Maximum Cut Algorithms

Professor Dr. Petra Mutzel
slides partially by Christine Dahn

Lehrstuhl für Algorithm Engineering

Fakultät für Informatik

TU Dortmund

Selected Topics on Combinatorial Optimization

Vienna Graduate School on Computational Optimization

Lecture 7, November 13, 2018

Outline

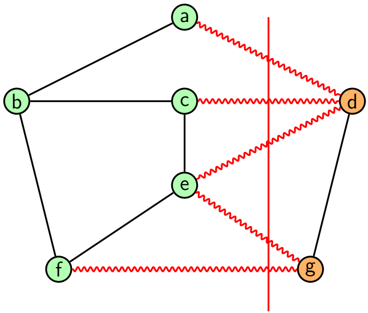
Literature

- Dahn, C., N. M. Kriege, and P. Mutzel: A fixed-parameter algorithm for the max-cut problem on embedded 1-planar graphs, IWOCA 2018, LNCS 10979, Springer, 2018, 141–152
- F. Hadlock: Finding a maximum cut of a planar graph in polynomial time, SIAM Journal on Computing, 4(3), 1975, 221–225
- F. Liers and G. Pardella: Computational Optimization and Applications 51:(1), 2012, 323–344
- M. X. Goemans and D. P. Williamson: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, Journal of the ACM (42) (6), 1995, 1115–1145

Definition (Cut in a Graph)

Let $G = (V, E, c)$ be an undirected graph with edge weights $c(e)$ for all edges $e \in E$.

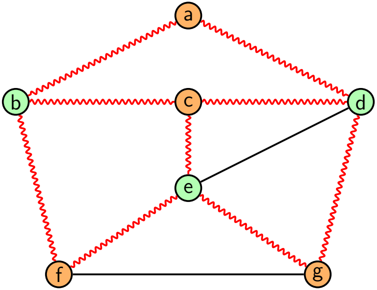
- $S \subseteq V$ defines a cut $F \subseteq E$ in $G = (V, E, c)$ with $F = (S \times \overline{S}) \cap E$
- value of F: *sum over all edge weights* in F



value: 5

Definition (Max-Cut Problem)

Given: An undirected, weighted graph $G = (V, E, c)$.
Find: A cut $F \subseteq E$ in G with maximum value.



value: 9

Complexity of the Max-Cut Problem

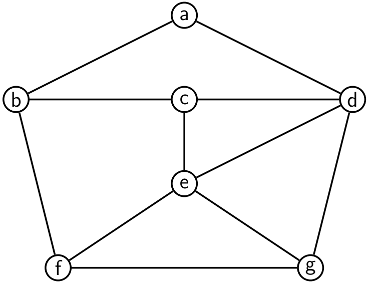
- undirected graphs: NP-hard
- planar graphs: polynomial time solvable [Hadlock 1875]
- embedded k-almost planar graphs: polynomial time solvable [Dahn, Kriege, Mutzel 2018]
- embedded 1-planar graphs: Fixed-Parameter-Tractable [Dahn, Kriege, Mutzel 2018]

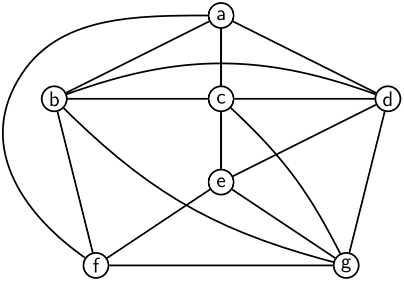
Definition (planar graph)

- *no edges crossing*

Definition (1-planar graph)

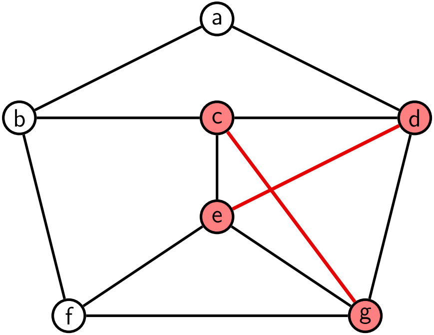
- *at most 1 crossing per edge*





Definition (k-almost planar graph)

- 1-planar
- *with at most k crossings*



a 1-almost planar graph

note: $k \in \mathcal{O}(n)$

Approximation Results

Approximation Results

- Max-Cut has a 0.87856-Approximation (Goemans and Williamson 1994 and Mahajan and Ramesh 1995)
- Max-Cut has no 0.94-Approximation under the assumption that $\mathrm{P} \neq \mathrm{NP}$ (Håstad 1996)

For the unweighted case (all edge weights 1) there exist simple $\frac{1}{2}$-approximation algorithms:

