Algorithm Engineering for Optimal Graph Bipartization

Falk Hüffner

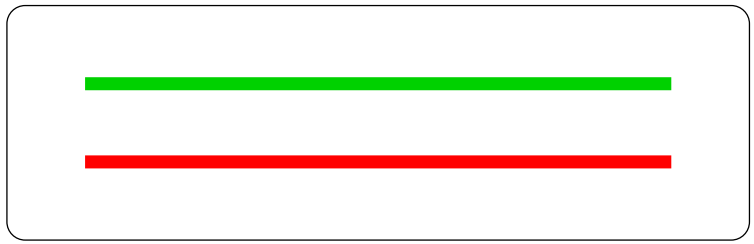
Institut für Informatik
Friedrich-Schiller-Universität Jena

4th International Workshop on Efficient and Experimental Algorithms

Outline

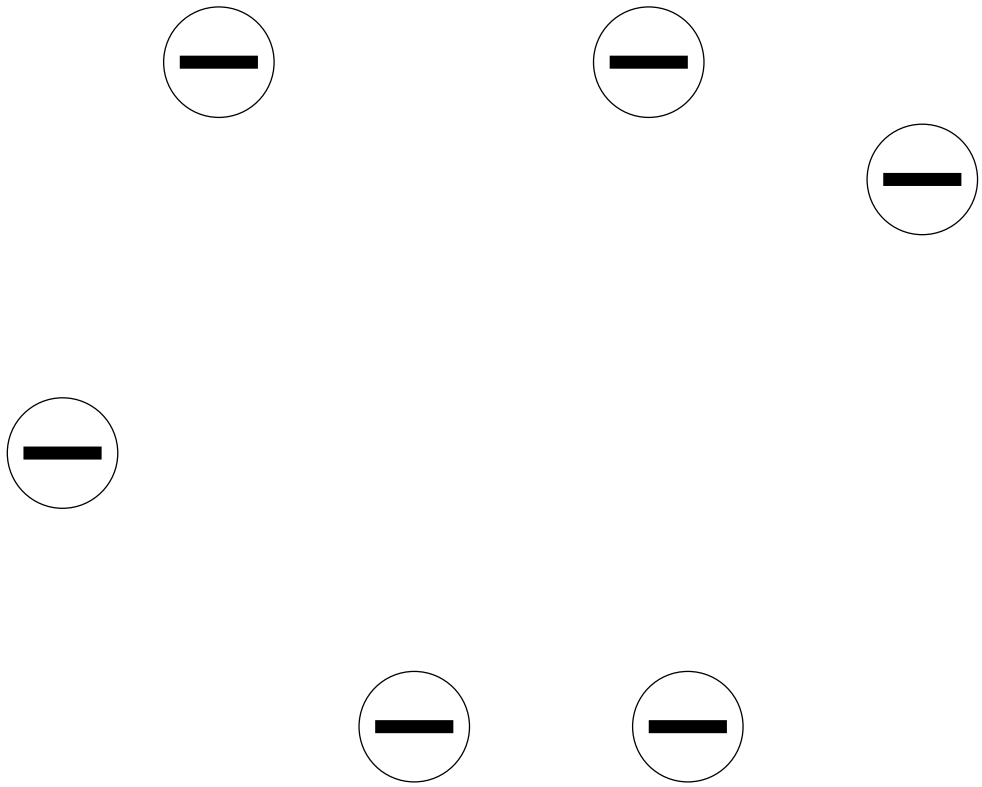
DNA Sequence Assembly

Diploid cells have two copies of each chromosome



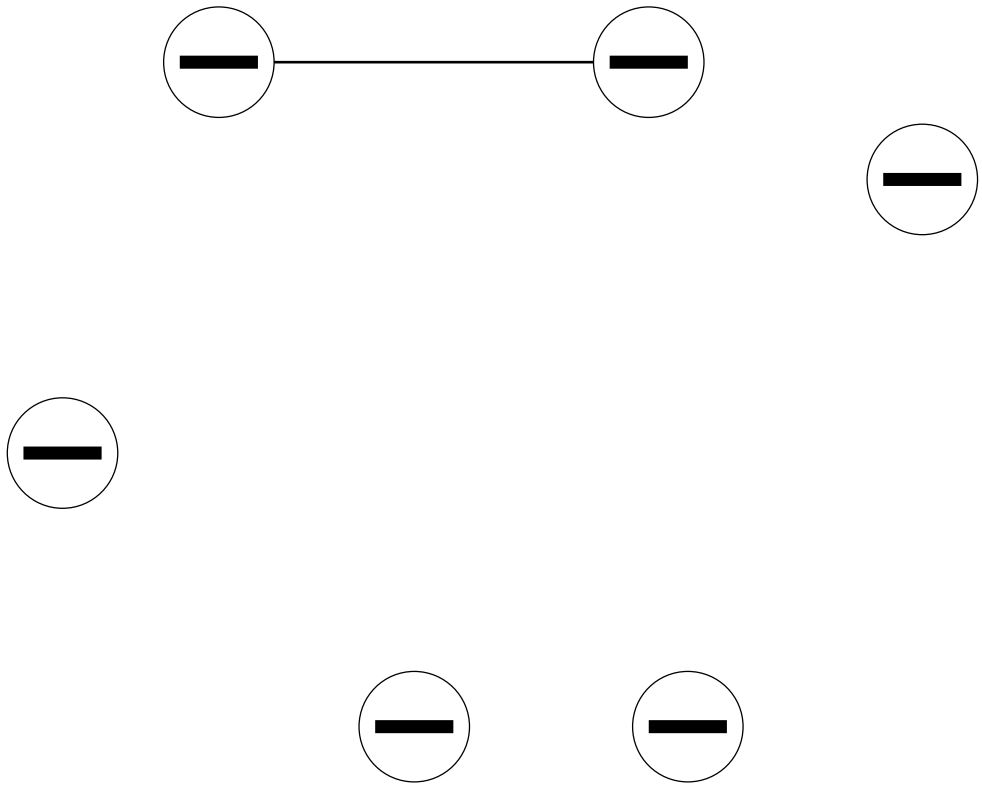
DNA Sequence Assembly

Chromosome assignments of the fragments in shotgun assembly are initially unknown



