Algorithmic Graph Theory
Part IV - Further Examples of Tractable Problems

Martin Milanič
martin.milanic@upr.si

University of Primorska, Koper, Slovenia

Dipartimento di Informatica
Università degli Studi di Verona, March 2013

What we'll do

1. 2-SATISFIABILITY.
2. 3-coloring Graphs with Small Dominating Sets.
3. Maximum Independent Set Problem: an Overview of Combinatorial Techniques.
4. Max Cut Problem in Planar Graphs.

A POLYNOMIAL ALGORITHM FOR 2-SATISFIABILITY.

Recall the SATISFIABILITY problem:

SATISFIABILITY	
Input:	Boolean variables $x_1, \ldots, x_n$,
	clauses $C_1, \ldots, C_m$ over $x_1, \ldots, x_n$
	[clause = a disjunction of *literals*
	(variables or their negations)]
Question:	Is there a satisfying truth assignment?

Example:
Suppose we are given the following input to SATISFIABILITY:
$\{x_1, x_2, x_3\}$,
$C_1 = x_1 \vee x_2 \vee x_3$,
$C_2 = x_1 \vee \overline{x_2} \vee \overline{x_3}$,
$C_3 = \overline{x_2} \vee x_3$,
$C_4 = \overline{x_1} \vee \overline{x_3}$.
$(\overline{x}_i \equiv \neg x_i)$

Truth assignment $x_1 = \top$, $x_2 = x_3 = \bot$ makes all clauses satisfied.

2-SATISFIABILITY: just like SATISFIABILITY, except that every clause consists of exactly 2 literals, $C_i = \lambda_{i1} \vee \lambda_{i2}$

This problem can be solved by a polynomial algorithm based on the **implication digraph** $D = (V, A)$:

$$V = \{x_1, \ldots, x_n\} \cup \{\overline{x_1}, \ldots, \overline{x_n}\},$$

$$A = \bigcup_{i=1}^{m} \left\{ (\overline{\lambda_{i1}}, \lambda_{i2}), (\overline{\lambda_{i2}}, \lambda_{i1}) \right\}.$$

Directed edge $(\overline{\lambda_{i1}}, \lambda_{i2})$ means: if $\overline{\lambda_{i1}} = \top$, then also $\lambda_{i2} = \top$.

A Polynomial Algorithm for 2-SATISFIABILITY

Algorithm 2-SAT:
Input: Boolean variables $x_1, \ldots, x_n$, clauses $C_1, \ldots, C_m$ of length 2
Output: assignment of values $\{\top, \bot\}$ for $x_1, \ldots, x_n$ for which the proposition $C_1 \wedge C_2 \wedge \ldots \wedge C_m$ is true, if such an assignment exists, NO, otherwise

Construct the implication digraph $D = (V, A)$.
for each $v \in V$
    Let $S(v)$ be the set of points reachable from v. (Use BFS.)
end for
if $\exists$ a variable x_i such that $\overline{x_i} \in S(x_i)$ and $x_i \in S(\overline{x_i})$ **then**
    return NO;
end if
while $\exists$ a variable x_i such that the value of x_i is still undetermined **do**
    if $\overline{x_i} \notin S(x_i)$
        set the value $\top$ to all elements of $S(x_i)$;
        set the value $\bot$ to the negations of all elements of $S(x_i)$;
    else
        set the value $\top$ to all elements of $S(\overline{x_i})$;
        set the value $\bot$ to the negations of all elements of $S(\overline{x_i})$;
    end if
end while

return the computed assignment;

Example:

variables: $x_1, \ldots, x_6$

clauses:

$C_1 = \overline{x_1} \lor x_2$

$C_2 = \overline{x_1} \lor \overline{x_4}$

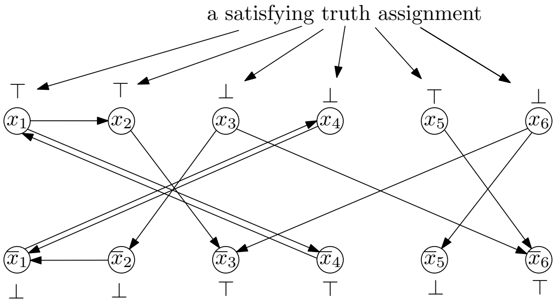
$C_3 = x_1 \lor x_4$

$C_4 = \overline{x_2} \lor \overline{x_3}$

$C_5 = \overline{x_3} \lor \overline{x_6}$

$C_6 = \overline{x_5} \lor \overline{x_6}$

implication digraph



$S(x_1) = \{x_1, x_2, \overline{x}_3, \overline{x}_4\}$

$S(x_2) = \{x_2, \overline{x}_3\}$

$S(x_3) = \{x_3, x_4, \overline{x}_1, \overline{x}_2, \overline{x}_6\}$

$S(x_4) = \{x_4, \overline{x}_1\}$

$S(x_5) = \{x_5, \overline{x}_6\}$

$S(x_6) = \{x_5, x_6, \overline{x}_3, \overline{x}_5, \overline{x}_6\}$

$S(\overline{x}_1) = \{x_4, \overline{x}_1\}$

$S(\overline{x}_2) = \{x_4, \overline{x}_1, \overline{x}_2\}$

$S(\overline{x}_3) = \{\overline{x}_3\}$

$S(\overline{x}_4) = \{x_1, x_2, \overline{x}_3, \overline{x}_4\}$

$S(\overline{x}_5) = \{x_5, \overline{x}_5, \overline{x}_6\}$

$S(\overline{x}_6) = \{\overline{x}_6\}$

A Polynomial Algorithm for 2-SATISFIABILITY

Time complexity of the algorithm is polynomial ($O(n(n+m))$):

- $2n$ BFS's on a digraph with $2n$ vertices and $2m$ edges;
- at most n iterations of the **while** loop; each uses $O(n)$ time.

Correctness of the algorithm follows from the following facts:

- If there exists a variable x_i such that $\overline{x_i} \in S(x_i)$ and $x_i \in S(\overline{x_i})$, then the proposition is not satisfiable since $x_i \Rightarrow \overline{x_i}$ and $\overline{x_i} \Rightarrow x_i$.
- Claim: *If $u \in S(v)$, then also $\overline{v} \in S(\neg u)$.*

  Proof: Directly from the construction of the implication digraph, since $(u,v) \in E \Rightarrow (\overline{v},\overline{u}) \in E$.
- When setting the values in the **while** loop, no conflict arises:

  Suppose that when traversing $S(x_i)$ we set the value $\top$ both to a variable x_j and its negation $\overline{x_j}$. This means that $x_j \in S(x_i)$ and $\overline{x_j} \in S(x_i) \Rightarrow \overline{x_i} \in S(x_j)$. Therefore in D there is an x_i–x_j path and an x_j–$\overline{x_i}$ path, therefore there also exists an x_i–$\overline{x_i}$ path. This however is a contradiction with the assumption $\overline{x_i} \notin S(x_i)$.

  A similar reasoning can be used for the traversal of $S(\overline{x_i})$.

A Linear Algorithm for 2-SATISFIABILITY

There exist linear time implementations.

Aspvall, Plass and Tarjan (1979) gave a simple linear time algorithm:

- compute the strongly connected components (SCCs) of the implication digraph,
- if a variable and its negation are in the same SCC, then there is no satisfying assignment,
- else, shrink the SCCs to obtain the *condensation digraph* (which is an acyclic digraph),
- topologically order the condensation digraph, traverse the SCCs in this order, setting all unset terms in a SCC to be false, and all terms in the complementary component to true.

3-coloring Graphs with Small Dominating Sets.

3-coloring Graphs with Small Dominating Sets

3-COLORABILITY is NP-complete.

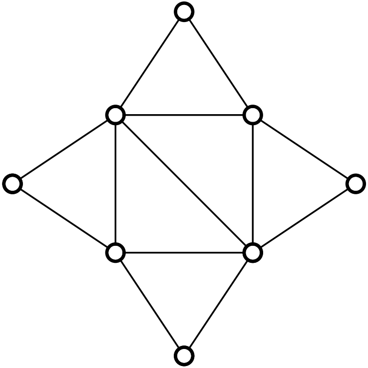
Theorem

For every k, 3-COLORABILITY is solvable in polynomial time for graphs that contain a dominating set of size at most k.

dominating set in a graph $G = (V, E)$:
a set $D \subseteq V$ such that every vertex not in the set has a neighbor in the set:

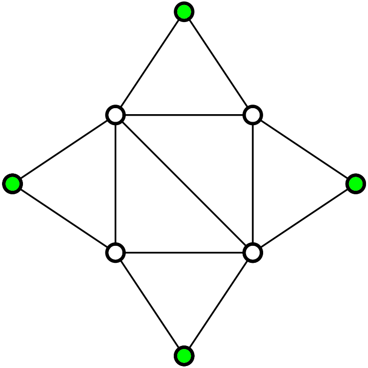
$$(\forall u \in V \setminus D)(\exists v \in D)(uv \in E)\,.$$

Example:

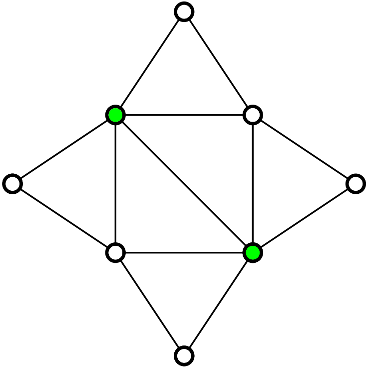


3-coloring Graphs with Small Dominating Sets

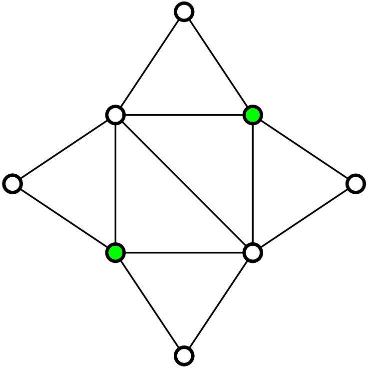
Example:



Example:



Example:



3-coloring Graphs with Small Dominating Sets

Suppose that G has a dominating set of size at most k.

