Almost Consistent Systems of Linear Equations

George Osipov

PCCR 2022, Haifa

LINKÖPINGS UNIVERSITET

Join Work With

Konrad Dabrowski

Newcastle University
UK

Peter Jonsson

Linköping University
Sweden

Sebastian Ordyniak

University of Leeds
UK

Magnus Wahlström

Royal Holloway
University of London
UK

Systems of Linear Equations

A set of equations over some domain (e.g. the rationals).

$$2x - y = 1$$
$$x + y = 5$$
$$z - 2y = 1$$
$$w + 2y = 2$$
$$2z + w = 4$$

Is there an assignment that satisfies all equations?

Systems of Linear Equations

A set of equations over some domain (e.g. the rationals).

$$2x - y = 1$$
$$x + y = 5$$
$$z - 2y = 1$$
$$w + 2y = 2$$
$$2z + w = 4$$

Is there an assignment that satisfies all equations?
We can use e.g. Gaussian elimination.

Systems of Linear Equations

A set of equations over some domain (e.g. the rationals).

$$
\begin{aligned}
&2x - y = 1 \\
&x + y = 5 \\
&z - 2y = 1 \\
&w + 2y = 2 \\
&2z + w = 4
\end{aligned}
$$

Is there an assignment that satisfies all equations?
We can use e.g. Gaussian elimination.
For this example the answer is **no**.

Systems of Linear Equations

A set of equations over some domain (e.g. the rationals).

$$2x - y = 1$$
$$x + y = 5$$
$$z - 2y = 1$$
$$w + 2y = 2$$
$$2z + w = 4$$

$$x = 2$$
$$y = 3$$

Is there an assignment that satisfies all equations?
We can use e.g. Gaussian elimination.
For this example the answer is **no**.

Systems of Linear Equations

A set of equations over some domain (e.g. the rationals).

$$2x - y = 1$$
$$x + y = 5$$
$$z - 2y = 1$$
$$w + 2y = 2$$
$$2z + w = 4$$

$$x = 2$$
$$y = 3$$
$$z = 7$$

Is there an assignment that satisfies all equations?
We can use e.g. Gaussian elimination.
For this example the answer is **no**.

Systems of Linear Equations

A set of equations over some domain (e.g. the rationals).

$$2x - y = 1$$
$$x + y = 5$$
$$z - 2y = 1$$
$$w + 2y = 2$$
$$2z + w = 4$$

$$x = 2$$
$$y = 3$$
$$z = 7$$
$$w = -12$$

Is there an assignment that satisfies all equations?
We can use e.g. Gaussian elimination.
For this example the answer is **no**.

Systems of Linear Equations

A set of equations over some domain (e.g. the rationals).

$$\begin{aligned} 2x - y &= 1 \\ x + y &= 5 \\ z - 2y &= 1 \\ w + 2y &= 2 \\ 2z + w &\neq 4 \end{aligned} \qquad \begin{aligned} x &= 2 \\ y &= 3 \\ z &= 7 \\ w &= -12 \\ 2 \cdot 7 - 12 &\neq 4 \end{aligned}$$

Is there an assignment that satisfies all equations?
We can use e.g. Gaussian elimination.
For this example the answer is **no**.

Systems of Linear Equations

A set of equations over some domain (e.g. the rationals).

$$2x - y = 1$$
$$x + y = 5$$
$$z - 2y = 1$$
$$w + 2y = 2$$
$$2z + w = 4$$

Is there an assignment that satisfies all equations? No.

What can we do?

MaxLin Problem

Max-r-Lin(D)

Given a linear system with at most r variables per equation, find an assignment of values from D to the variables that *maximizes the number of satisfied equations*.

MinLin Problem

Min-r-Lin(D)

Given a linear system with at most r variables per equation, find an assignment of values from D to the variables that *minimizes the number of unsatisfied equations*.

MinLin Problem

Min-r-Lin(D)

Given a linear system with at most r variables per equation, find an assignment of values from D to the variables that minimizes the number of unsatisfied equations.

- NP-hard for $r = 2$ and $D = \mathbb{F}_2$ (Max-2-Lin($\mathbb{F}_2$) = MaxCut).

MinLin Problem

Min-r-Lin(D)

Given a linear system with at most r variables per equation, find an assignment of values from D to the variables that minimizes the number of unsatisfied equations.

- NP-hard for r = 2 and D = $\mathbb{F}_2$ (Max-2-Lin($\mathbb{F}_2$) = MaxCut).
- UGC-hard to approximate within any constant.

Parametized Complexity of MinLin

Parameter is #unsatisfied equations.

Given a system of r-variable equations over D and an integer k, find an assignment leaves at most k equations unsatisfied.

Goal: find fpt algorithms = running in $f(k) \cdot n^{O(1)}$ time, where n is instance size and f() is some computable function.

Parametized Complexity of MinLin

Parameter is #unsatisfied equations.

Given a system of r-variable equations over D and an integer k, find an assignment leaves at most k equations unsatisfied.

Goal: find fpt algorithms = running in $f(k) \cdot n^{O(1)}$ time.

Contrast with straightforward $n^{O(k)}$ time subset enumeration.

Plan for This Talk

1. ~~Introduction~~
2. Related Work and Results
3. Biased Graphs 🛠️
4. Important Balanced Subgraphs 🛠️
5. Conclusion

Related Work

Min-r-Lin	$\mathbb{F}_2$	$\mathbb{F}_q$	$\mathbb{Q}$	$\mathbb{Z}$	$\mathbb{Z}_6$	$\mathbb{Z}_4$
r = 2	FPT [CGJY12]	FPT [CPPH12]	?	?	?	?
r > 2	W[1] [CGJY12]	?	?	?	?	?