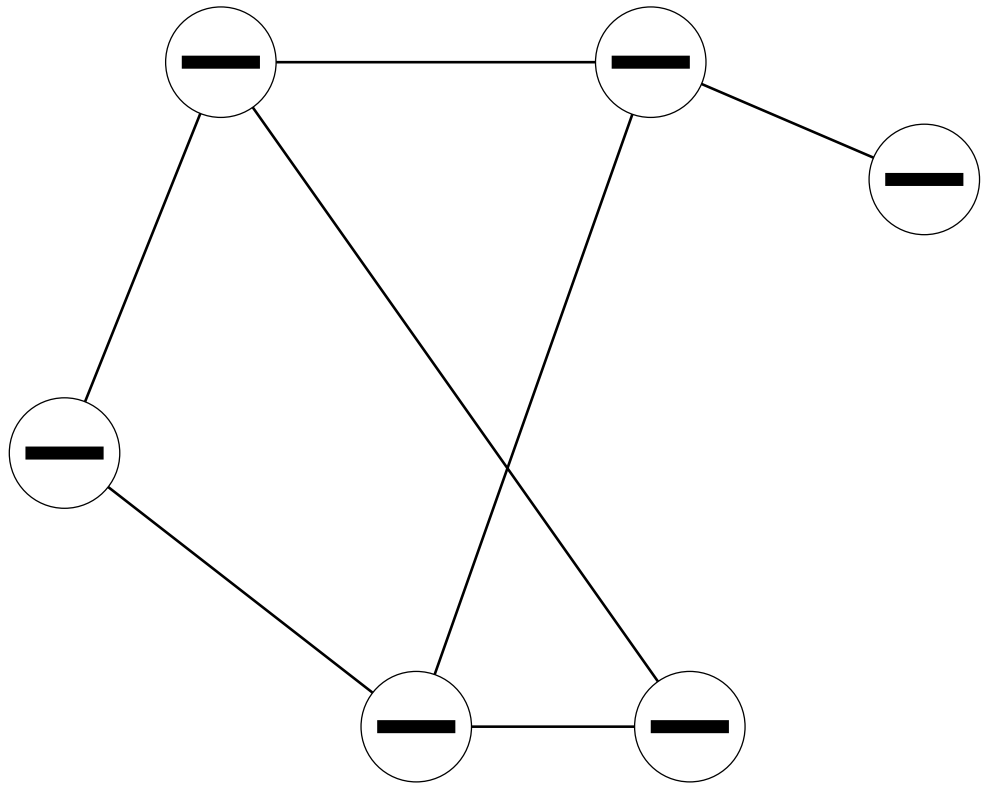
DNA Sequence Assembly

Pairwise conflicts indicate that two fragments are from different copies



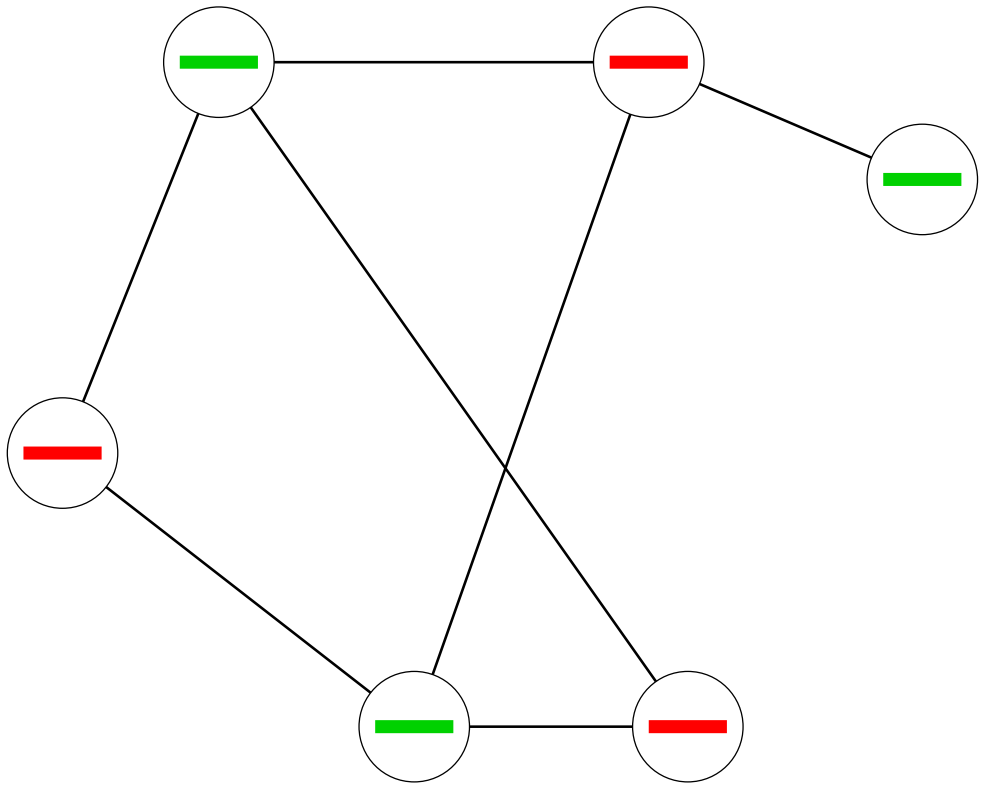
DNA Sequence Assembly

Pairwise conflicts indicate that two fragments are from different copies