Algorithm for 3-coloring:

1. Find a dominating set D of size k in G. $O(n^k)$
2. Generate the set $\mathcal{C}_3$ all possible 3-colorings of the subgraph of G induced by D. $O(3^k)$
3. **for each** $c \in \mathcal{C}_3$ **do** $O(3^k f(n))$
   if c can be extended to a 3-coloring of G **then** $O(f(n))$
   return YES; (G is 3-colorable)
4. **return** NO; (G is not 3-colorable)

3-coloring Graphs with Small Dominating Sets

How to check whether a given 3-coloring of $G[D]$ can be extended to a 3-coloring of G?

Reduce to 2-SAT.

3-coloring Graphs with Small Dominating Sets

for each $v \in V(G) \setminus D$ **do**

$L(v) =$ list of allowed colors at v

end for

(Since D is a dominating set, $|L(v)| \leq 2$ *for all* v*.)*

if some $L(v)$ is empty **return** NO;

(the coloring cannot be extended)

else . . .

3-coloring Graphs with Small Dominating Sets

Create input instance $I(c)$ for 2-SAT:

- variables:
$$x_v \ : v \in V(G) \setminus D\,.$$
meaning: $x_v = \top \;\Leftrightarrow\; c(v) = \min L(v)$

3-coloring Graphs with Small Dominating Sets

- clauses:
  - for all $v \in V(G) \setminus D$ with $|L(v)| = 1$ add the clause

$$C_v = x_v \ \vee \ x_v$$

  - for each edge $uv \in E(G \setminus D)$ and for each color $\gamma \in L(u) \cap L(v)$ add the clause

$$C_{uv,\gamma} = \overline{\lambda_{u,\gamma}} \ \vee \ \overline{\lambda_{v,\gamma}}\,;$$

    where

$$\lambda_{u,\gamma} = \begin{cases} x_u, & \text{if } \gamma = \min L(u); \\ \overline{x_u}, & \text{else.} \end{cases}$$

    The above clause is equivalent to the implication $\lambda_{u,\gamma} \ \Rightarrow \ \overline{\lambda_{v,\gamma}}\,.$
    meaning : $\lambda_{u,\gamma} = \top \ \Leftrightarrow \ c(u) = \gamma$.

3-coloring Graphs with Small Dominating Sets

return answer to 2-SAT on the input $I(c)$.

Correctness of the algorithm:

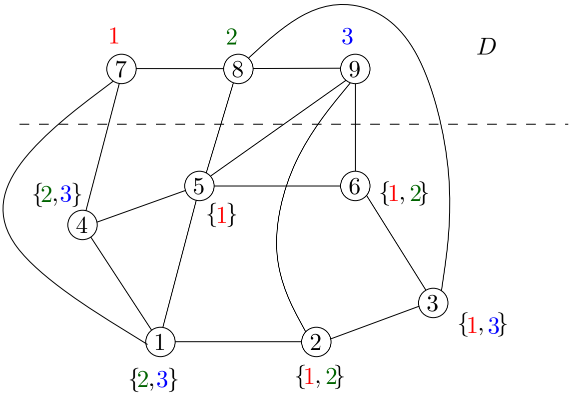
- G is 3-colorable if and only if some $c \in \mathcal{C}_3$ can be extended to a 3-coloring of G.
- A given $c \in \mathcal{C}_3$ can be extended to a 3-coloring of G if and only if $I(c)$ is satisfiable.

Time complexity:

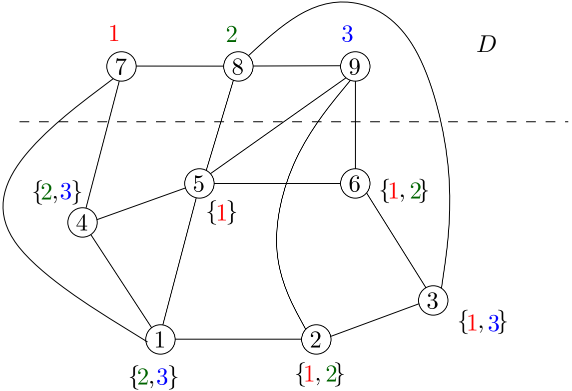
$$O(n^k + 3^k(n+m))$$

- We need to solve 3^k instances of 2-SAT, each of which has $O(n)$ variables and $O(n+m)$ clauses.

Example:



Example:



variables: $x_1, \ldots, x_6$

clauses:

$C_5 = x_5 \vee x_5$

$C_{12,2} = \overline{x_1} \vee x_2$

$C_{14,2} = \overline{x_1} \vee \overline{x_4}$

$C_{14,3} = x_1 \vee x_4$

$C_{23,1} = \overline{x_2} \vee \overline{x_3}$

$C_{36,1} = \overline{x_3} \vee \overline{x_6}$

$C_{56,1} = \overline{x_5} \vee \overline{x_6}$

Example:

implication digraph

variables: $x_1, \ldots, x_6$

clauses:

$C_5 = x_5 \vee x_5$

$C_{12,2} = \neg x_1 \vee x_2$

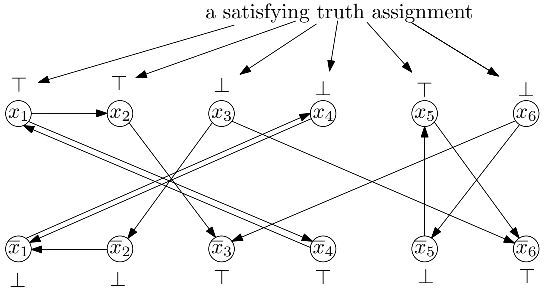
$C_{14,2} = \neg x_1 \vee \neg x_4$

$C_{14,3} = x_1 \vee x_4$

$C_{23,1} = \neg x_2 \vee \neg x_3$

$C_{36,1} = \neg x_3 \vee \neg x_6$

$C_{56,1} = \neg x_5 \vee \neg x_6$



$S(x_1) = \{x_1, x_2, \overline{x}_3, \overline{x}_4\}$

$S(x_2) = \{x_2, \overline{x}_3\}$

$S(x_3) = \{x_3, x_4, \overline{x}_1, \overline{x}_2, \overline{x}_6\}$

$S(x_4) = \{x_4, \overline{x}_1\}$

$S(x_5) = \{x_5, \overline{x}_6\}$

$S(x_6) = \{x_5, x_6, \overline{x}_3, \overline{x}_5, \overline{x}_6\}$

$S(\overline{x}_1) = \{x_4, \overline{x}_1\}$

$S(\overline{x}_2) = \{x_4, \overline{x}_1, \overline{x}_2\}$

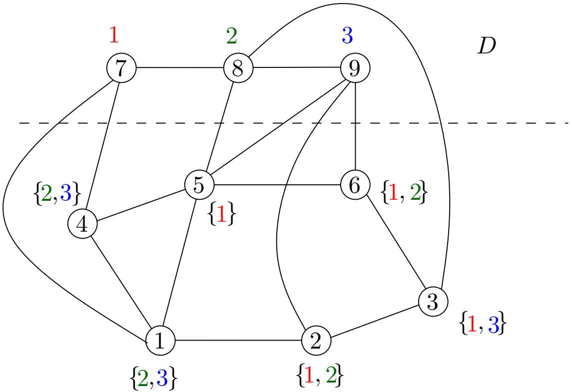
$S(\overline{x}_3) = \{\overline{x}_3\}$

$S(\overline{x}_4) = \{x_1, x_2, \overline{x}_3, \overline{x}_4\}$

$S(\overline{x}_5) = \{x_5, \overline{x}_5, \overline{x}_6\}$

$S(\overline{x}_6) = \{\overline{x}_6\}$

Example:



variables: $x_1, \ldots, x_6$

clauses:

$C_5 = x_5 \vee x_5$

$C_{12,2} = \overline{x_1} \vee x_2$

$C_{14,2} = \overline{x_1} \vee \overline{x_4}$

$C_{14,3} = x_1 \vee x_4$

$C_{23,1} = \overline{x_2} \vee \overline{x_3}$

$C_{36,1} = \overline{x_3} \vee \overline{x_6}$

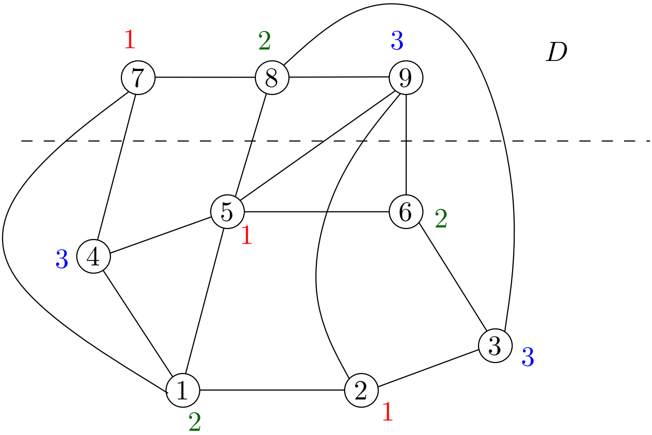
$C_{56,1} = \overline{x_5} \vee \overline{x_6}$

A satisfying assignment:

$x_1 = x_2 = x_5 = \top, x_3 = x_4 = x_6 = \bot$

Example:

a 3-coloring of G:



THE INDEPENDENT SET PROBLEM.