- Min-r-Lin($\mathbb{F}_2$) studied by [CGJY12].
- Min-2-Lin($\mathbb{F}_2$) is equivalent to Graph Bipartization [RSV04, GGHNW06].
- Min-r-Lin($\mathbb{F}_q$) for any finite field $\mathbb{F}_q$ is a special case of Unique Label Cover [CPPH12, W14, IYY18].

Results

Min-r-Lin	$\mathbb{F}_2$	$\mathbb{F}_q$	$\mathbb{Q}$	$\mathbb{Z}$	$\mathbb{Z}_6$	$\mathbb{Z}_4$
r = 2	FPT [CGJY12]	FPT [CPPH12]	FPT	FPT	W[1]	?
r > 2	W[1] [CGJY12]	W[1]				

- We show that Min-2-Lin(D) is in FPT for any *Euclidean domain* D.
- For r > 2, Min-r-Lin becomes W[1]-hard.
- If D is a product ring (e.g. $\mathbb{Z}_6 = \mathbb{Z}_2 \times \mathbb{Z}_3$), then even Min-2-Lin(D) is W[1]-hard.

*A *Euclidean domain* is an abstract algebraic structure where the Euclidean algorithm works. Examples include all fields, integers $\mathbb{Z}$, Gaussian integers $\mathbb{Z}[i]$, Eisenstein integers $\mathbb{Z}[\omega]$, univariate polynomials over a field $\mathbb{F}[x]$.

FPT Algorithms for Deletion Problems

Problem	Solved in	Technique	FPT-reduces to
Bipartization	[RSV04]	Iterative compression	Min-2-Lin($\mathbb{F}_2$)
q-Multiway Cut	[Marx06]	Important separators	Min-2-Lin($\mathbb{F}_q$)
Multiway Cut	[Marx06] [CPPW13]	Important separators, LP-branching	Min-2-Lin($\mathbb{Q}$)
Multicut	[MR11] [BDT11]	Random sampling of imporant separators, Problem-specific approach	Min-2-Lin($\mathbb{Z}$)

FPT Algorithms for Deletion Problems

Problem	Solved in	Technique	FPT-reduces to
Bipartization	[RSV04]	Iterative compression	Min-2-Lin($\mathbb{F}_2$)
q-Multiway Cut	[Marx06]	Important separators	Min-2-Lin($\mathbb{F}_q$)
Multiway Cut	[Marx06] [CPPW13]	Important separators, LP-branching	Min-2-Lin($\mathbb{Q}$)
Multicut	[MR11] [BDT11]	Random sampling of imporant separators, Problem-specific approach	Min-2-Lin($\mathbb{Z}$)

Bipartization

Input: a graph G and an integer k.

Goal: delete k edges to make G bipartite

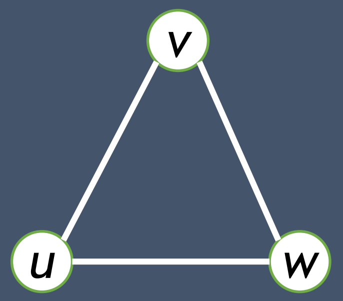
Bipartization

Input: a graph G and an integer k.

Goal: delete k edges to make G bipartite

Reduction to Min-2-Lin($\mathbb{F}_2$):

For every edge uv in G, add equation $u + v = 1 \mod 2$.



$u = 1$
$v = 0$
$w = ???$

Bipartization

Input: a graph G and an integer k.

Goal: delete k edges to make G bipartite

Reduction to Min-2-Lin($\mathbb{F}_2$):

For every edge uv in G, add equation $u + v = 1 \mod 2$.

Iterative compression allows to assume that at every step the algorithm has access to a solution of size k + 1.

Multiway Cut

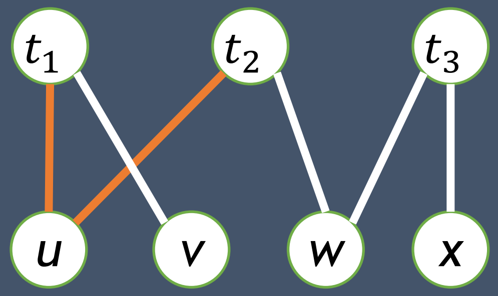
Input: a graph G, an integer k, terminal vertices $t_1, \ldots, t_m$.

Goal: delete k edges to separate all terminals in G.

Multiway Cut

Input: a graph G, an integer k, terminal vertices $t_1, \dots, t_m$.
Goal: delete k edges to separate all terminals in G.

Reduction to Min-2-Lin($\mathbb{Q}$):
Add equations $t_i = i$ for terminals and $u = v$ for edges uv in G.



$t_1 = 1$ $\quad t_2 = 2$ $\quad 1 = 2$

$t_1 = u$ $\quad u = t_2$

Multiway Cut

Input: a graph G, an integer k, terminal vertices $t_1, \dots, t_m$.
Goal: delete k edges to separate all terminals in G.

Reduction to Min-2-Lin($\mathbb{Q}$):
Add equations $t_i = i$ for terminals and $u = v$ for edges uv in G.

Important separators: while $\#st$-cuts of size k is unbounded,
∃ 4^k important cuts maximizing reach of s.