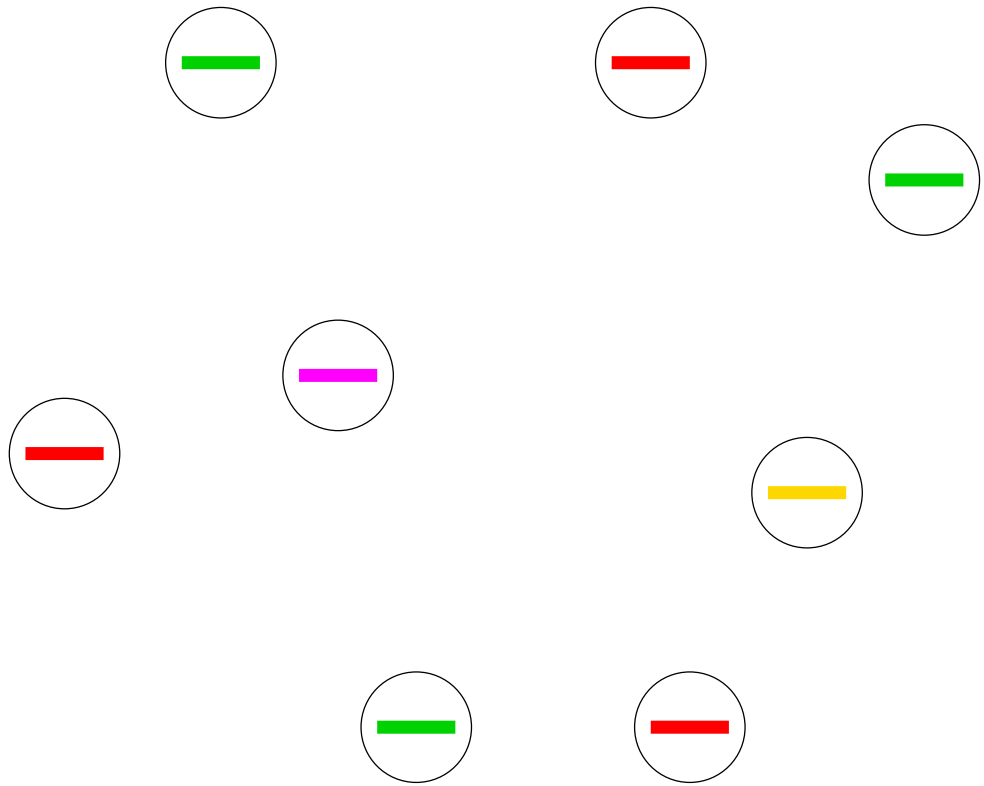
DNA Sequence Assembly

Reconstruction of chromosome assignment from the bipartite conflict graph



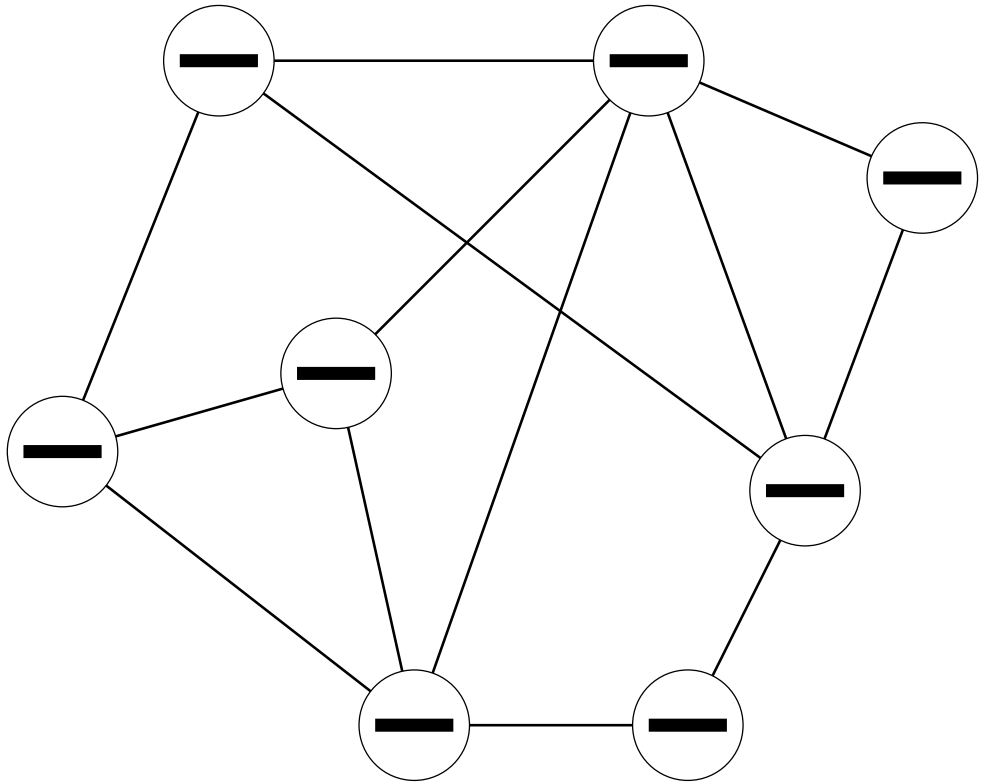
Minimum Fragment Removal

In practise, contaminations occur.