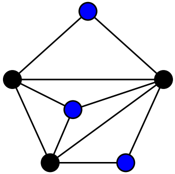
What we'll do

1. The Independent Set Problem in Hereditary Graph Classes.
2. Matchings and the IS problem.
3. Augmenting Graphs.
4. Decompositions.
5. Combining the Methods.

THE INDEPENDENT SET PROBLEM IN HEREDITARY GRAPH CLASSES.

Graphs and Independent Sets

- $G = (V, E)$ - a finite simple undirected graph
- independent set: a subset of pairwise non-adjacent vertices
- $\alpha(G)$ = max size of an independent set in G [independence #]



The Independent Set Problem

INDEPENDENT SET

Input: Graph $G = (V, E)$, $k \in \mathbb{N}$

Question: Does G contain an independent set of size k?

The IS problem is NP-**hard**.

Complexity of the IS Problem

NP-hard

maximum degree at most 3

[Garey-Johnson 1977]

triangle-free

[Poljak 1974]

planar

[Garey-Johnson-Stockmeyer 1976]

polynomial

claw-free

[Minty 1980, Sbihi 1980]

chordal

[Gavril 1972]

cographs

[Corneil-Perl-Stewart 1984]

bipartite

Complexity of the Problem in Hereditary Classes

$\mathcal{M}$: a set of graphs

Recall:

A graph G is $\mathcal{M}$-free if it does not contain any graph from $\mathcal{M}$ as an induced subgraph.

X hereditary $\iff$ $X = \{\mathcal{M}\text{-free graphs}\}$ for some $\mathcal{M}$

$\mathcal{M}$: the set of *forbidden induced subgraphs* for X

The graphs $S_{i,j,k}$

Question

Under what conditions on $\mathcal{M}$ is the IS problem solvable in polynomial time in the class of $\mathcal{M}$-free graphs? When is it NP-*hard?*

We will provide some partial answers to these questions:

- a general hardness result
- various techniques for developing polynomial time algorithms in restricted classes

A Hardness Result

$\mathcal{S}$ = the set of graphs whose **every connected component is a tree with at most three leaves**

Theorem (Alekseev 1982)

Let $\mathcal{M}$ be a finite set with $\mathcal{M} \cap \mathcal{S} = \emptyset$. Then, the IS problem is NP-**hard** *in the class of $\mathcal{M}$-free planar graphs of maximum degree at most* 3.

Proof idea:
Reduction from the (NP-hard) IS problem in planar graphs of maximum degree at most 3:
Given a planar graph G with $\Delta(G) \leq 3$, construct a graph G' by replacing each edge with a P_4. Then $\alpha(G') = \alpha(G) + |E(G)|$.
Repeating the reduction sufficiently many times produces an $\mathcal{M}$-free graph. $\square$

The IS Problem in $\mathcal{M}$-free Planar Graphs

Exercise

Show that replacing an edge by a P_4 increases the independence number by exactly 1.

Example:

The IS problem is NP-hard for:

- triangle-free planar graphs
- $\mathcal{M}$-free planar graphs where $\mathcal{M}$ is any finite set of graphs with cycles

What if $M \cap \mathcal{S} \neq \emptyset$?
Is this a sufficient condition for $\mathcal{M}$ such that the IS problem is polynomial for $\mathcal{M}$-free graphs?

Open question!

Polynomial Approaches

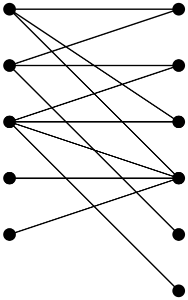
Since the 1960s, several approaches have been developed for solving the IS problem optimally in polynomial time in hereditary graph classes:

- matching techniques
- augmenting graphs
- divide-and-conquer approach whenever the input graph admits a tree-like decomposition (e.g. for cographs)
- dynamic programming for graphs which admit a recursive, linear decomposition (e.g. for interval graphs)
- local transformations
- polyhedral optimization
- semidefinite programming
- algebraic methods (e.g. struction)

MATCHINGS AND THE IS PROBLEM.

Bipartite Graphs

A graph G is bipartite if there exists a partition of the vertex set into two parts such that every edge has one endpoint in each part.

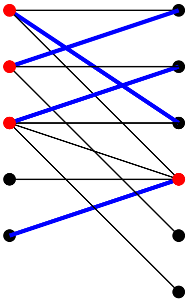


Bipartite Graphs

Theorem (Kőnig-Egerváry 1931)

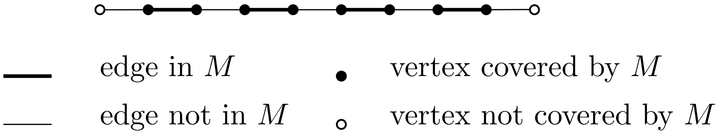
In a bipartite graph G, the maximum size of a matching equals the minimum size of a vertex cover.

- matching = a set of pairwise disjoint edges
- vertex cover = a subset of vertices covering all edges



Finding Maximum Matchings

Augmenting path for M:



Theorem (Petersen 1891, Berge 1957)

M is maximum $\iff$ there are no augmenting paths for M.

Kőnig gave an $O(m)$ algorithm that finds an augmenting path for a given matching.

$\Longrightarrow$ A maximum matching in a bipartite graph can be found in time $O(nm)$. The algorithm also produces a minimum vertex cover.

Maximum Independent Sets in Bipartite Graphs

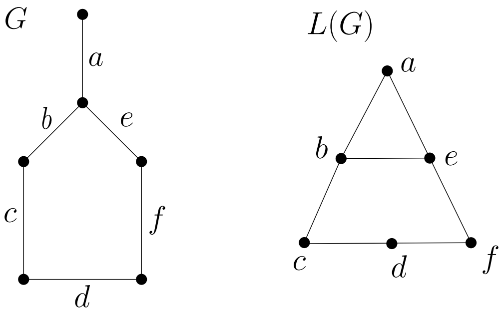
A maximum independent set in a bipartite graph $G = (V, E)$ can also be found in polynomial time:

1. Compute a maximum matching M.
2. Using M, compute a minimum vertex cover C.
3. The set $V \setminus C$ is a maximum independent set.

Line Graphs

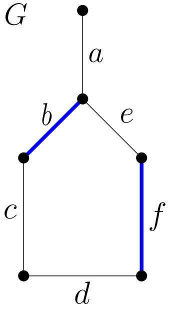
Line graph $L(G)$

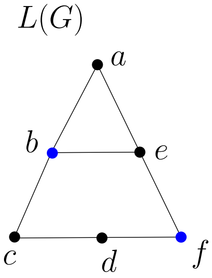
- vertices are edges of G
- two are adjacent iff they share a common vertex in G



Matchings = Independent Sets in Line Graphs

- matching in $G \equiv$ independent set in $L(G)$





Matchings = Independent Sets in Line Graphs

IS problem in line graphs = maximum matching problem in general graphs

- Edmonds 1965: a polynomial time algorithm for the maximum matching problem

J. Edmonds, *Paths, trees, and flowers*, Canad. J. Math. 17 (1965) 449-467.

- Roussopoulos 1973: a linear-time algorithm to determine G from its line graph $L(G)$

Corollary

The IS problem can be solved in polynomial time in the class of line graphs.

AUGMENTING GRAPHS.

Augmenting Graphs

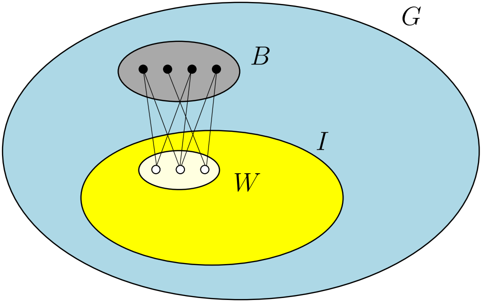
A natural approach for solving optimization problems is the following:

1. Start with some feasible solution.
2. Move to a better solution, if possible.
3. Repeat step 2.

In the case of the IS problem, step 2 is formalized by means of **augmenting graphs**.

Augmenting Graphs

G graph, I independent set in G

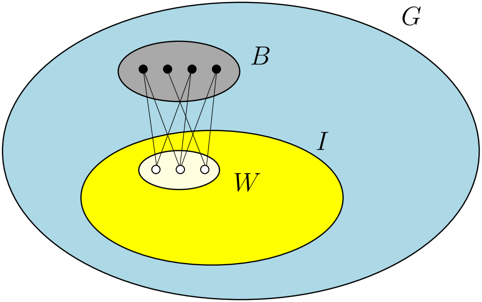


An augmenting graph for I is an induced bipartite subgraph $H = (W, B; E)$ of G with

- $W \subseteq I$,
- $B \subseteq V \backslash I$,
- $|B| > |W|$, and
- no edges from B to $I \backslash W$.

Augmenting Graphs

G graph, I independent set in G

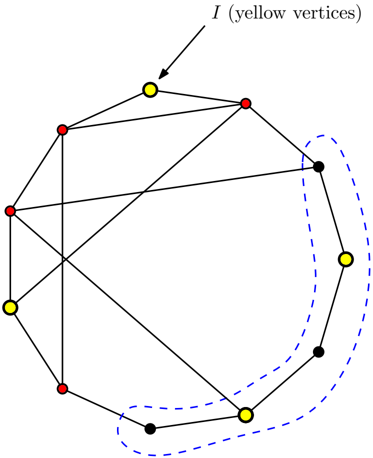


The set $I' = (I \backslash W) \cup B$ is independent with $|I'| > |I|$.

Augmenting Graphs

$W \subseteq I$, $B \subseteq V \backslash I$, $|B| > |W|$, and no edges from B to $I \backslash W$.