Simple $\frac{1}{2}$-Approximation Algorithms for Unweighted Case

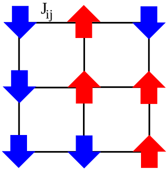
- simple randomized algorithm with expected approximation factor $\frac{1}{2}$
- simple local search algorithm with approximation factor $\frac{1}{2}$

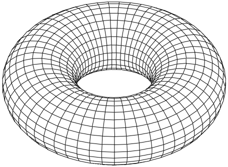
Motivation

Applications

- Quadratic 0/1-Optimization $\min\{x^T Qx + q^T x \mid x \in \{0,1\}^n\}$ for a matrix Q and a vector q
- Via minimization for designing electronic circuits
- Graph partitioning helps for Divide-and-Conquer approaches for big data into 2 or k parts of similar size, e.g. distributed computing or if the graph is too large to be handeled
- Break minimization with the home/away allocation for sport leagues
- Energy minimization for Ising Spin Glasses in Physics
- Adiabatic Quantum Computing

Ising Spin Glasses

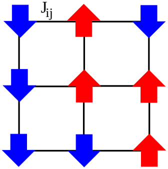


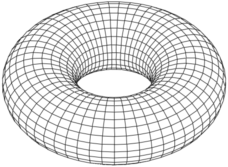


Ising Spin Glasses

- Ising Spin Glasses are certain alloys, e.g., AuFe or $Rb_2Cu_{1-x}Co_xF_4$.
- We can think of a pure material polluted by some iron atoms.
- Slowly reduce the temperature of the alloy to absolute zero.
- We are interested in the magnetic moments of the spins (iron atoms).
- This state of minimal energy is called the ground state.

Ising Spin Model Problem





Definition (Ising Spin Model Problem (ISM))

- **Given:** $G = (V, E)$ with weights h_i $(i \in V)$ and J_{ij} $((i,j) \in E)$
- **Find:** Values for the spin variables $s_i \in \{-1, +1\}$, $i \in V$, so that

$$E(s) = \sum_{i \in V} h_i s_i + \sum_{(i,j) \in E} J_{ij} s_i s_j$$

is minimized. $\leftarrow$ ground state

(ISM) equivalent to QUBO, MAX W2SAT, MAX CUT

Transformation to Max-Cut

We identify a spin configuration with a partition $V = V^+ \cup V^-$ ($V^+ \cap V^- = \emptyset$), so that $V^+ = \{i \in V \mid s_i = +1\}$ and $V^- = \{i \in V \mid s_i = -1\}$.

$$
\begin{aligned}
H(S) &= \sum_{\{i,j\} \in E} J_{ij} s_i s_j \\
&= \sum_{\{i,j\} \in E,\ i,j \in V^+} J_{ij} + \sum_{\{i,j\} \in E,\ i,j \in V^-} J_{ij} - \sum_{\{i,j\} \in E,\ i \in V^+, j \in V^-} J_{ij}
\end{aligned}
$$

and thus

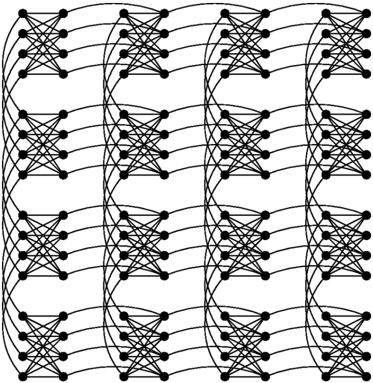
$$
H(S) - \sum_{\{i,j\} \in E} J_{ij} = -2 \sum_{\{i,j\} \in E,\ i \in V^+,\ j \in V^-} J_{ij}.
$$

Setting $c_{ij} = -J_{ij}$ for all adjacent pairs of spins leads to Max-Cut.

Adiabatic Quantum Computing

Structure of Adiabatic Quantum Computer by D-Wave:

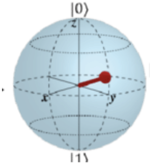
4x4 Chimera grid

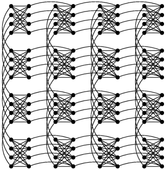


D-Wave 2017: 2000Q with 2048 Qubits (16x16 Chimera grid)

mit Jnger (Köln), Reinelt (Heidelberg), Rinaldi (Rom); Stollenwerk, Lobe (DLR), Kaibel (Magdeburg); McGeoch (D-Wave)

Adiabatic Quantum Computing





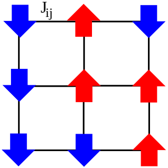
Idee: Adiabatic Quantum Computing

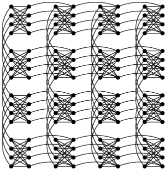
- construct system (A) with (unknown) ground state, which corresponds to the solution of my problem
- construct system (B) whose ground state can be prepared and measured experimentally
- transfer system (B) slowly to system (A) and measure the ground state ← Adiabatic Theorem