Multiway Cut

Input: a graph G, an integer k, terminal vertices $t_1, \dots, t_m$.
Goal: delete k edges to separate all terminals in G.

Reduction to Min-2-Lin($\mathbb{Q}$):
Add equations $t_i = i$ for terminals and $u = v$ for edges uv in G.

LP-branching: LP-relaxation of Multiway Cut admits ½-integral optima & is persistant, branch on ½-integral values.

Multicut

Input: a graph G, an integer k, terminal pairs $(s_1, t_1), \ldots, (s_m, t_m)$.
Goal: delete k edges to separate terminal pairs in G.

Multicut

Input: a graph G, an integer k, terminal pairs $(s_1, t_1), \ldots, (s_m, t_m)$.
Goal: delete k edges to separate terminal pairs in G.

Reduction to Min-2-Lin($\mathbb{Z}$)
For (s_i, t_i), select i^{th} prime π_i and add equations
$s_i = \pi_i s'_i$ and $t_i = \pi_i t'_i + 1$, and $u = v$ for all edges uv in G.

Multicut

Input: a graph G, an integer k, terminal pairs $(s_1, t_1), \dots, (s_m, t_m)$.
Goal: delete k edges to separate terminal pairs in G.

Reduction to Min-2-Lin($\mathbb{Z}$)
For (s_i, t_i), select i^{th} prime π_i and add equations
$s_i = \pi_i s'_i$ and $t_i = \pi_i t'_i + 1$, and $u = v$ for all edges uv in G.

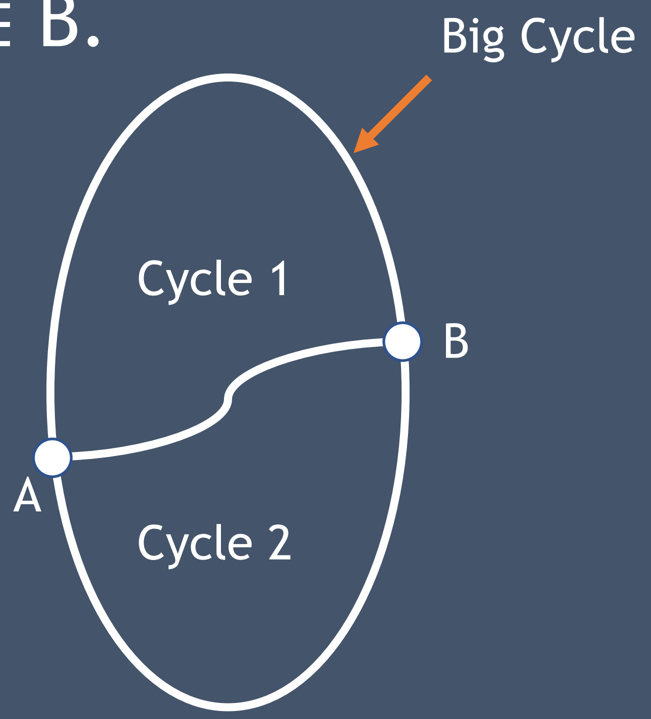
$\Longrightarrow$ Equations imply that $s_i \equiv 0 \bmod \pi_i$ and $t_i \equiv 1 \bmod \pi_i$, so the solution must break every (s_i, t_i)-path.
$\Longleftarrow$ If no (s_i, t_i)-path remains, apply CRT in each component.

Biased Graphs and Important Balanced Subgraphs

Biased Graphs

G – graph, B – *balanced family* of cycles, i.e.

Cycle 1 ∈ B **and** Cycle 2 ∈ B ⇒ Big Cycle ∈ B.

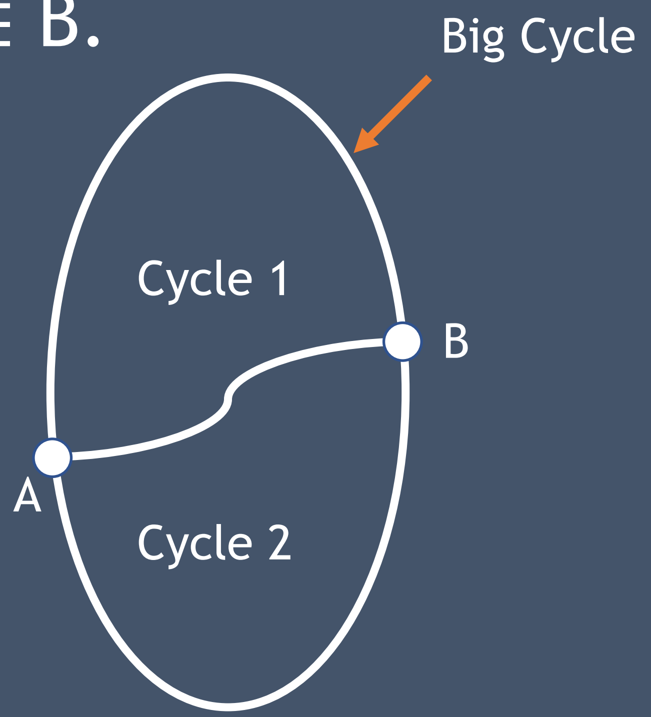


Biased Graphs

G – graph, B – *balanced family* of cycles, i.e.