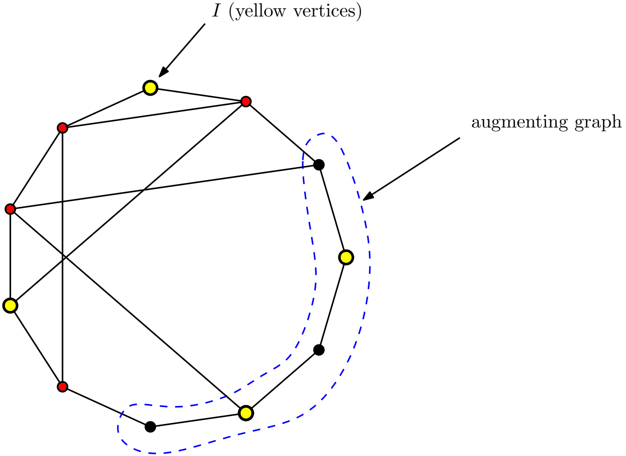
Example:



Augmenting Graphs

$W \subseteq I$, $B \subseteq V \backslash I$, $|B| > |W|$, and no edges from B to $I \backslash W$.

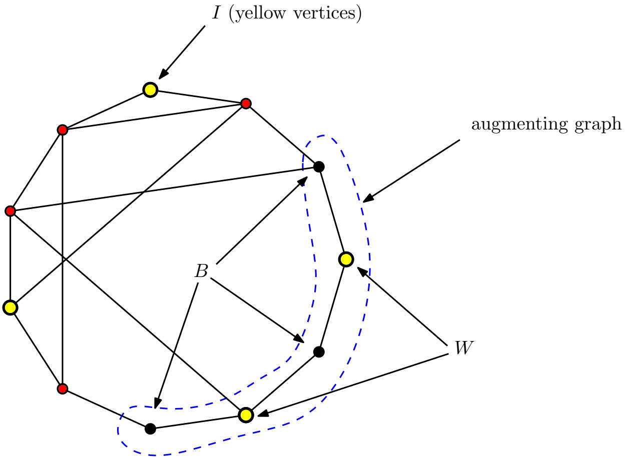
Example:



Augmenting Graphs

$W \subseteq I$, $B \subseteq V \backslash I$, $|B| > |W|$, and no edges from B to $I \backslash W$.

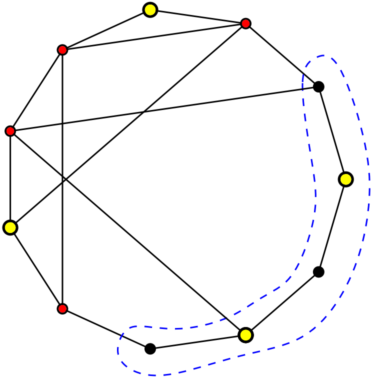
Example:



Augmenting Graphs

$W \subseteq I$, $B \subseteq V \backslash I$, $|B| > |W|$, and no edges from B to $I \backslash W$.

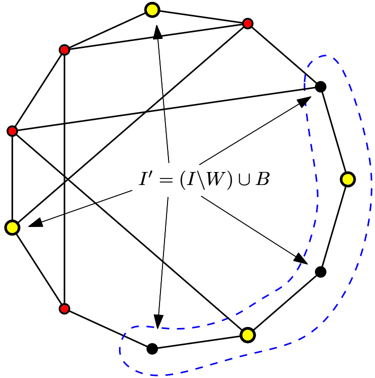
Example:



Augmenting Graphs

$W \subseteq I$, $B \subseteq V \backslash I$, $|B| > |W|$, and no edges from B to $I \backslash W$.

Example:



The Method of Augmenting Graphs

Theorem of augmenting graphs

I is maximum $\iff$ *there are no augmenting graphs for I.*

To find a maximum independent set in G:

1. Start with any independent set I.
2. Find an augmenting graph for I.
3. Augment the set, and go to 2.

- At most n augmentations.
- $\Rightarrow$ Finding augmenting graphs is NP-hard.

The Method of Augmenting Graphs

Finding augmenting graphs is NP-hard for general graphs. But it can be solved in polynomial time for particular graph classes.

It suffices to consider **minimal** augmenting graphs:

- An augmenting graph for I is minimal if it does not contain any smaller augmenting graph for I.

Proposition

An augmenting graph $(W, B; E)$ *for* I *is minimal if and only if* $|B| = |W| + 1$ *and for every nonempty subset* $X \subseteq W$, $|N(X)| > |X|$.

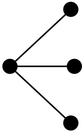
Exercise

Prove the above proposition.

The Method of Augmenting Graphs

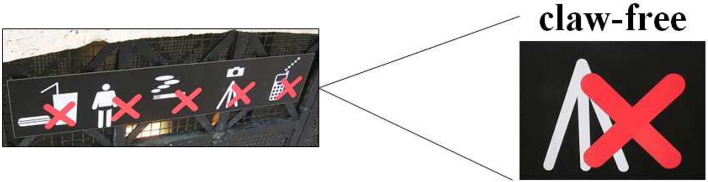
To solve IS in a class of graphs X by augmentation, we have to:

1. *Characterize minimal augmenting graphs in* X, and
2. *Develop a poly-time procedure for finding them.*



claw = $K_{1,3}$

claw-free graphs $\supset$ line graphs



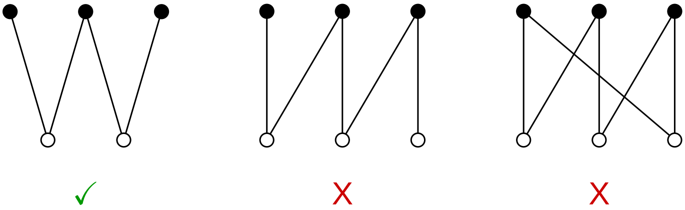
Application 1: Claw-free Graphs

Which claw-free graphs are augmenting?

- bipartite and claw-free $\Rightarrow$ maximum vertex degree is 2
- bipartite of max degree 2 $\Rightarrow$ paths and even cycles

Which claw-free graphs are augmenting?

- bipartite and claw-free $\Rightarrow$ maximum vertex degree is 2
- bipartite of max degree 2 $\Rightarrow$ paths and even cycles
- augmenting $\Rightarrow$ the number of vertices must be odd



Application 1: Claw-free Graphs

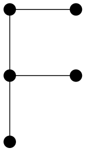
The only claw-free augmenting graphs are
paths with an even number of edges = augmenting chains

Theorem (Minty 1980, Sbihi 1980, Lovász-Plummer 1986)

A maximum independent set in a claw-free graph can be found in polynomial time.

- Sbihi's solution is based on the augmenting graph method.

Application 2: Fork-free Graphs



fork

fork-free graphs $\supset$ claw-free graphs

Theorem (Alekseev 1999)

The IS problem is solvable in polynomial time in the class of fork-free graphs.

The algorithm is based on the augmenting graph method.

Application 2: Fork-free Graphs

There only minimal fork-free augmenting graphs are:

- augmenting chains,
- augmenting complexes: bipartite graphs every vertex of which contains at most one non-neighbor in the opposite part.

- If G contains both a path P_k with $k \geq 8$ and a claw as induced subgraphs, then G admits a decomposition reducing the problem to subgraphs with fewer vertices.
- The problem of finding an augmenting complex of maximum increment in G is recursively reduced to the IS problem in induced subgraphs of G.

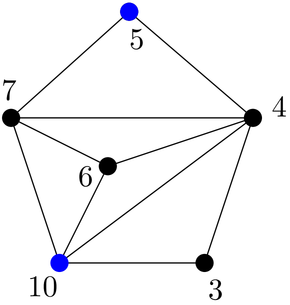
DECOMPOSITIONS.

Decompositions.

1. Decomposition by clique separators
2. Modular decomposition
3. Tree decompositions and graphs of bounded treewidth
4. Graphs of bounded clique-width

The Weighted Independent Set Problem

WEIGHTED INDEPENDENT SET (WIS) Problem:
Input: $G = (V, E)$, $w : V \to \mathbb{N}$
Task: Compute $\alpha_w(G)$ = max weight of an IS.



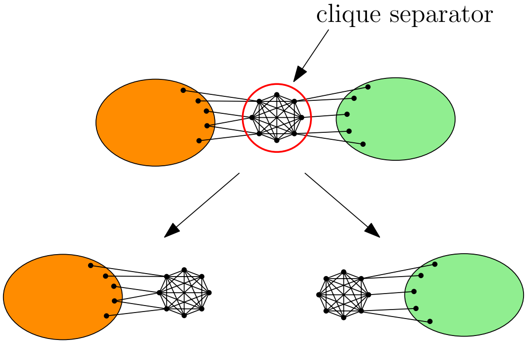
$\alpha_w(G) = 15$

Decomposition by Clique Separators

Theorem (Whitesides 1981, Tarjan 1985)

Given a graph G, its decomposition by clique separators can be computed in polynomial time. It reduces the WIS problem to graphs without clique separators.

separator = cutset = a set X such that $G - X$ is disconnected



Application: Trees and Chordal Graphs

Trees: every tree on at least three vertices has a clique separator.

Corollary

The WIS problem is solvable in polynomial time for trees.

Chordal graphs:

Theorem (Dirac 1961): Every chordal graph is either complete, or has a clique separator.

Corollary

The WIS problem is solvable in polynomial time in the class of chordal graphs.

Application: 2-separable Graphs

Definition

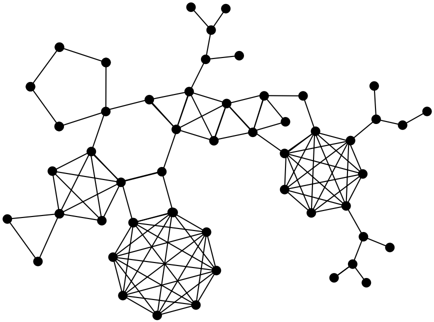
A graph G is 2-separable if every two nonadjacent vertices can be separated by at most two other vertices.