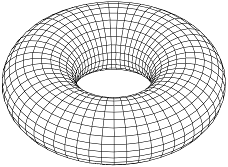
D-Wave: 2017: 2000Q with 2048 Qubits (16x16 Chimera grid)

Source:
https://www.researchgate.net/publication321133310_Molecular_Spin_Qudits_for_Quantum_Algorithms/figures?lo=1

Quantum Annealing







Definition (Ising Spin Model Problem (ISM))

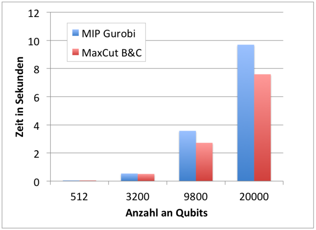
- **Given:** $G = (V, E)$ with weights h_i ($i \in V$) and J_{ij} ($(i, j) \in E$)
- **Find:** Values for the spin variables $s_i \in \{-1, +1\}$, $i \in V$, so that

$$E(s) = \sum_{i \in V} h_i s_i + \sum_{(i,j) \in E} J_{ij} s_i s_j$$

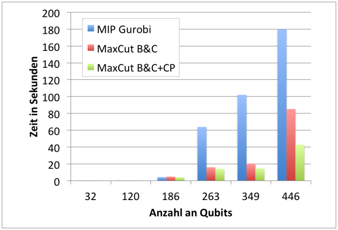
is minimized. ← ground state

(ISM) equivalent to QUBO, MAX W2SAT, MAX CUT

Experimental Evaluation (Exact Solution)



random QUBO instances



random ISM instances

- D-Wave Quantum Annealer (C_8 with 500 Qubits) vs. QP CPLEX [McGeoch,Wang 2013]
- Experiments with MIP-solver CPLEX [Dash/Puget 2015]
- Our MaxCut Branch-and-Cut vs. strengthened MaxCut B&C by cutting planes ← provable exact solutions

Simple Randomized Max-Cut Algorithm

Algorithm Randomized Max-Cut

1. $V_1 := V_2 := \emptyset$
2. **for** all $v \in V$
3. with probability $\frac{1}{2}$ set $V_1 = V_1 \cup \{v\}$
   otherwise set $V_2 = V_2 \cup \{v\}$
4. **Return** V_1, V_2

Theorem

Algorithm Randomized Max-Cut computes a cut in graph $G = (V, E)$ containing an expected number of $|E|/2$ edges. Hence, for an unweighted graph it provides an expected approximation factor of $\frac{1}{2}$. The algorithm runs in linear time $O|V|$.

Proof

We show $\mathrm{E}\,(\#\text{cut edges}) = \mathrm{E}\,(w(V_1, V_2)) = |E|/2$

Define indicator variables X_e for $e \in E$

$$\text{with } X_e := \begin{cases} 1 & \text{if } |e \cap V_1| = |e \cap V_2| = 1 \\ 0 & \text{else} \end{cases}$$

Observation $w(V_1, V_2) = \sum_{e \in E} X_e$

$$\begin{aligned} \mathrm{E}\,(w(V_1, V_2)) &= \mathrm{E}\left(\sum_{e \in E} X_e\right) \\ &= \sum_{e \in E} \mathrm{E}\,(X_e) \\ &= \sum_{e \in E} \mathrm{Prob}\,(X_e = 1) \\ &= |E| \cdot \mathrm{Prob}\,(X_e = 1)\,, \text{ for an edge } e = (u, v) \end{aligned}$$

Expected Number of Edges in the Cut

We have

$$
\begin{aligned}
& \mathrm{E}\left(w(V_1, V_2)\right) \\
= \;& |E| \cdot \mathrm{Prob}\left(X_e = 1\right), \text{ for an edge } e = (u, v) \\
= \;& |E| \cdot \mathrm{Prob}\left(((u \in V_1) \wedge (v \in V_2)) \vee ((u \in V_2) \wedge (v \in V_1))\right) \\
= \;& |E| \cdot \left(\mathrm{Prob}\left((u \in V_1) \wedge (v \in V_2)\right) + \mathrm{Prob}\left((u \in V_2) \wedge (v \in V_1)\right)\right) \\
= \;& |E| \cdot \left(\mathrm{Prob}\left(u \in V_1\right) \cdot \mathrm{Prob}\left(v \in V_2\right) + \mathrm{Prob}\left(u \in V_2\right) \cdot \mathrm{Prob}\left(v \in V_1\right)\right) \\
= \;& |E| \cdot \left(\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2}\right) = |E| \cdot \left(\frac{1}{4} + \frac{1}{4}\right) = \frac{|E|}{2}
\end{aligned}
$$

Since $\mathrm{OPT} \leq |E|$, we get an expected approximation factor of $\frac{1}{2}$. $\square$

Local Search for Unweighted Max-Cut

General idea of local search

1. Find a feasible initial solution.
2. Repeat until convergence: Find a solution in the neighbourhood with better objective value.

