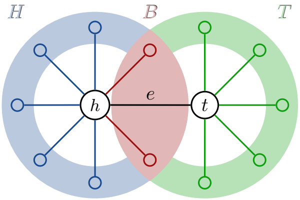
Lift the inequality

- distribute coefficients of edges of the shrunk graph

Required information

When shrinking edge $e = (h, t)$ store sets:

- $H = \{$ exclusive neighbors of $h \}$,
- $T = \{$ exclusive neighbors of $t \}$,
- $B = \{$ common neighbors of h and $t \}$.



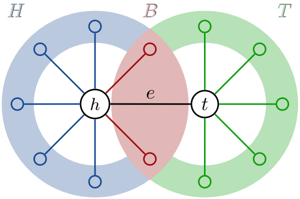
Lift the inequality

- distribute coefficients of edges of the shrunk graph to edges of the original graph w.r.t. above sets,

Required information

When shrinking edge $e = (h, t)$ store sets:

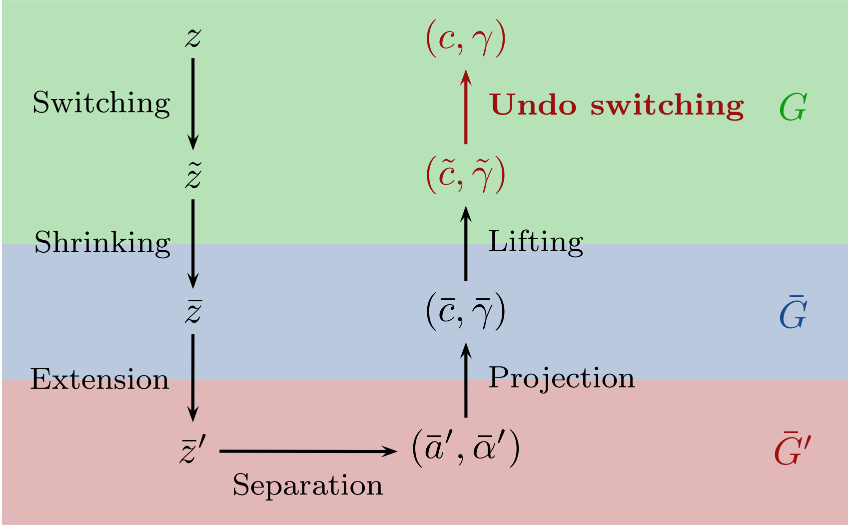
- $H = \{$ exclusive neighbors of $h \}$,
- $T = \{$ exclusive neighbors of $t \}$,
- $B = \{$ common neighbors of h and $t \}$.



Lift the inequality

- distribute coefficients of edges of the shrunk graph to edges of the original graph w.r.t. above sets,
- edge e gets coefficient $-\min\{\sum_{v\in T} |\bar{c}_{wv}|, \sum_{v\in H} |\bar{c}_{wv}|\}$.

Switch back inequality.



Outline

First computational experiments

Implementation in C++ using the B&C framework ABACUS.

Implementation in C++ using the B&C framework ABACUS.

Examined problem classes

2d/3d torus graphs related to spin glass problems. Edge weights:

- ± 1 (probability 0.5 for positive weight),
- Gaussian distributed.

Implementation in C++ using the B&C framework ABACUS.

Examined problem classes

2d/3d torus graphs related to spin glass problems. Edge weights:

- ± 1 (probability 0.5 for positive weight),
- Gaussian distributed.

Benchmark settings

Compute B&C root bound with following separator settings:

1. **no shrink**: SHOC, 4-cycles, exact odd-cycles.
2. **bike**: additional bicycle-p-wheels on shrunk graph.
3. **target**: additional target cuts on shrunk graph.

Tentative results compared to “no shrink” setting

2d torus graphs

Setting	plus-minus			gauss		
	Gap	#LPs	Time	Gap	#LPs	Time
bike	⋆	▼ 30 %	▼ 20 %	⋆	▼ 79 %	▼ 58 %
target	⋆	▼ 30 %	▼ 17 %	⋆	▼ 79 %	▼ 44 %

3d torus graphs

Setting	plus-minus			gauss		
	Gap	#LPs	Time	Gap	#LPs	Time
bike	▼ 0.8 %	▲ 39 %	▲ 8233 %	▼ 6.3 %	▲ 13 %	▲ 470 %
target	▼ 0.1 %	▼ 8 %	▲ 296 %	▼ 4.4 %	▲ 5 %	▲ 90 %

Conclusion

Separation method for max-cut problems based on graph shrinking:

- enables transfer of separation techniques from dense/complete graphs to sparse graphs,
- shows potential to improve solvability of max-cut problems at least for certain problem classes.

Conclusion

Separation method for max-cut problems based on graph shrinking:

- enables transfer of separation techniques from dense/complete graphs to sparse graphs,
- shows potential to improve solvability of max-cut problems at least for certain problem classes.

Future work

- identify and eliminate bottlenecks,
- test different perturbations on ± 1-torus graphs,
- test different shrinking orders (e.g. randomization),
- develop alternative to usage of cycle polytope M(G).

Acknowledgment

- **Prof. G. Reinelt** (University of Heidelberg)
- **Dr. M. Oswald** (University of Heidelberg)
- **Prof. G. Rinaldi** (IASI Rome)
- **Prof. M. Jünger** (University of Cologne)

Thank you!

Thank you for your attention!