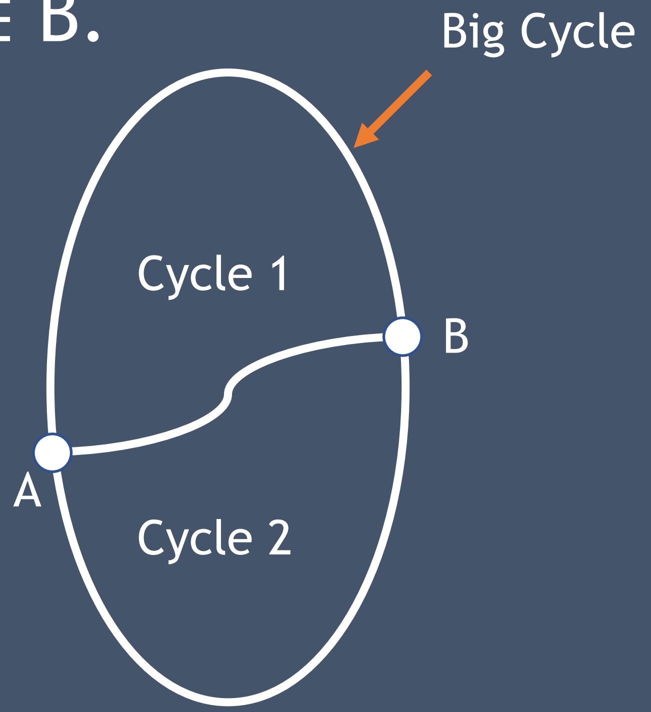
Cycle 1 ∈ B **and** Cycle 2 ∈ B ⇒ Big Cycle ∈ B.

Big Cycle ∉ B ⇒ Cycle 1 **or** Cycle 2 ∉ B .



Biased Graphs

G – graph, B – *balanced family* of cycles, i.e.
Cycle 1 ∈ B **and** Cycle 2 ∈ B ⇒ Big Cycle ∈ B.
Big Cycle ∉ B ⇒ Cycle 1 **or** Cycle 2 ∉ B .



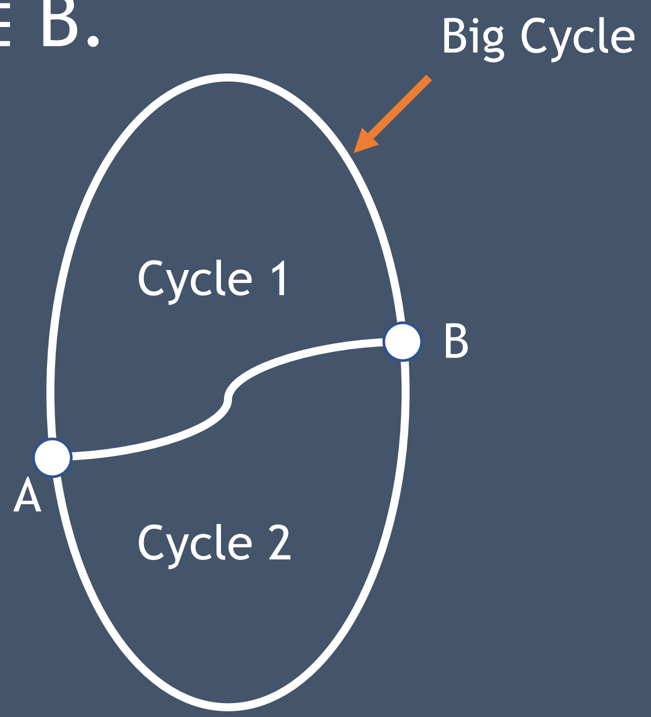
Example 1: B = no cycles.

Biased Graphs

G – graph, B – *balanced family* of cycles, i.e.

Cycle 1 ∈ B **and** Cycle 2 ∈ B ⇒ Big Cycle ∈ B.

Big Cycle ∉ B ⇒ Cycle 1 **or** Cycle 2 ∉ B .

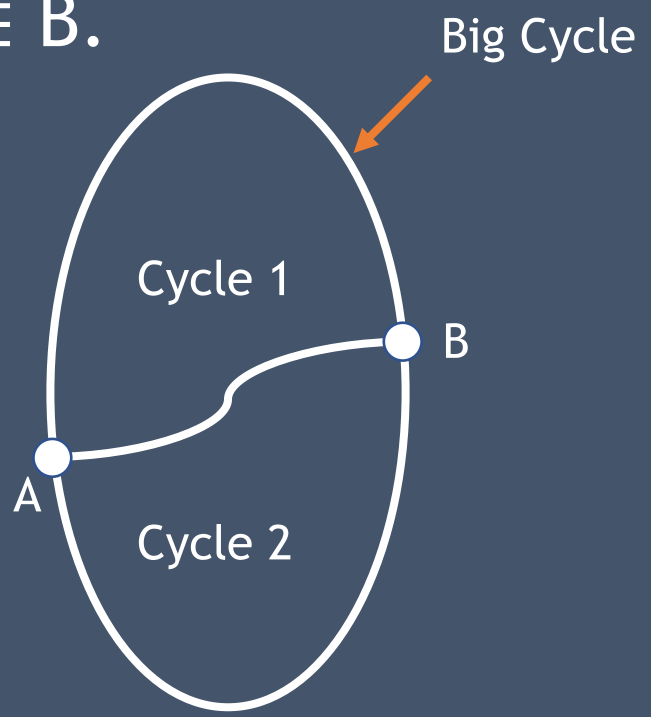


Example 2: B = even cycles.

(Large odd cycle + chord, then
at least one smaller cycle is odd).