Complete graphs, cycles are 2-separable.

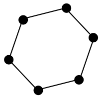
Theorem (Cicalese-M. 2012)

A connected graph G is 2-separable if and only if G arises from complete graphs and cycles by pasting along vertices or edges.

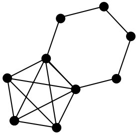
Application: 2-separable Graphs



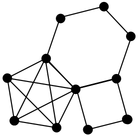
Application: 2-separable Graphs



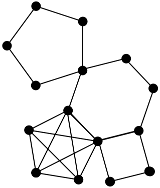
Application: 2-separable Graphs



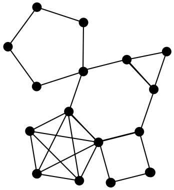
Application: 2-separable Graphs



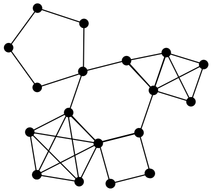
Application: 2-separable Graphs



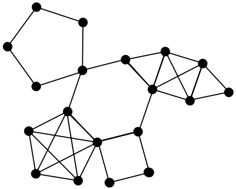
Application: 2-separable Graphs



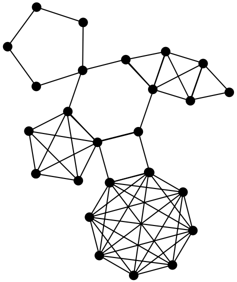
Application: 2-separable Graphs



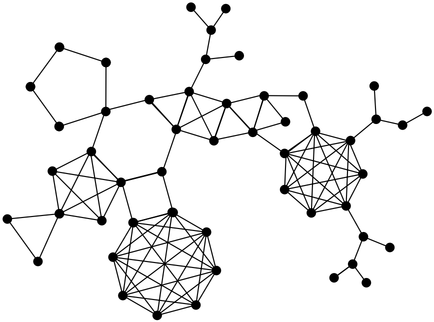
Application: 2-separable Graphs



Application: 2-separable Graphs



Application: 2-separable Graphs



Application: 2-separable Graphs

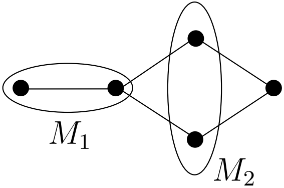
The WIS problem is solvable in polynomial time for complete graphs, and also for cycles (for cycles, we will see later why).

Corollary

The WIS problem is solvable in polynomial time for 2-separable graphs.

Modular Decomposition

$M \subseteq V(G)$ is a *module* of G if
every vertex outside M is adjacent to either
all vertices of M or to none of them.



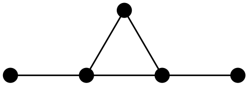
M_1 is not a module, M_2 is!

Modular Decomposition

Trivial modules:

- $M = V$
- $|M| \leq 1$

G is **prime** if its only modules are trivial.



a prime graph

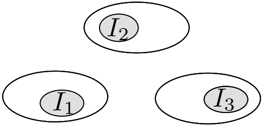
Modular Decomposition

Goal: Given a weighted graph G, compute $\alpha_w(G)$

Modular Decomposition

Goal: Given a weighted graph G, compute $\alpha_w(G)$

If G is disconnected:

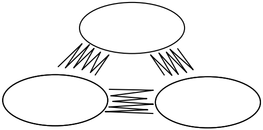


- solve the problem on components and combine by taking the union

$$\alpha_w(G) = \sum_{i=1}^{k} \alpha_w(G_i)\,.$$

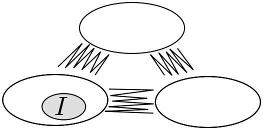
Modular Decomposition

If $\overline{G}$ is disconnected:



Modular Decomposition

If $\overline{G}$ is disconnected:

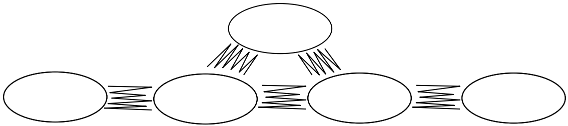


- solve the problem on co-components and combine by taking the heaviest solution

$$\alpha_w(G) = \max_{1 \le i \le k} \alpha_w(G_i)\,.$$

Modular Decomposition

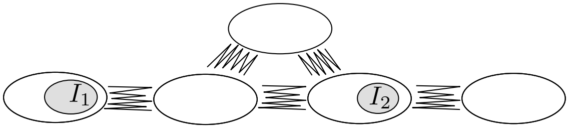
If both G and $\overline{G}$ are connected, then we can partition $V(G)$ into maximal modules:



module: $U \subseteq V(G)$ such that $\forall\, v \in V \backslash U,\ N(v) \cap U \in \{\emptyset, U\}$

Modular Decomposition

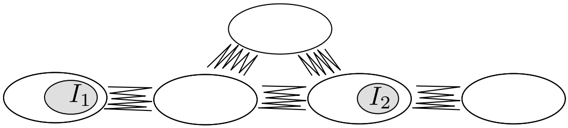
If both G and $\overline{G}$ are connected, then we can partition $V(G)$ into maximal modules:



module: $U \subseteq V(G)$ such that $\forall\, v \in V \backslash U$, $N(v) \cap U \in \{\emptyset, U\}$

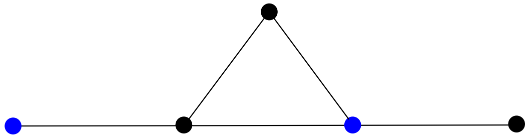
Modular Decomposition

If both G and $\overline{G}$ are connected, then we can partition $V(G)$ into maximal modules:

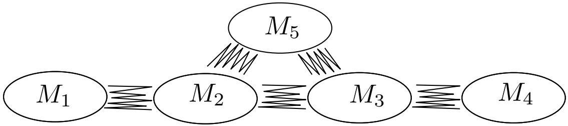


module: $U \subseteq V(G)$ such that $\forall\, v \in V \setminus U$, $N(v) \cap U \in \{\emptyset, U\}$

$\rightarrow$ characteristic graph



Modular Decomposition



- compute $\alpha_w(G[M_1]), \ldots, \alpha_w(G[M_k])$ recursively

Let $w'(M_i) = \alpha_w(M_i)$.
Then

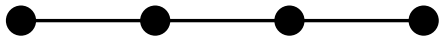
$$\alpha_w(G) = \alpha_{w'}(G')\,.$$

Modular Decomposition

Theorem

The modular decomposition tree is unique and can be computed in linear time.
It reduces the WIS problem in a class of graphs X to prime induced subgraphs of graphs in X.

Application 1: Cographs



P_4

cographs = P_4-free graphs

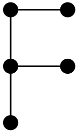
Every cograph with at least two vertices is either disconnected, or its complement is disconnected.

In particular, the only prime cograph is K_1.

Corollary

The WIS problem is solvable in polynomial time for cographs.

Application 2: Fork-free Graphs



fork

Theorem (Alekseev 1999)

A maximum independent set in a fork-free graph can be found in polynomial time.

- augmenting graph approach
- running time: $O(n^{10})$

Application 2: Fork-free Graphs

Theorem (Lozin–M. 2008)

An independent set of maximum weight in a fork-free graph can be found in time $O(nT)$, where T is the time needed to solve the same problem in claw-free graphs.

Theorem

An independent set of maximum weight in a claw-free graph can be found in polynomial time T.

Minty 1980, Nakamura-Tamura 2001, Oriolo-Pietropaoli-Stauffer 2008, Faenza-Oriolo-Stauffer 2011

- $T = O(n^7) \to O(n^3)$

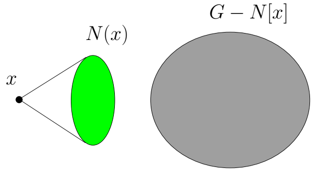
Application 2: Fork-free Graphs

Improvements over Alekseev's result:

- unweighted $\rightarrow$ weighted
- improved time complexity ($O(n^4)$ instead of $O(n^{10})$).

Anti-Neighborhoods

$$\alpha_w(G) = \max_{x \in V(G)} \{w(x) + \alpha_w(G - N[x])\}$$



Main observation:

G prime fork-free graph, $x \in V(G)$.
Then, every prime induced subgraph of $G - N[x]$ is claw-free.

Treewidth and Tree Decompositions

Definition

A *tree-decomposition* of a graph $G = (V, E)$:
a tree $T = (\mathcal{I}, F)$ where each vertex $i \in \mathcal{I}$ has a label $X_i \subseteq V$ such that:

(i) $\cup_{i \in \mathcal{I}} X_i = V$,

(ii) For every edge $uv \in E$, there exists an $i \in \mathcal{I}$ such that $u, v \in X_i$, and

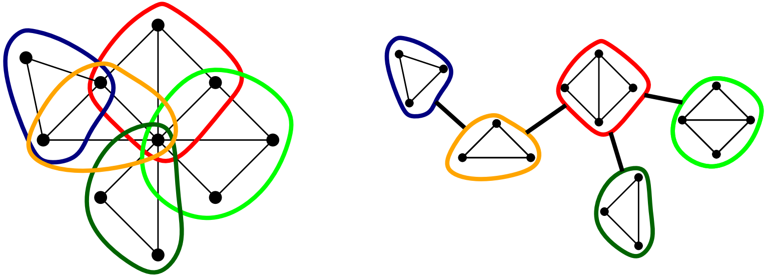
(iii) For every $v \in V$, the vertices of T whose label contains v induce a connected subgraph of T.