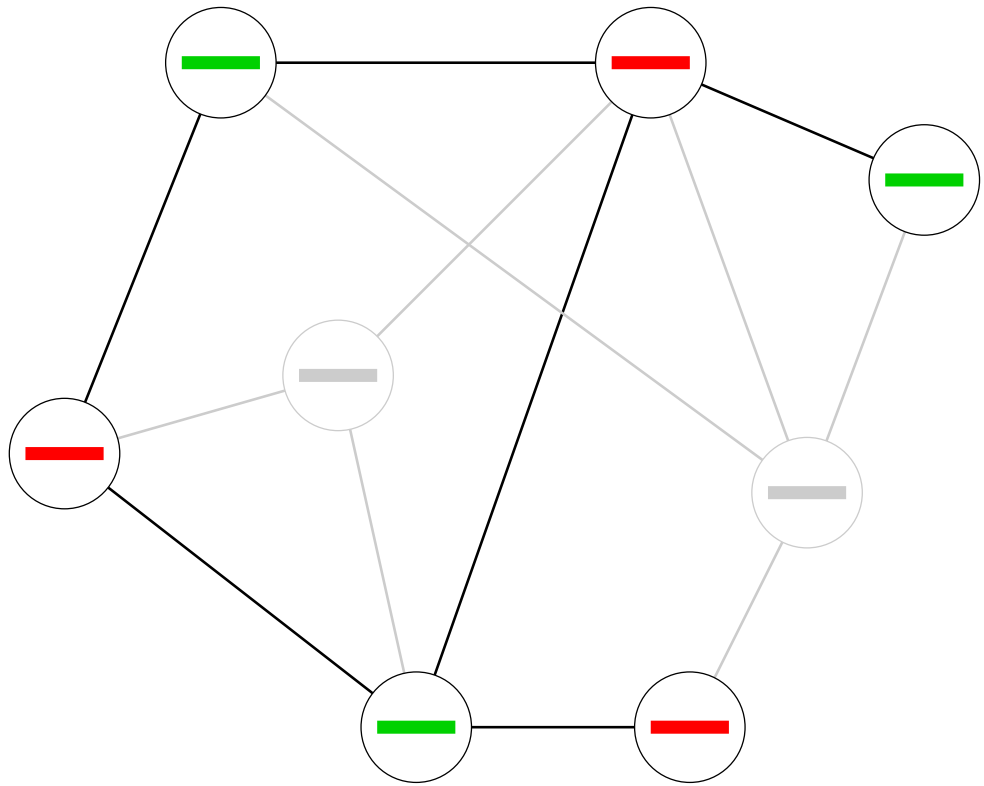
Minimum Fragment Removal

Contamination fragments will conflict with fragments from both copies.



Minimum Fragment Removal

The task is to recognize contamination fragments.



Formalization as Graph Bipartization

Graph Bipartization

Input: *An undirected graph* $G = (V, E)$ *and a nonnegative integer* k*.*

Task: *Find a subset* $C \subseteq V$ *of vertices with* $|C| = k$ *such that* $G[V \setminus C]$ *is bipartite.*

Formalization as Graph Bipartization

Graph Bipartization
Input: *An undirected graph $G = (V, E)$ and a nonnegative integer k.*
Task: *Find a subset $C \subseteq V$ of vertices with $|C| = k$ such that $G[V \setminus C]$ is bipartite.*

Equivalent formulation:

Odd Cycle Cover
Task: *Find a subset $C \subseteq V$ of vertices with $|C| = k$ such that C touches every odd cycle in G.*

Graph Bipartization

- Graph Bipartization is NP-complete [Lewis and Yannakakis, JCSS 1980]; it has numerous applications, e. g. in VLSI design and register allocation

Graph Bipartization

- Graph Bipartization is NP-complete [Lewis and Yannakakis, JCSS 1980]; it has numerous applications, e. g. in VLSI design and register allocation
- Graph Bipartization is MaxSNP-hard [Papadimitriou and Yannakakis, JCSS 1991]. The best known polynomial-time approximation is by a factor of $\log |V|$
  [Garg, Vazirani, and Yannakakis, SIAM J. Comput. 1996]

Parameterization

Approach: For Minimum Fragment Removal, $k \ll n$. Try to confine the combinatorial explosion to k

Parameterization

Approach: For Minimum Fragment Removal, $k \ll n$. Try to confine the combinatorial explosion to k

Definition

For some *parameter* k of a problem, the problem is called *fixed-parameter tractable* with respect to k if there is an algorithm that solves it in $f(k) \cdot n^{O(1)}$.

Parameterization

Approach: For Minimum Fragment Removal, $k \ll n$. Try to confine the combinatorial explosion to k

Definition

For some *parameter* k of a problem, the problem is called *fixed-parameter tractable* with respect to k if there is an algorithm that solves it in $f(k) \cdot n^{O(1)}$.