Biased Graphs

G – graph, B – *balanced family* of cycles, i.e.
Cycle 1 ∈ B **and** Cycle 2 ∈ B ⇒ Big Cycle ∈ B.
Big Cycle ∉ B ⇒ Cycle 1 **or** Cycle 2 ∉ B .



Example 3: B = cycles avoiding vertex s.

(Large cycle contains s, then
at least one smaller cycle contains).

Biased Graph Cleaning

Input: a biased graph (G,B) and an integer k.

Goal: delete k edges to make G balanced.

Balanced Cycles	Resulting Problem
No cycles	Feedback Edge Set
Even cycles	Bipartization
Cycles avoiding s	Multiway Cut

Biased Graph Cleaning

Input: a biased graph (G,B) and an integer k.

Goal: delete k edges to make G balanced.

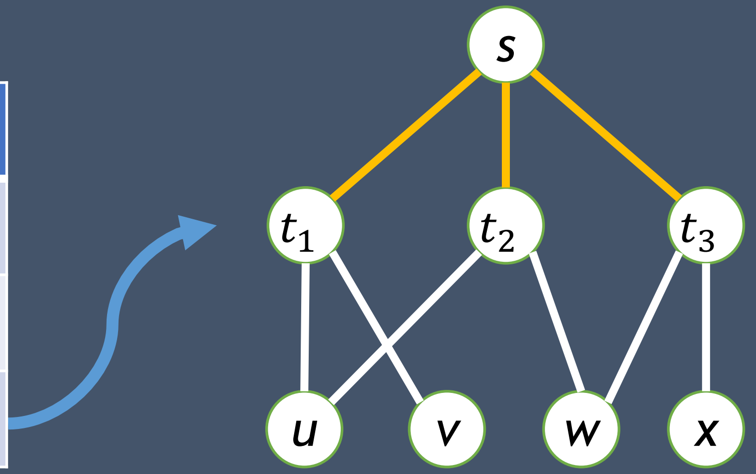
Unbalanced Cycles	**Resulting Problem**
All cycles	Feedback Edge Set
Odd cycles	Bipartization
Cycles through s	Multiway Cut

Biased Graph Cleaning

Input: a biased graph (G,B) and an integer k.

Goal: delete k edges to make G balanced.

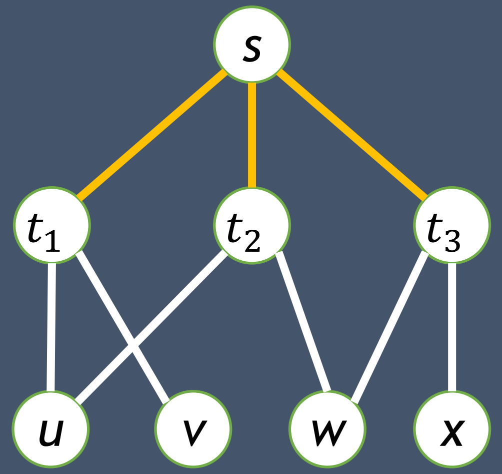
Unbalanced Cycles	Resulting Problem
All cycles	Feedback Edge Set
Odd cycles	Bipartization
Cycles through s	Multiway Cut



Rooted Biased Graph Cleaning

Input: a biased graph (G,B), a root vertex s and an integer k.

Goal: delete k edges to make component of s in G balanced.

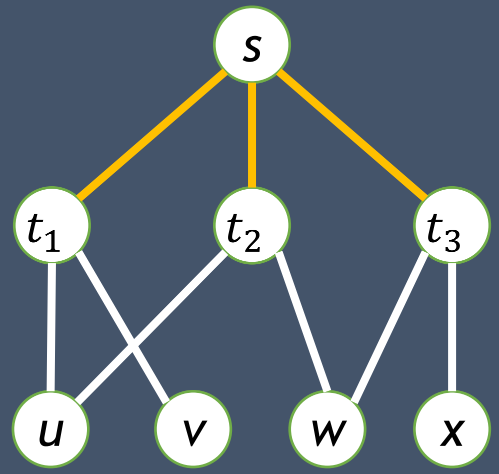


Rooted Biased Graph Cleaning

Input: a biased graph (G,B), a root vertex s and an integer k.
Goal: delete k edges to make component of s in G balanced.

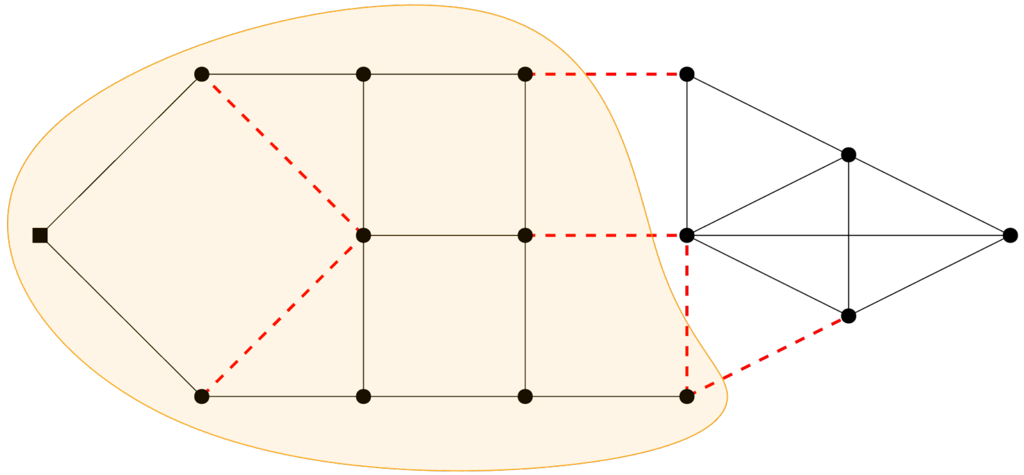
[Wahlström17] showed $O^*(2^k)$ algorithm based on a ½-integral LP branching.