The *width* of such a decomposition is $\max_{i \in \mathcal{I}} |X_i| - 1$.

The treewidth of a graph G is the minimum k such that G has a tree-decomposition of width k.

Example:

A graph with a tree decomposition of width 3:



Equivalently:

The treewidth of G is the minimum clique number over all chordal supergraphs of G, minus one.

Example:
Cycles are of treewidth two.

Exercise

Determine the treewidth of complete graphs K_n and of complete bipartite graphs $K_{m,n}$.

Treewidth and Tree Decompositions

Many problems that are generally NP-hard can be solved in polynomial time on graphs of tree-width at most k, for every fixed k.

Theorem (Courcelle's Theorem, 1990)

Every property of graphs expressible by a monadic second order sentence with quantifiers over vertex and edge sets is decidable in linear time on graphs of tree-width at most k.

Courcelle 1990, Arnborg-Lagergren-Seese 1991, Borie-Parker-Tovey 1991, Courcelle-Mosbah 1993

For the WIS problem, a direct algorithm with time complexity $O(2^k n)$ can be developed.

Graphs of Bounded Clique-Width

- clique-width = another important graph parameter generalizing treewidth
- $cwd(G)$ = minimum number of labels needed to construct G using the following four operations:
  - **(i)** $\boldsymbol{i(v)}$: creation of a new vertex v with label i
  - **(ii)** $\boldsymbol{G \oplus H}$: disjoint union of two labeled graphs G and H
  - **(iii)** $\boldsymbol{\eta_{i,j}}$: joining by an edge each vertex with label i to each vertex with label j (where $i \neq j$)
  - **(iv)** $\boldsymbol{\rho_{i \to j}}$: renaming label i to j

An expression formed with the above operations is called a k**-expression** if it uses at most k labels.

Example: A path (a, b, c, d, e) can be defined with the following 3-expression:

$$\eta_{3,2}(3(e) \oplus \rho_{3\to 2}(\rho_{2\to 1}(\eta_{3,2}(3(d) \oplus \rho_{3\to 2}(\rho_{2\to 1}(\eta_{3,2}(3(c) \oplus \eta_{2,1}(2(b) \oplus 1(a)))))))))$$

Graphs of Bounded Clique-Width

Exercise

Prove that for every k, $cwd(P_k) \leq 3$.

Exercise

Prove that for every tree T, $cwd(T) \leq 3$.

Graphs of Bounded Clique-Width

- Many problems that are NP-hard in general are polynomially solvable for $\{G : cwd(G) \leq k\}$
- P_4-free graphs $\equiv$ graphs of clique-width ≤ 2
- graphs of bounded treewidth are of bounded clique-width: $cwd(G) \leq 3 \cdot 2^{twd(G)-1}$ [Corneil-Rotics 2005]
- but not vice-versa (complete graphs are of clique-width ≤ 2 but have unbounded treewidth)

Graphs of Bounded Clique-Width

- for graphs of cwd $\leq k$, it is easy to develop a direct poly-time solution to the WIS problem
- More generally:

Theorem (Courcelle-Makowsky-Rotics 1993)

Every optimization problem expressible by a monadic second order sentence with quantifiers over vertices and vertex sets is solvable in linear time, given a k-expression defining the input graph.

COMBINING THE METHODS.

atom of G = an induced subgraph of G that has no clique separators

Theorem (Brandstädt-Hoáng 2007)

If WIS is solvable in time T on prime atoms of a graph G then it is solvable in time n^2T on G.

Applications: poly time algorithms for the WIS problem in:

- (P_5,banner)-free graphs (Brandstädt-Hoáng 2007)

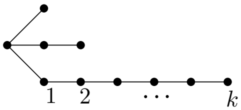
  Every anti-neighborhood graph in a prime (P_5,banner)-free atom is chordal.

- several other subclasses of P_5-free graphs (Brandstädt-Le-Mahfud 2007)

$S_{1,2,k}$-free Planar Graphs

Theorem (Lozin-M. 2010)

A maximum independent set in an $S_{1,2,k}$-free planar graph can be found in polynomial time.



$S_{1,2,k}$

Ingredients of the proof:

- reduction from $S_{1,2,k}$-free graphs to $S_{1,2,2}$-free graphs via bounded treewidth,
- decomposition by clique separators (reduction to 2-connected components),
- method of augmenting graphs.

Two Open Problems

Two open questions:

- Is there a hereditary graph class X such that the IS problem is polynomially solvable for graphs in X, while the WIS problem is NP-hard?
- What is the complexity of the IS problem in the class of P_5-free graphs?

MAX CUT PROBLEM IN PLANAR GRAPHS.

The Max Cut Problem

Max Cut	
Input:	Multigraph $G = (V, E)$, $k \in \mathbb{N}$
Question:	Does V admit a cut of size at least k?

cut: a partition of V into two pairwise disjoint sets A and B
size of a cut (A, B): the number $|E(A, B)|$, where
$E(A, B) = E \cap \{ab : a \in A, b \in B\}$.

The Max Cut problem is equivalent to asking:
Does the input graph G have a bipartite subgraph with k edges?

The NP-completeness of the Max Cut Problem

Theorem (Karp, 1972)

The MAX CUT *problem is* NP*-complete.*

Proof:

1. MAX CUT $\in$ NP : we can verify in polynomial time whether (A, B) is a cut of size at least k.

The NP-completeness of the Max Cut Problem

2. Reduction from VERTEX COVER.
$I = (G = (V, E), k)$ instance for VERTEX COVER;
we may assume G has no isolated vertices
$\mapsto$
$J(I) = (G', 2|E| - k)$ instance for MAX CUT, where
G' is a graph obtained from G by adding to it a new vertex u and connecting each $v \in V$ with $d_G(v) - 1$ parallel edges to u.

Exercise

Prove that G contains a vertex cover of size k if and only if G' contains a cut of size at least $2|E| - k$.

The Max Cut Problem in Planar Graphs

Theorem (Hadlock, 1975)

The MAX CUT *problem is solvable in polynomial time if G is planar.*

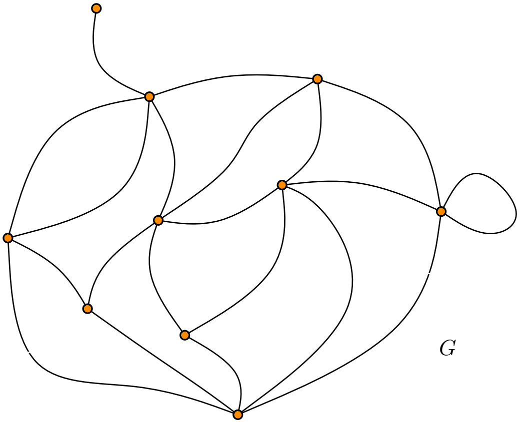
To explain this result, we need to recall some facts about duals of planar graphs.

Duality

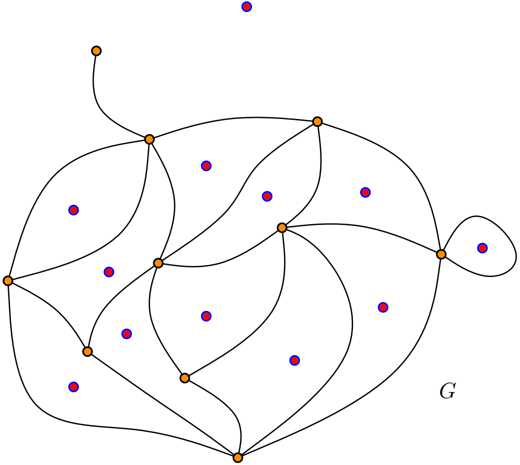
Given a connected **plane multigraph**
(= a planar multigraph embedded in the plane) G,
we create its **dual multigraph** G^*, as follows:

- Create a vertex for each face of G, and
- For each edge e of G, create an edge e^* connecting the two vertices corresponding two the two faces e is incident with.