It is necessary to define problem dependent neighbourhoods.

This strategy leads to a local optimum with respect that there is no better solution in the neighbourhood.

Local search for unweighted Max-Cut

Initial solution e.g. $V_1 = \emptyset$, $V_2 = V$

Definition (Neighbourhood $\mathcal{N}$ of a cut)

The 1-Exchange neighbourhood w.r.t. a cut (V_1, V_2) consists of all partitions (V_{1k}, V_{2k}) for $k = 1, 2, \ldots, |V|$ with

1. If $v_k \in V_1$, then $V_{1k} = V_1 \setminus \{v_k\}$ und $V_{2k} = V_2 \cup \{v_k\}$,
2. If $v_k \in V_2$, then $V_{1k} = V_1 \cup \{v_k\}$ und $V_{2k} = V_2 \setminus \{v_k\}$.

Theorem (Local Search for Max-Cut)

For an unweighted Max-Cut instance G with optimum value OPT let $m_{\mathcal{N}}(G)$ the solution value of a local optimum w.r.t $\mathcal{N}$. Then we have

$$m_{\mathcal{N}}(G) \geq \frac{\mathrm{OPT}}{2}.$$

Proof: Analysis of the 1-Exchange for Max-Cut

Let $m := |E|$ and V_1, V_2 the corresponding vertex sets to $m_{\mathcal{N}}$.

Since $\mathrm{OPT} \leq m$ it suffices to show $2m_{\mathcal{N}}(G) \geq m$.

For $W \subseteq V$ let $E(W) = \{(u, v) \in E \mid u, v \in W\}$ and $\delta(W) = \{(u, v) \in E \mid u \in W \text{ and } v \notin W\}$, $m_1 = |E(V_1)|$, and $m_2 = |E(V_2)|$. Then $m = m_1 + m_2 + m_{\mathcal{N}}(G)$.

We can show that $m_1 + m_2 - m_{\mathcal{N}}(G) \leq 0$

Together this leads to

$$\implies \quad m - 2m_{\mathcal{N}}(G) \leq 0$$

$$\implies \quad 2m_{\mathcal{N}}(G) \geq m$$

Proof of $m_1 + m_2 - m_{\mathcal{N}}(G) \leq 0$

(V_1, V_2) is local optimum with value $m_{\mathcal{N}}(G)$ and V_{1k}, V_{2k} let be the neighbourhood partitions. For $v_i \in V$ let

$$\begin{aligned} m_{1i} &= |\{v \mid v \in V_1, (v, v_i) \in E\}|, \\ m_{2i} &= |\{v \mid v \in V_2, (v, v_i) \in E\}|. \end{aligned}$$

$$\Longrightarrow \quad (\forall k) \begin{cases} |\delta(V_{1k})| \leq m_{\mathcal{N}}(G) \\ |\delta(V_{2k})| \leq m_{\mathcal{N}}(G) \end{cases}$$

$$\Longrightarrow \quad \left\{ \begin{array}{l} (\forall v_i \in V_1)\ m_{1i} - m_{2i} \leq 0 \\ (\forall v_j \in V_2)\ m_{2j} - m_{1j} \leq 0 \end{array} \right\} \geq \text{ that many neighbours on the other side}$$

$$\Longrightarrow \quad \begin{cases} \sum_{v_i \in V_1} (m_{1i} - m_{2i}) = 2m_1 - m_{\mathcal{N}}(G) \leq 0 \\ \sum_{v_j \in V_2} (m_{2j} - m_{1j}) = 2m_2 - m_{\mathcal{N}}(G) \leq 0 \end{cases}$$

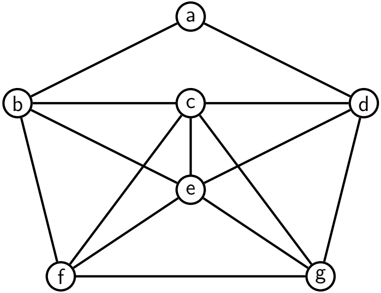
$$\Longrightarrow \quad m_1 + m_2 - m_{\mathcal{N}}(G) \leq 0 \qquad \square$$