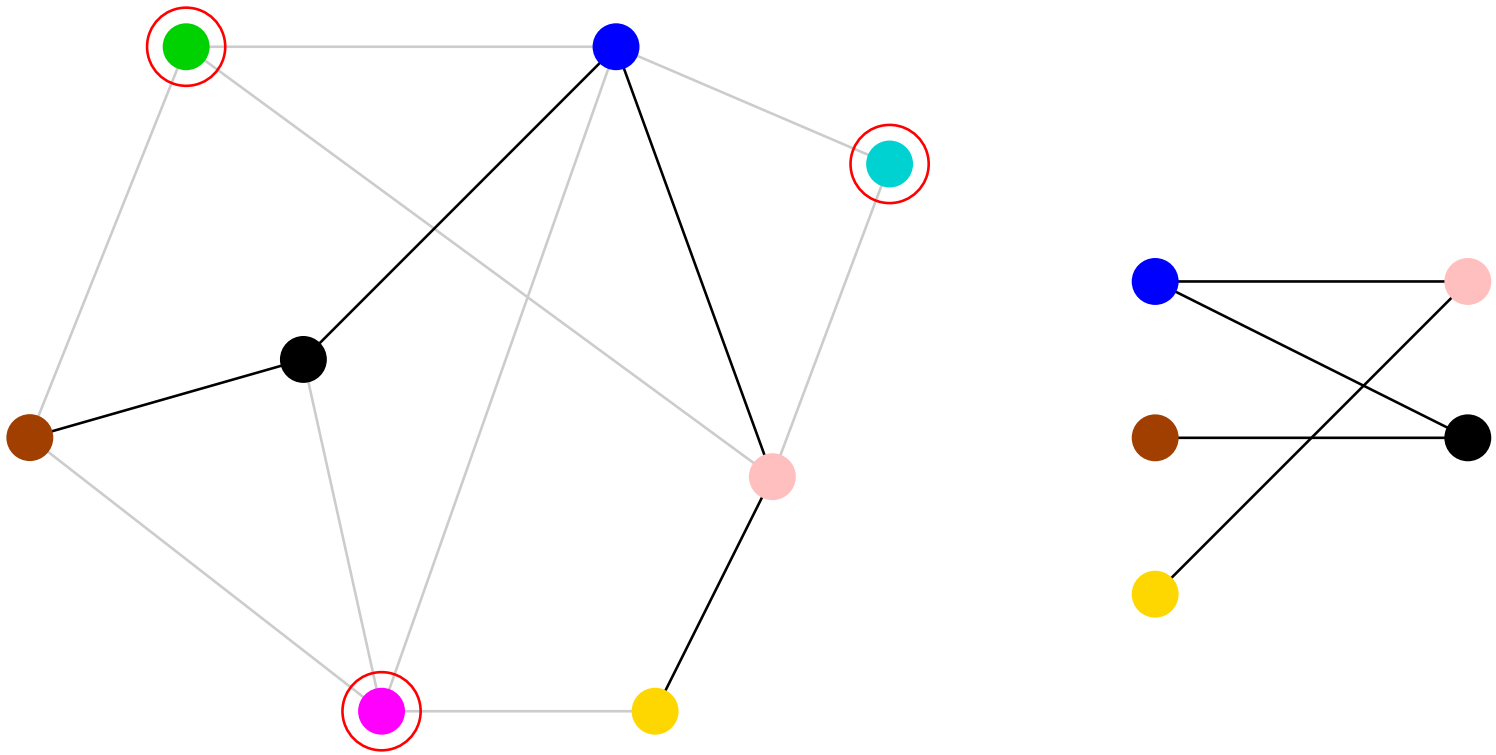
Graph Bipartization is fixed-parameter tractable with respect to k [Reed, Smith&Vetta, Oper. Res. Lett. 2004].

Iterative Compression

Approach: use a *compression routine* iteratively.
Compression routine: Given a size-$(k+1)$ solution, either computes a size-k solution or proves that there is no size-k solution.

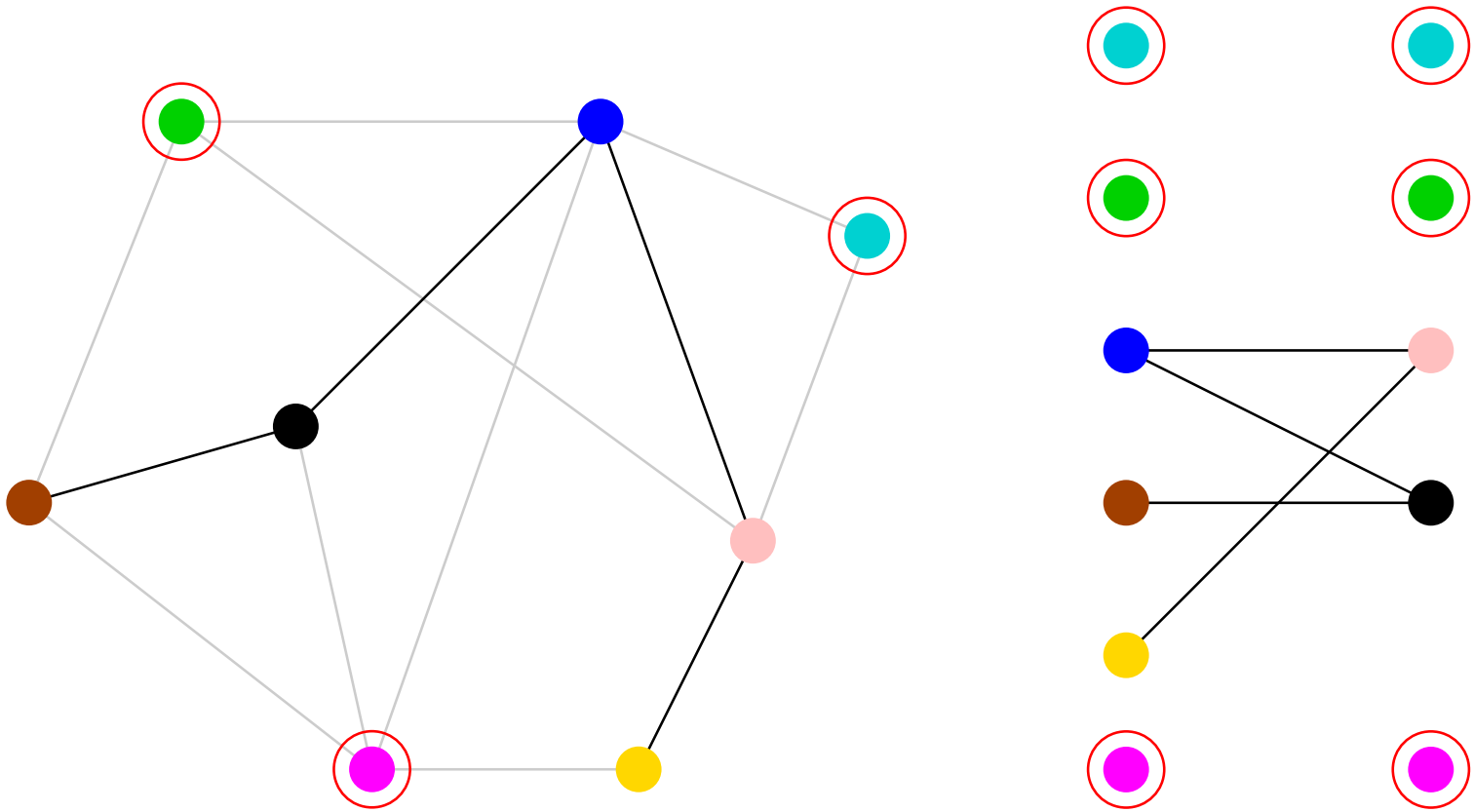
Compression Routine for Graph Bipartization