The result can be used for unrooted BGC and yields a $O^*(4^k)$ time algorithm.



Important Balanced Subgraphs

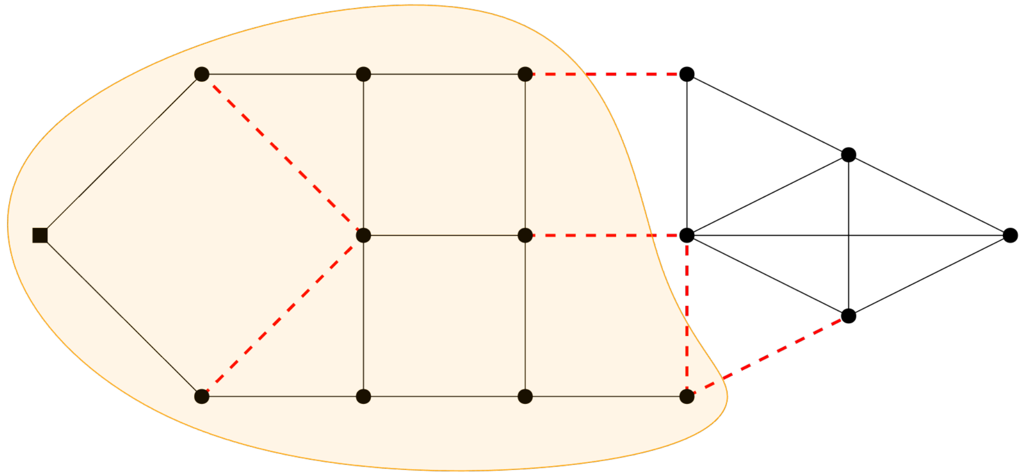
Generalization of important separators.



Important Balanced Subgraphs

Generalization of important separators.

For a subgraph H of G, let cost(H) = cost of carving H out of G.



Important Balanced Subgraphs

Generalization of important separators.

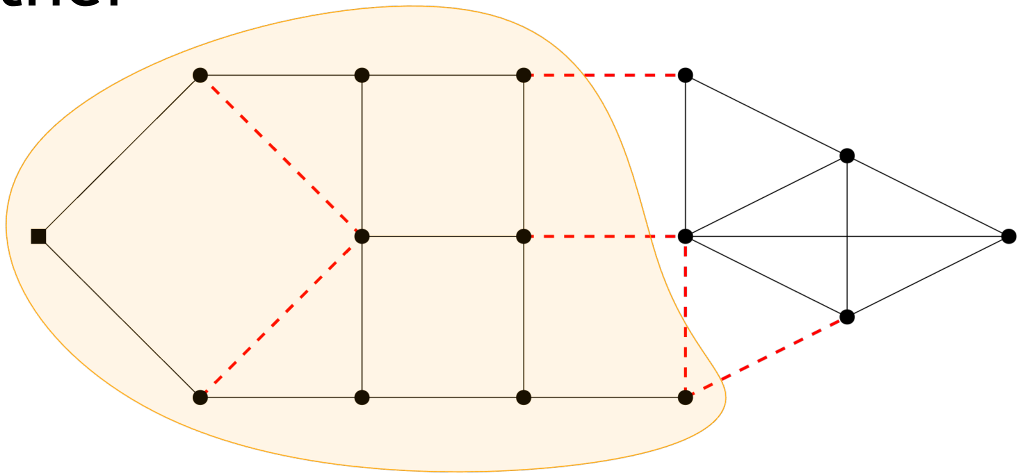
For a subgraph H of G, let cost(H) = cost of carving H out of G.

Consider two balanced subgraphs H_1 and H_2 containing root s.

Subgraph H_1 *dominates* H_2 if either

$\mathrm{cost}(H_1) < \mathrm{cost}(H_2)$ or

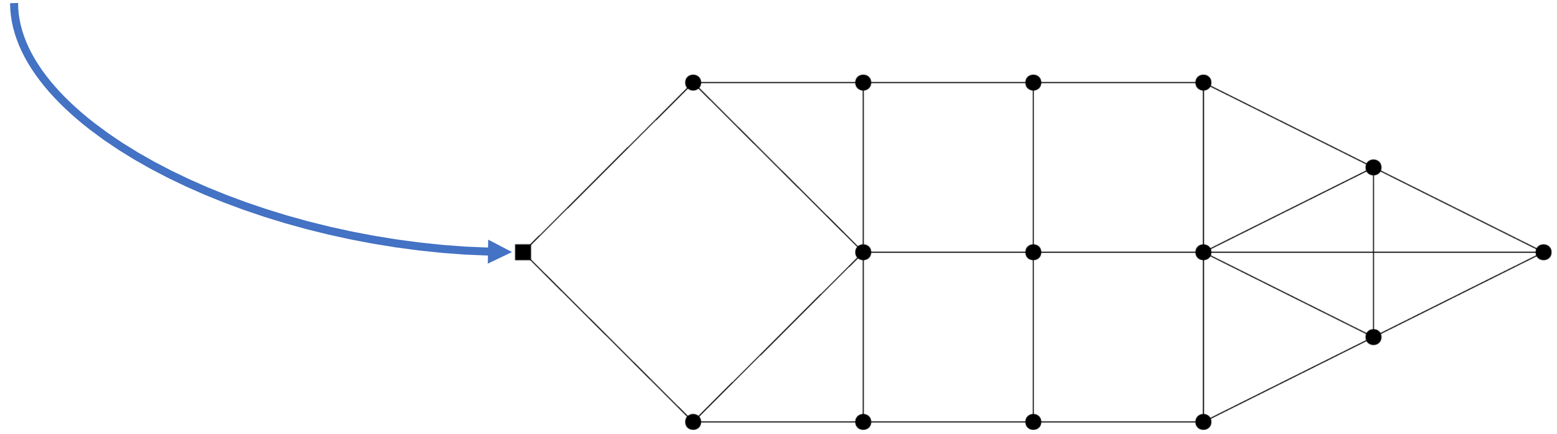
$V(H_1) \subsetneq V(H_2)$.