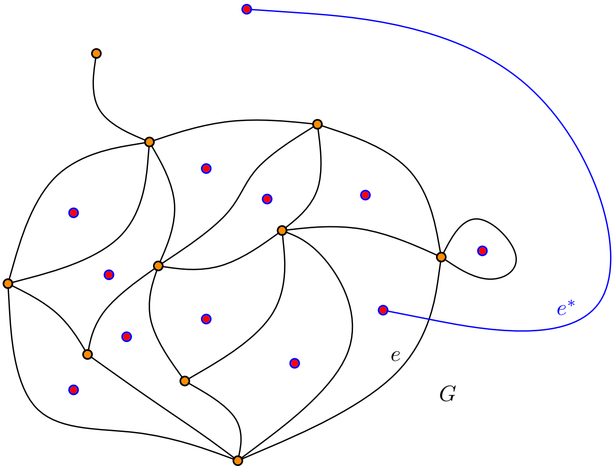
Duality – Example



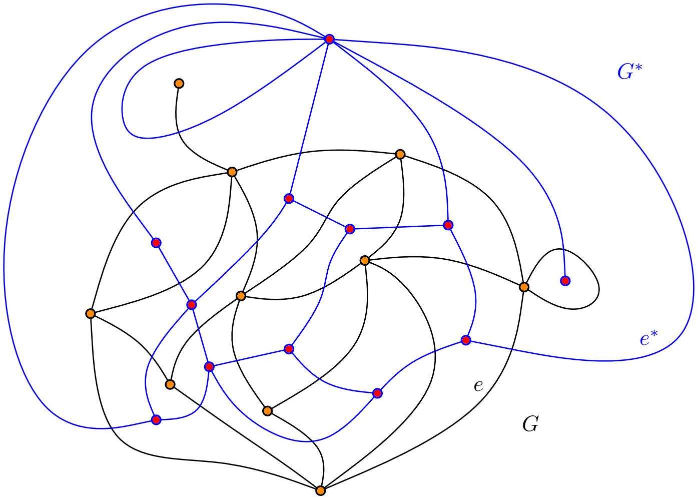
Duality – Example



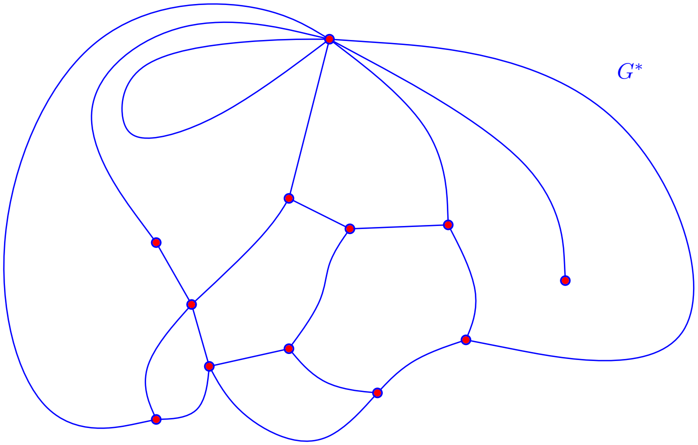
Duality – Example



Duality – Example



Duality – Example



Duality

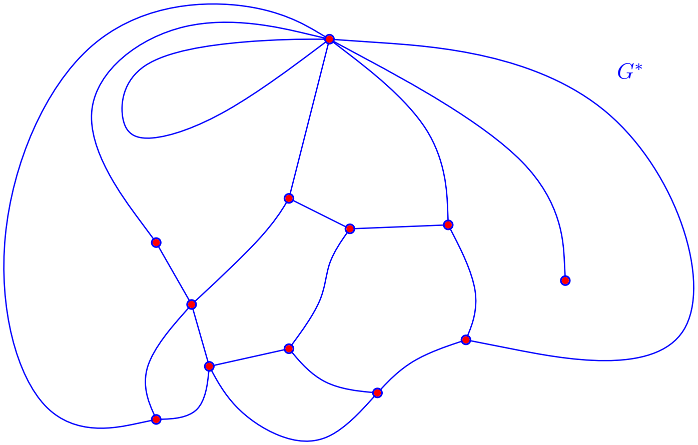
Proposition

$(G^*)^* \cong G$.

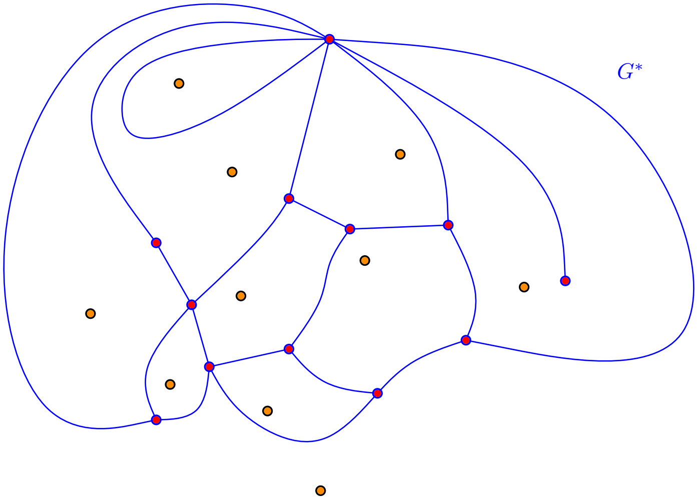
Moreover, there is a bijective correspondence between:

- *vertices of G and faces of G^*,*
- *edges of G and edges of G^*,*
- *faces of G and vertices of G^*.*

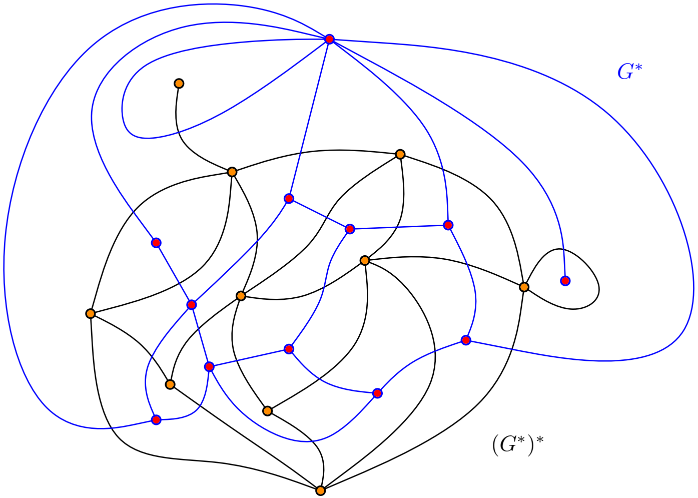
Duality – Example



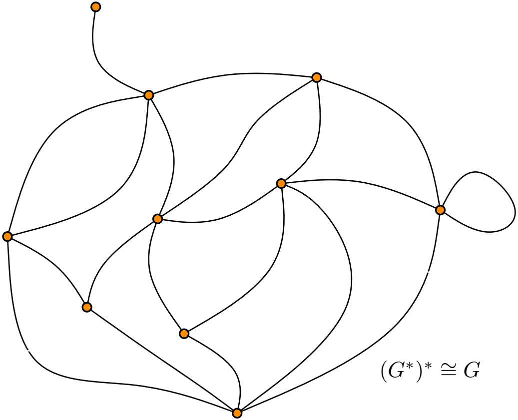
Duality – Example



Duality – Example

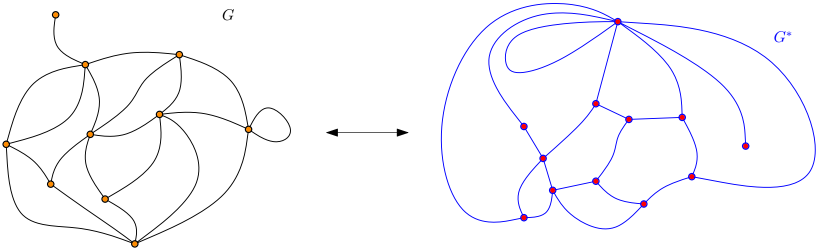


Duality – Example



$(G^*)^* \cong G$

Duality – Example



The Max Cut Problem in Planar Graphs

PLANAR MAX CUT	
Input:	A planar multigraph $G = (V, E)$, $k \in \mathbb{N}$
Question:	Does V admit a cut of size at least k?

Theorem (Hadlock, 1975)

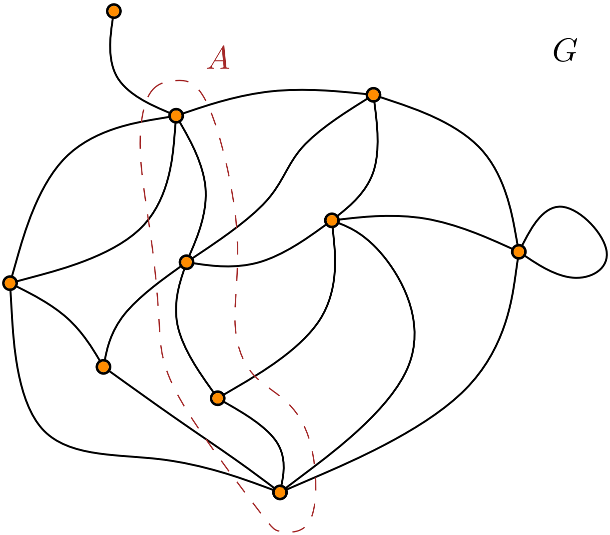
The PLANAR MAX CUT *problem is solvable in polynomial time.*

We will give a sketch of the proof of this theorem.

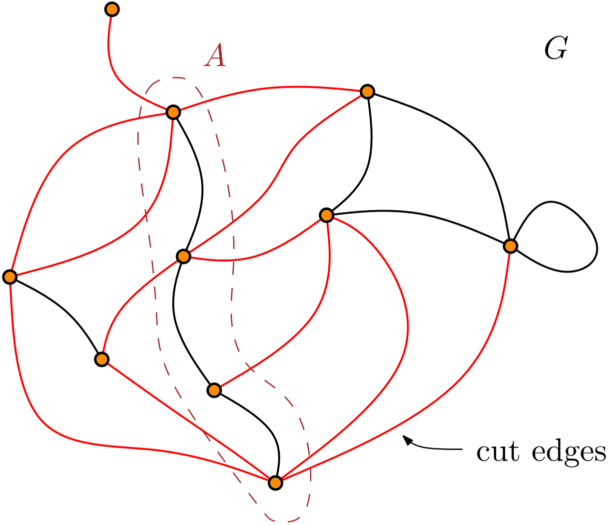
First, we may assume that

- G is connected (otherwise we solve the problem separately on connected components),
- G is embedded in the plane (there exists a linear time algorithm for finding a planar embedding of a planar multigraph).

The Max Cut Problem in Planar Graphs

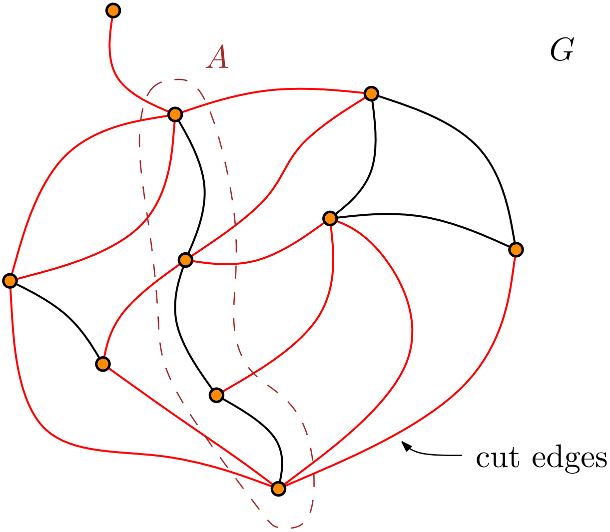


The Max Cut Problem in Planar Graphs



The Max Cut Problem in Planar Graphs

We may delete loops, as they will never count towards the size of a cut.