Idea: Convert the covering problem to a cut problem.



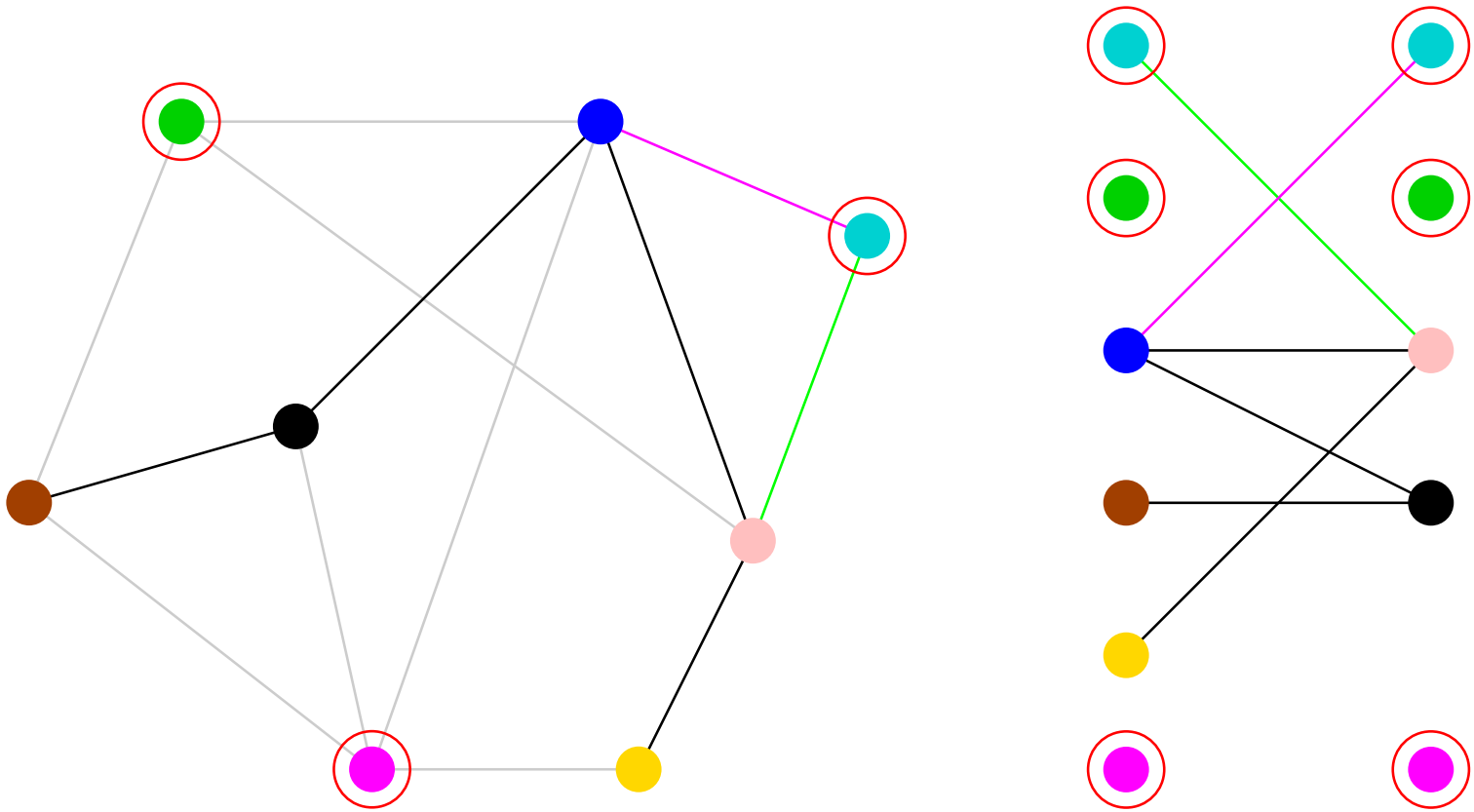
Compression Routine for Graph Bipartization

Idea: Convert the covering problem to a cut problem.