Important = undominated.

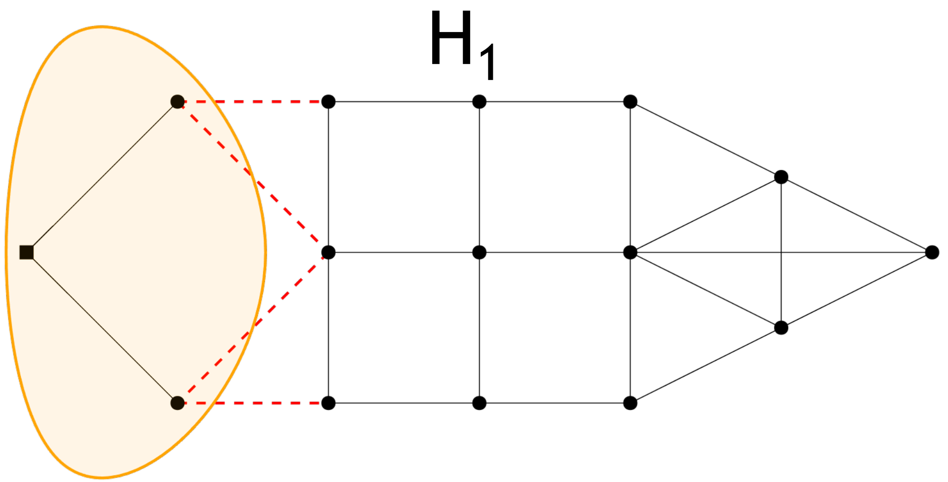
Important Balanced Subgraphs

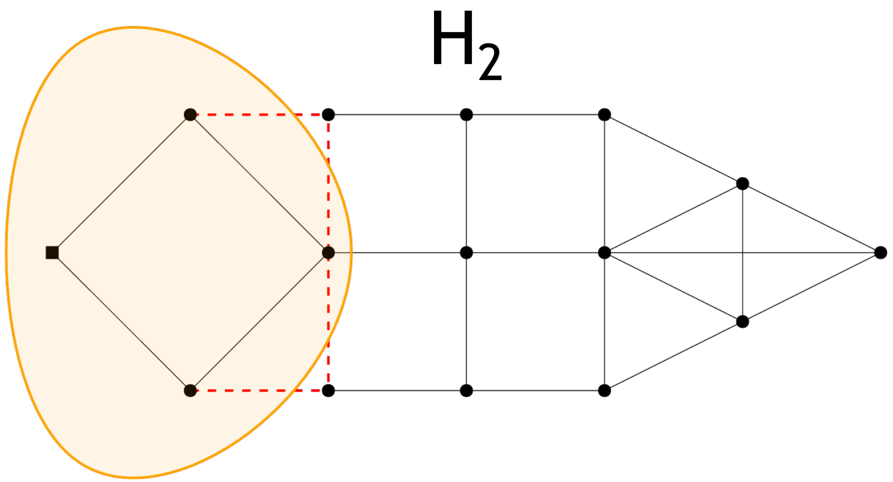
Example: balanced cycles = even cycles,
root on the left.



Important Balanced Subgraphs

Balanced (=bipartite) subgraphs of cost 4.