FPT-Algorithm for Max-Cut

Input of the Algorithm: Graph $G = (V, E, c)$ with non-negative edge weights and its set of crossing edges in a 1-planar embedding with k crossings (can be extended to arbitrary edge weights)

Definition (Set of crossing edges)

- embedding of G provides a set of crossing edges X
- crossing $\chi \in X$



$\chi_1 = \{cg, de\}$,
$\chi_2 = \{be, cf\}$,
$X = \{\chi_1, \chi_2\}$

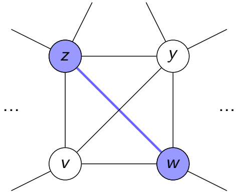
note: $|X| = k$

Max-Cut for 1-planar graphs: Idea

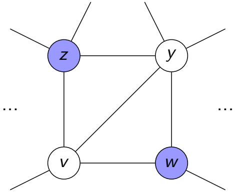
- transform the given 1-planar graph into a planar graph
- use a `MaxCut` algorithm for planar graphs
- transfer the result back onto the original graph

Planarize a crossing

Option 1: Deleting an edge



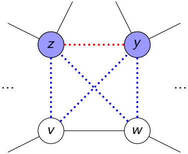
G



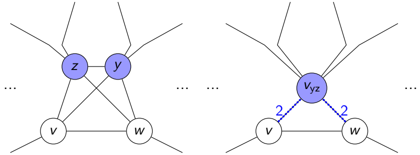
$G - wz$

Planarize a crossing

Option 2: Contracting two nodes



G



G/yz

Algorithm

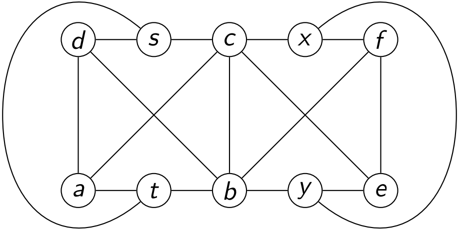
$\mathtt{GewMaxCut}_{\mathtt{k}}(G, \Gamma, X(\Gamma))$

Input: Graph G with $\mathtt{k}$-almost planar drawing Γ, crossing set $X(\Gamma)$ and $\mathtt{k} = |X(\Gamma)|$
Output: Maximal Cut $F \subseteq E_G$

if $\mathtt{k} == 0$ **then**
  $F = \textsc{GewMaxCut}_{\mathtt{pl}}(G, \Gamma)$
else
  choose an element $z = \{e_1 = ac, e_2 = bd\} \in X(\Gamma)$
  $G_1 = G/cd$;
  $G_2 = G/bc$;
  $G_3 = G - e_2$;
  $F_i = \textsc{MaxCut}_{|X_i|}(G_i, \Gamma_i, X_i)$ for $i \in \{1, 2, 3\}$
  $j = \operatorname{argmax}_{1 \leq i \leq 3} c(F_i)$
  $F =$ transfer optimal cut F_j to G
end if
return F

Example: Calculating $\texttt{MaxCut}(H, X)$

Given the 2-almost planar graph H with the set of crossings $X = \{\chi_1, \chi_2\}$:

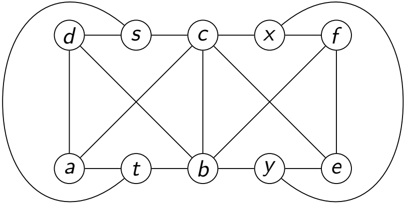


H

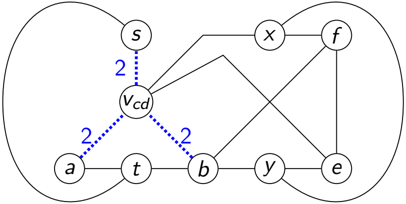
Choose $\chi_1 = \{ac, bd\}$ to be planarized.

Example: Calculating $\texttt{MaxCut}(H, X)$

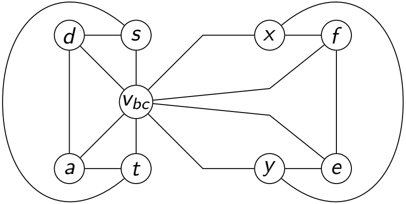
Calculate the three new subgraphs of H w.r.t. χ_1:



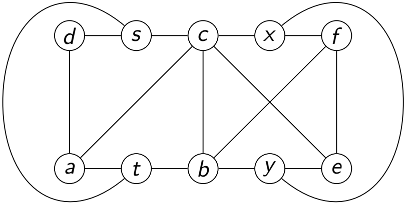
H



$H_1 = H/cd$



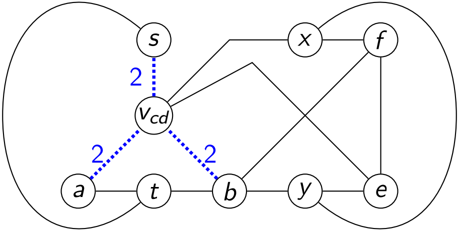
$H_2 = H/bc$



$H_3 = H - bd$

Example: Calculating $\texttt{MaxCut}(H_1, X_1)$

Recursive call for H_1 with $X_1 = \{\chi_2\}$:

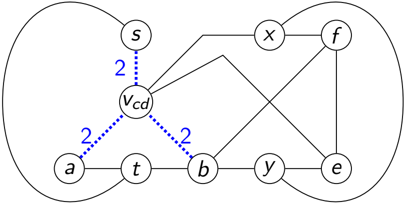


$H_1 = H/cd$

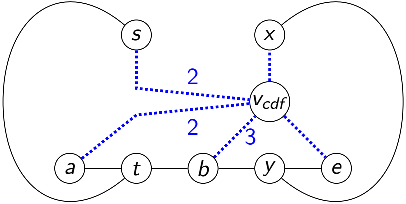
Choose $\chi_2 = \{bf, v_{cd}e\}$ to be planarized

Example: Calculating $\texttt{MaxCut}(H_1, X_1)$

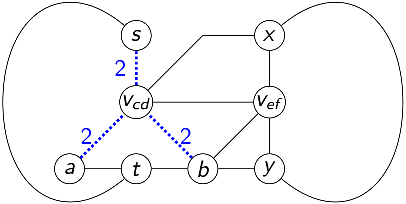
Calculate the three new subgraphs of H_1 w.r.t. χ_2:



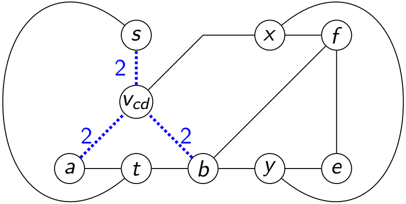
$H_1 = H/cd$



H_1/cf



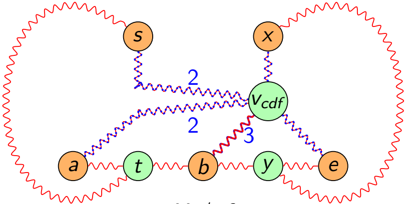
H_1/ef



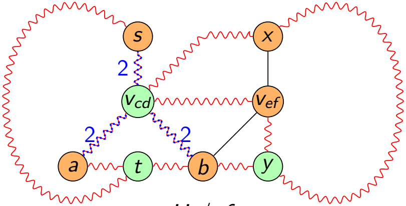
$H_1 - ce$

Example: Calculating $\texttt{MaxCut}(H_1, X_1)$

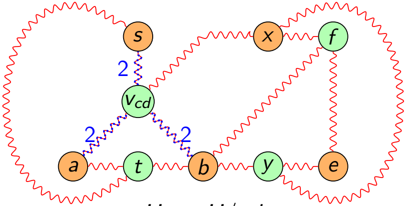
Recursively calculated solutions:



H_1/cf



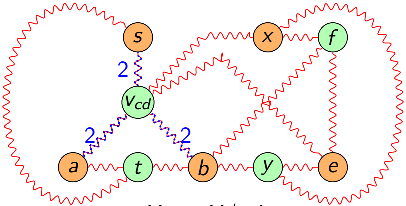
H_1/ef



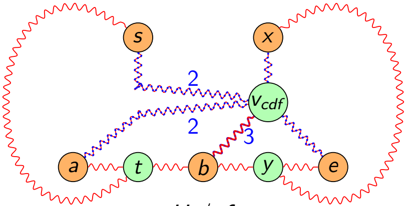
$H_1 = H/cd$

Example: Calculating $\mathtt{MaxCut}(H_1, X_1)$

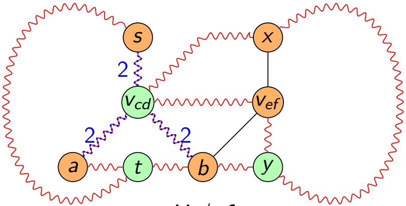
Finding the best solution and transferring it onto H_1:



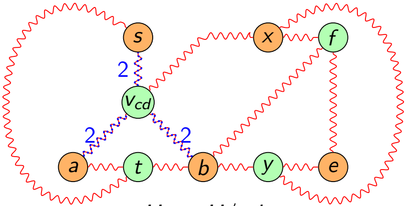
$H_1 = H/cd$



H_1/cf



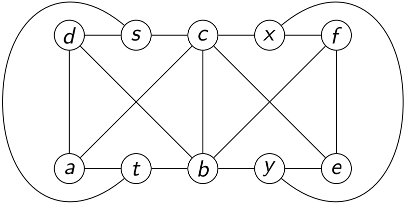
H_1/ef



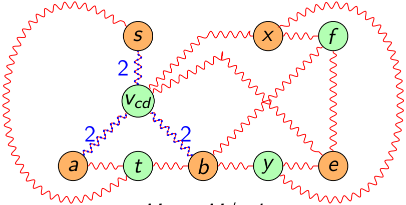
$H_1 = H/cd$

Example: Calculating $\texttt{MaxCut}(H, X)$

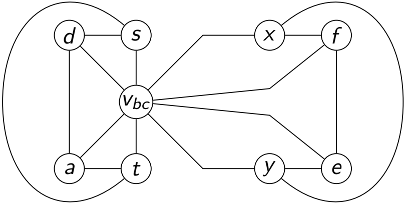
Recursively calculated solution for H_1:



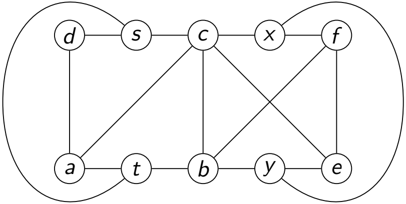
H



$H_1 = H/cd$



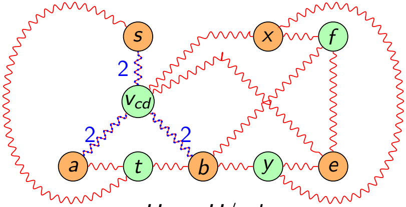
$H_2 = H/bc$



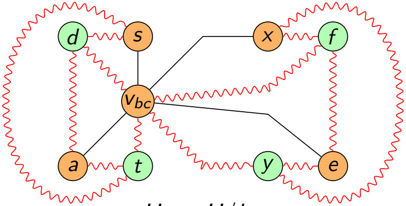
$H_3 = H - bd$

Example: Calculating $\texttt{MaxCut}(H, X)$

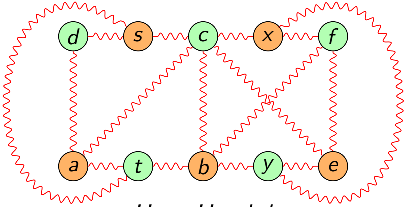
Recursively calculated solutions:



$H_1 = H/cd$



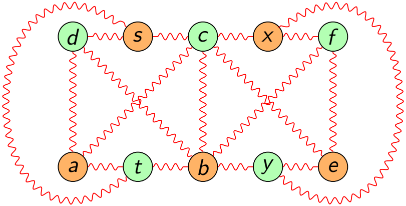
$H_2 = H/bc$



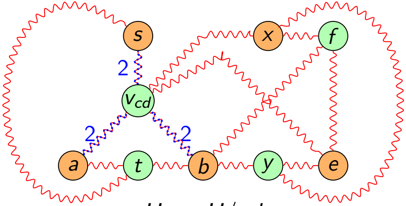
$H_3 = H - bd$

Example: Calculating $\texttt{MaxCut}(H, X)$

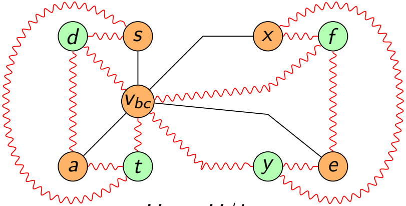
Finding the best solution and transferring it onto H:



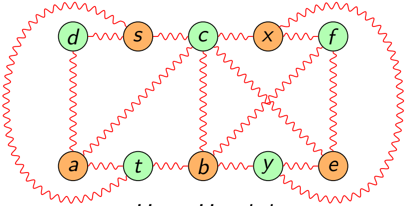
H



$H_1 = H/cd$



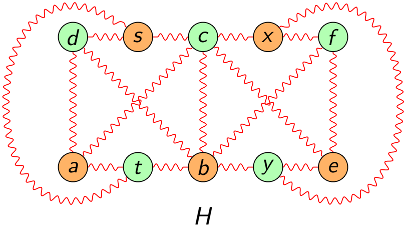
$H_2 = H/bc$



$H_3 = H - bd$

Example: Calculating $\texttt{MaxCut}(H, X)$

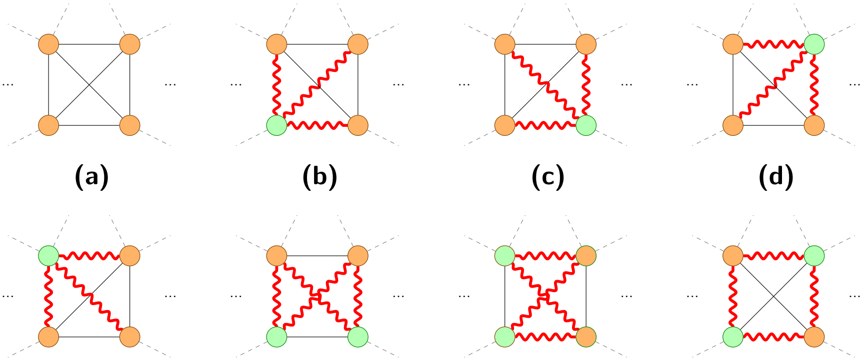
Maximum cut in H:



Value of the maximum cut in H: 17

Partitioning the endpoints of a crossing

8 non-isomorphic partitions of the four endpoints of a crossing:



(a) (b) (c) (d)

(e) (f) (g) (h)

Proof of optimality *(sketch)*

Theorem 1

`MaxCut` computes a maximum cut in a 1-planar graph G given a set of crossing edges X in a 1-planar embedding of G.

Proof by induction over $k = |X|$

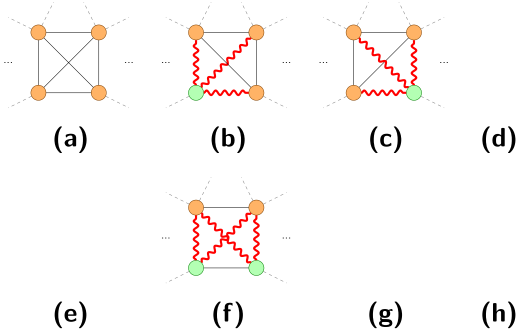
$k = 0$

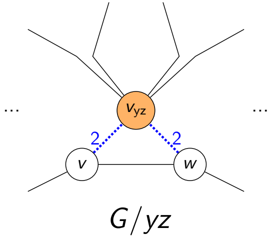
- G is planar

$k > 0$

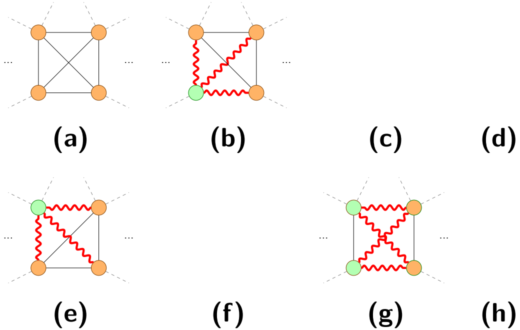
- look at 8 cases (a)-(h)

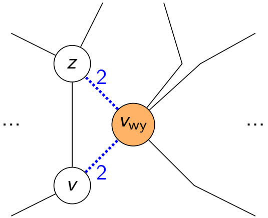
Proof of optimality *(sketch)*





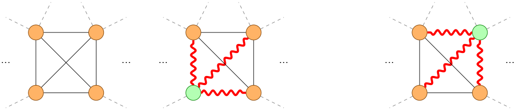
Proof of optimality *(sketch)*



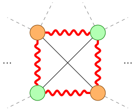


G/wy

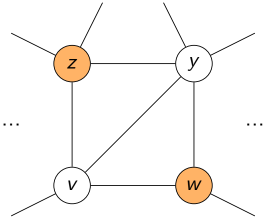
Proof of optimality *(sketch)*



(a) **(b)** **(c)** **(d)**



(e) **(f)** **(g)** **(h)**



$G - wz$

Running time

Theorem

`MaxCut` computes a maximum cut in an embedded 1-planar graph with n nodes and k crossings in time $\mathcal{O}(3^k \cdot n^{3/2} \log n)$.

Sketch of proof:

- no more than 3^k recursive calls
- dominating running time of one call: $\mathcal{O}(n^{3/2} \log n)$
- overall running time: $\mathcal{O}(3^k \cdot n^{3/2} \log n)$

FPT

Theorem

`MaxCut` computes a maximum cut in an embedded 1-planar graph with n nodes and k crossings in time $\mathcal{O}(3^k \cdot n^{3/2} \log n)$.

Corollary

`MaxCut` computes a maximum cut in an embedded k-almost planar graph in polynomial time.

Parameterization

The MAX-CUT problem on embedded 1-planar graphs with k crossings is *fixed-parameter-tractable* with parameter k.

Conclusion and open problems

Conclusion

- MAX-CUT on embedded k-almost planar graphs $\in P$
- MAX-CUT on embedded 1-planar graphs $\in FPT$

Open problems

- Drop the assumption that we are given a 1-planar embedding
- Generalize this approach to k-planar or non restricted graphs

Counter example on a 4-planar graph