The Max Cut Problem and Odd Cycle Covers

Recall:
The MAX CUT problem is equivalent to verifying whether G has a bipartite subgraph with k edges.

Equivalently:
Does G have an odd cycle cover of at most $|E| - k$ edges?

odd cycle cover in a graph $G = (V, E)$: a subset $E' \subseteq E$ that intersects every odd cycle in G

The Max Cut Problem in Planar Graphs

Proposition

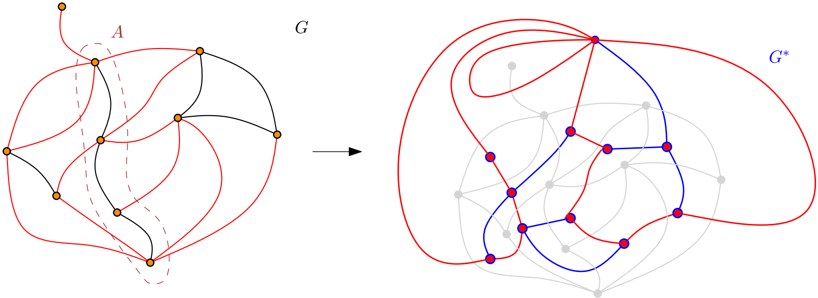
Let (A, B) be a cut in G. Then, the set

$$\{e^* \ : \ e \in E(A, B)\}$$

forms an even subgraph of G^*.

even multigraph: a multigraph in which every vertex has even degree (i.e., is contained in an even number of edges)

The Max Cut Problem in Planar Graphs



The Max Cut Problem in Planar Graphs

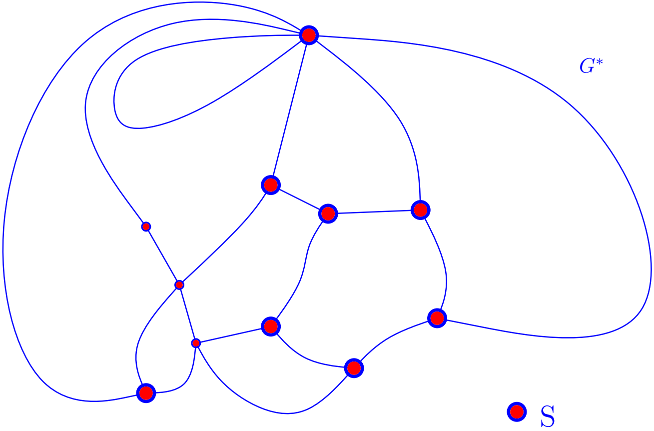
An **odd-vertex pairing** in a graph: a set of edges the removal of which makes the graph even.

Observation:
An edge set $E' \subseteq E$ is an odd-cycle cover of G if and only if the set $(E')^* = \{e^* \,:\, e \in E\}$ is an odd-vertex pairing of G^*.

Proposition

The PLANAR MAX CUT *problem on G is equivalent to the problem of finding a smallest odd-vertex pairing in G^*.*

Let S be the set of vertices of odd degree in G^*.



The Max Cut Problem in Planar Graphs

Proposition

Let E' be a smallest odd-vertex pairing in G^*. Then E' is the disjoint union of $|S|/2$ paths each connecting two different vertices in S.

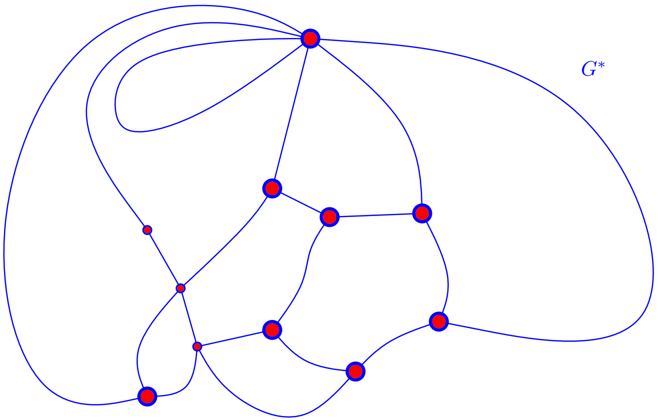
The problem can be reduced to that of finding a
perfect matching of minimum weight
in the complete graph K_S, with
$w(xy) =$length of a shortest x-y path in G^*.

perfect matching = a matching covering all vertices of a graph

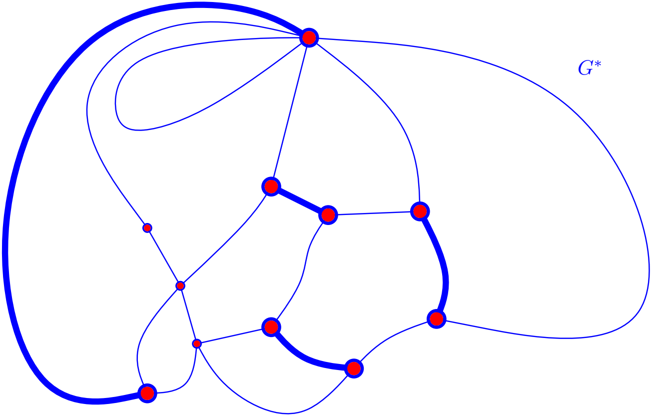
Theorem (Edmonds, 1965)

There exists a polynomial time algorithm to find a perfect matching of minimum weight in a given edge-weighted graph.

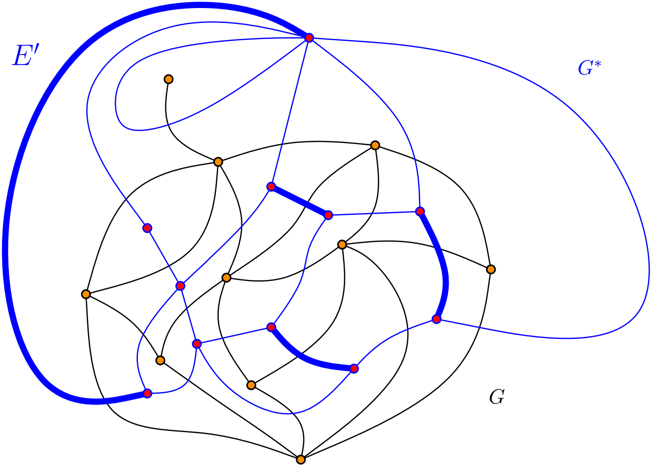
The Max Cut Problem in Planar Graphs



The Max Cut Problem in Planar Graphs

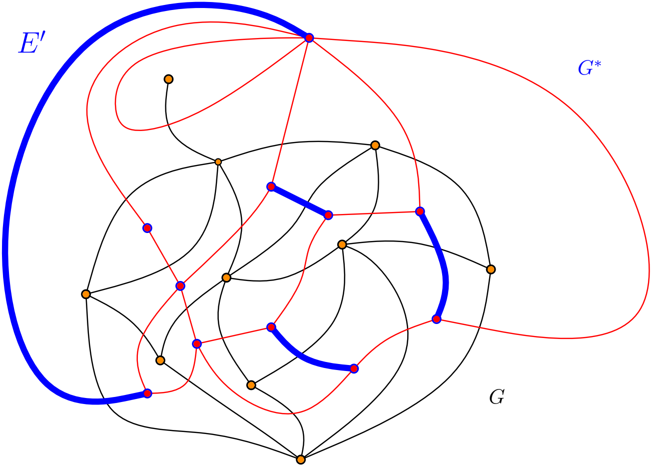


Let E' denote a smallest odd-vertex pairing in G^*.



The Max Cut Problem in Planar Graphs

Let E' denote a smallest odd-vertex pairing in G^*.

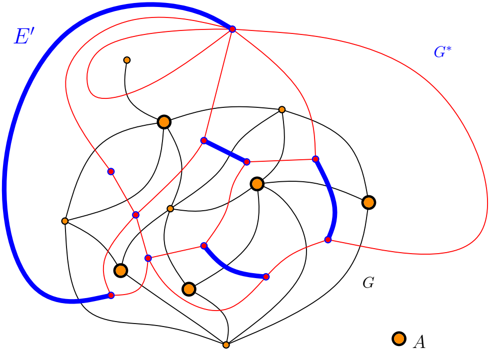


The Max Cut Problem in Planar Graphs

A maximum cut is given by any partition (A, B) such that

$$\{e \in E(G) \,:\, e^* \in E(G^*) \setminus E'\} = E(A, B)\,.$$

Such a partition can be found by placing a vertex in A, examining its neighbors to determine which ones are in B, and repeating this procedure. The fact that the graph $(V(G^*), E(G^*) \setminus E')$ is even will assure that no conflict will arise.



What we did – Week 1

1. Tue March 5: Review of basic notions in graph theory, algorithms and complexity ✓
2. Wed March 6: Graph colorings ✓
3. Thu March 7: Perfect graphs and their subclasses, part 1 ✓
4. Fri March 8: Perfect graphs and their subclasses, part 2 ✓

What we'll do – Week 2

1. Tue March 19: Further examples of tractable problems, part 1 ✓
2. Wed March 20:
   Further examples of tractable problems, part 2 ✓
   Approximation algorithms for graph problems
3. Thu March 21: Lectio Magistralis lecture, *"Graph classes: interrelations, structure, and algorithmic issues"*