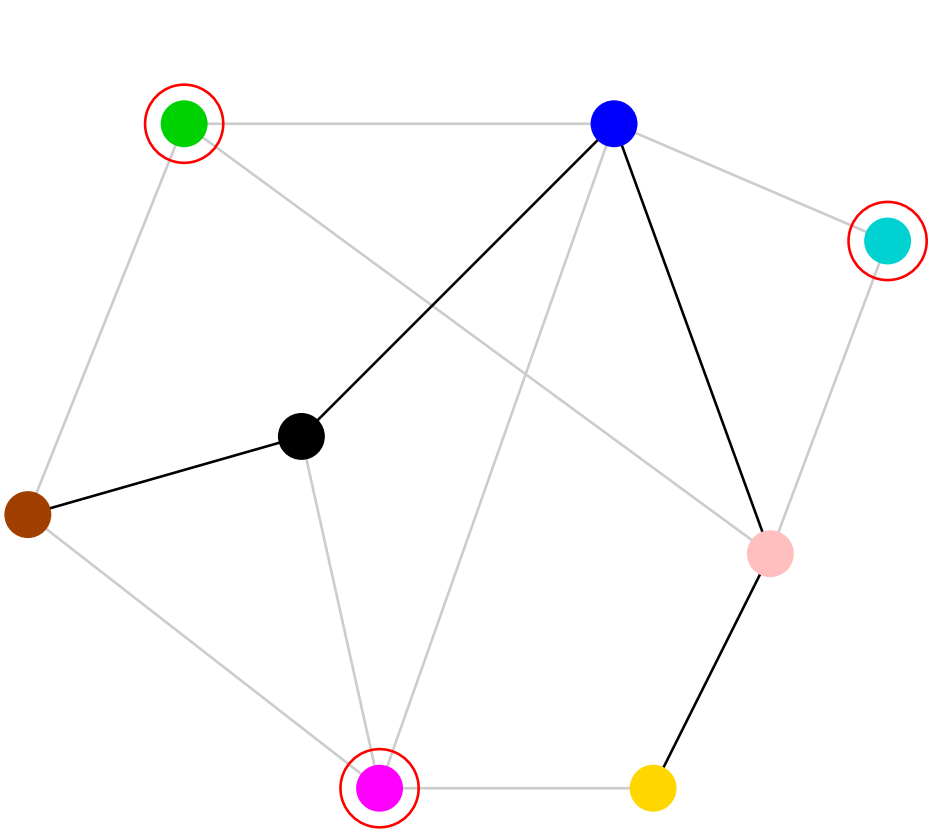
Compression Routine for Graph Bipartization

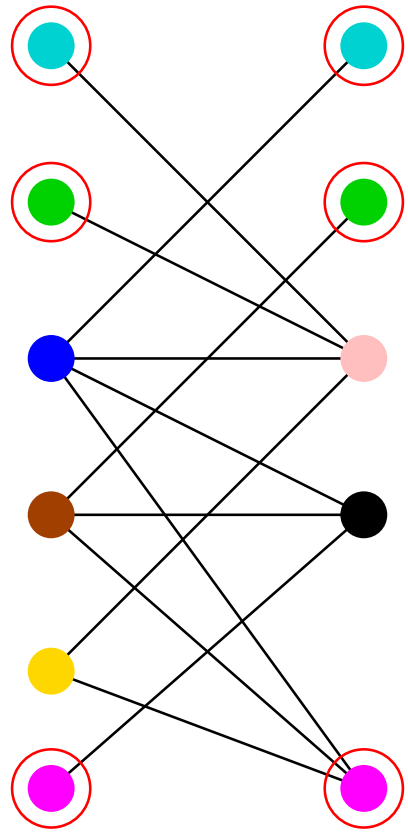
Idea: Convert the covering problem to a cut problem.



Compression Routine for Graph Bipartization

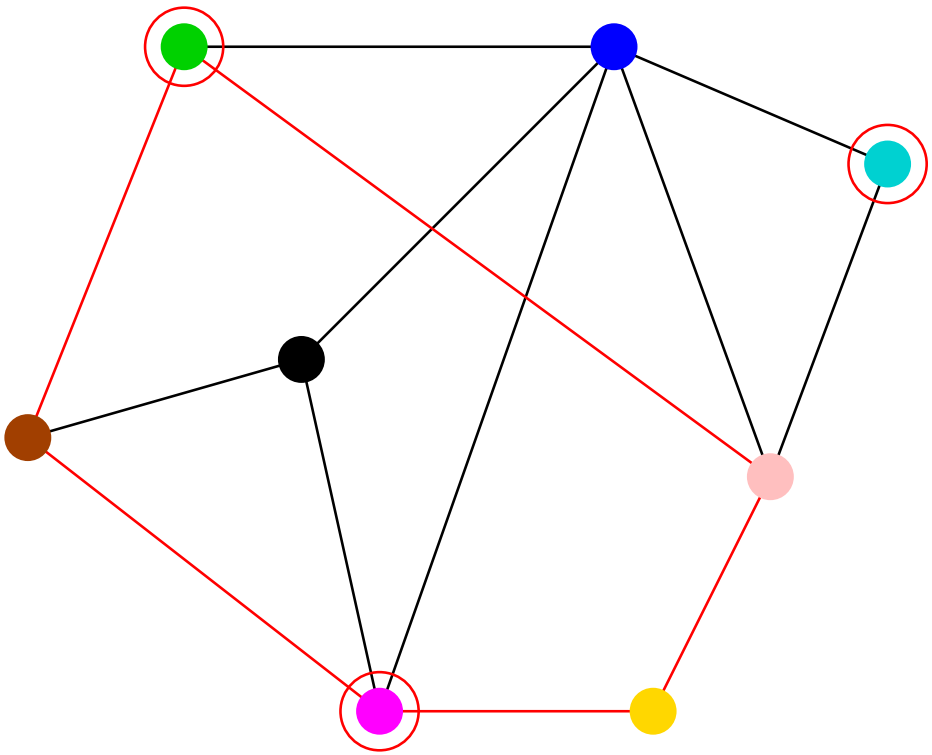
Idea: Convert the covering problem to a cut problem.