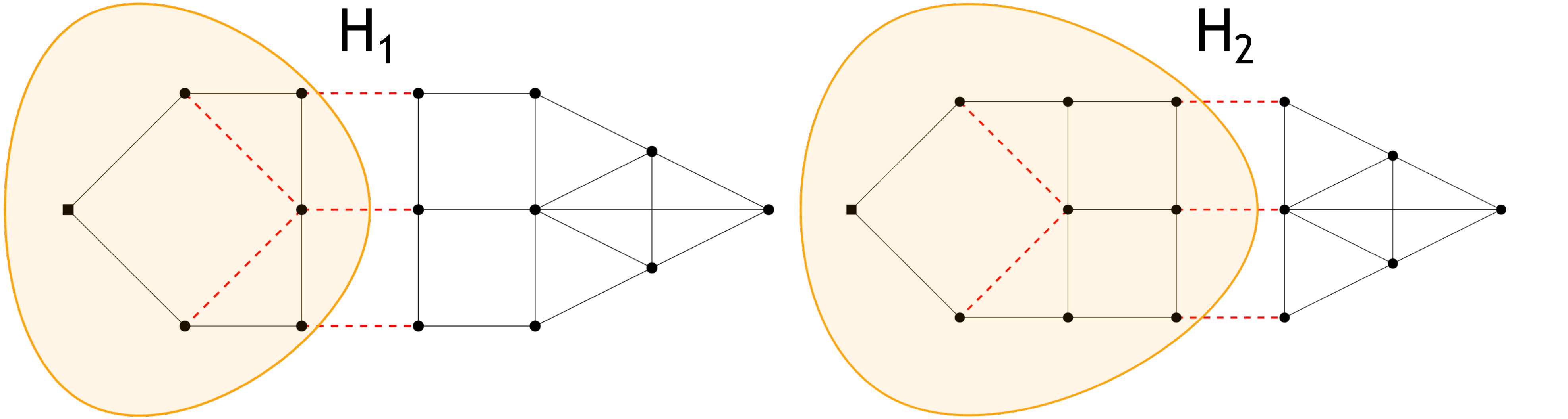




H_2 dominates H_1 since $V(H_1) \subsetneq V(H_2)$ while $cost(H_1) = cost(H_2)$.

Important Balanced Subgraphs

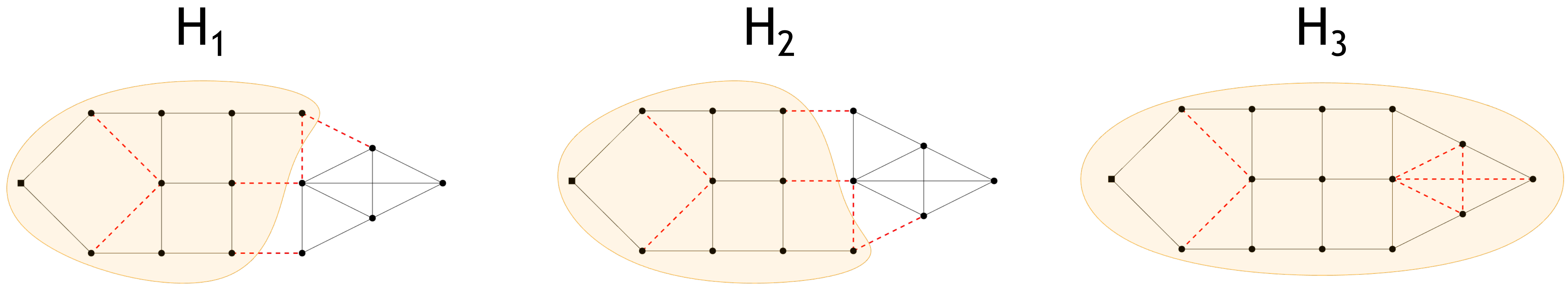
Balanced (=bipartite) subgraphs of cost 5.



H_2 dominates H_1 since $V(H_1) \subsetneq V(H_2)$ while $\mathrm{cost}(H_1) = \mathrm{cost}(H_2)$.

Important Balanced Subgraphs

Balanced (=bipartite) subgraphs of cost 6.

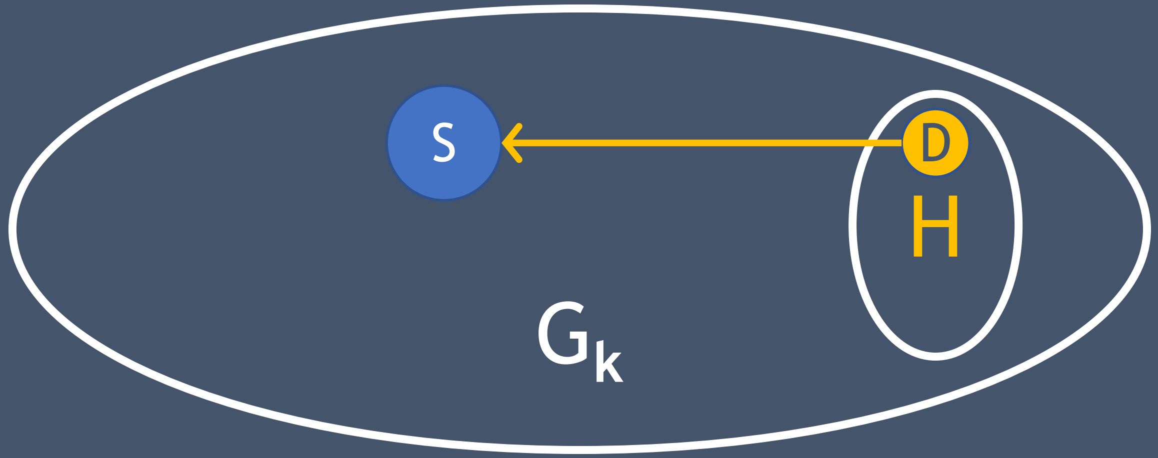


H_1 and H_2 are incomparable, H_3 dominates both.

Important Balanced Subgraphs

Theorem: Let G_k contain balanced rooted subgraphs of cost ≤ k.
There is a family H ⊂ G_k of 4^k (important) balanced subgraphs such that for every S in G_k there is D in H that dominates S.
Moreover, such H can be computed in $O^*(4^k)$ time.

G_k – balanced rooted cycles
H – important cycles



Applications

Immediate fpt algorithms:

- Bipartization,
- Subset Feedback Edge Set,
- Group Feedback Edge Set.

With more work:

- Min-2-Lin(D) for any Euclidean domain D (including $\mathbb{Q}$, $\mathbb{Z}$).

THANK YOU!