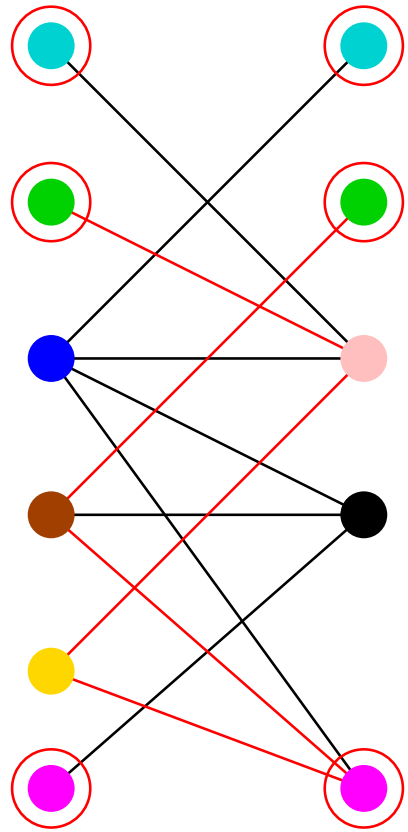




Compression Routine for Graph Bipartization

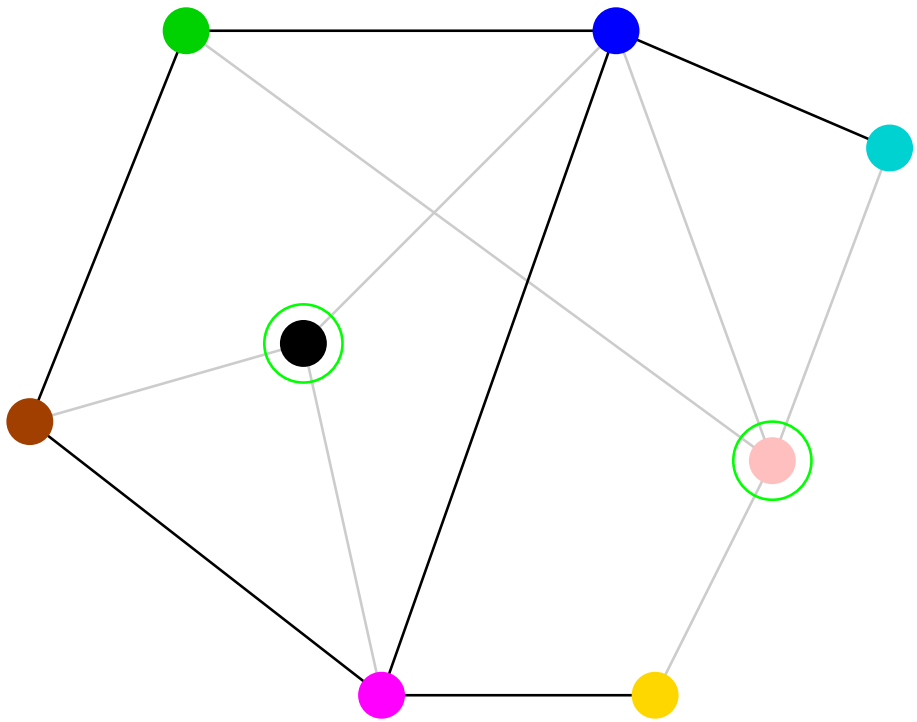
Idea: Convert the covering problem to a cut problem.

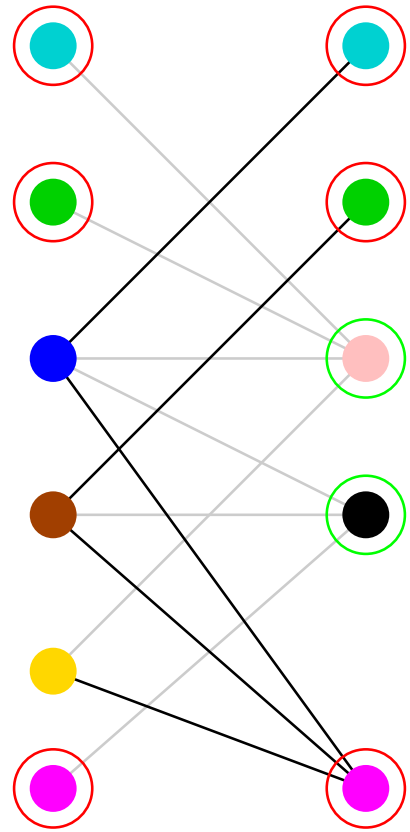




Compression Routine for Graph Bipartization

Idea: Convert the covering problem to a cut problem.





Valid Partitions

But: The resulting multi-cut problem is still NP-complete!

Definition

A *valid partition* divides the ○ vertices into input vertices →○ and output vertices ○→ such that for each pair one is input and one is output.

Valid Partitions

But: The resulting multi-cut problem is still NP-complete!

Definition

A *valid partition* divides the ○ vertices into input vertices →○ and output vertices ○→ such that for each pair one is input and one is output.

A cut between the input vertices and the output vertices of a valid partition provides a smaller bipartization solution.

Valid Partitions

But: The resulting multi-cut problem is still NP-complete!

Definition

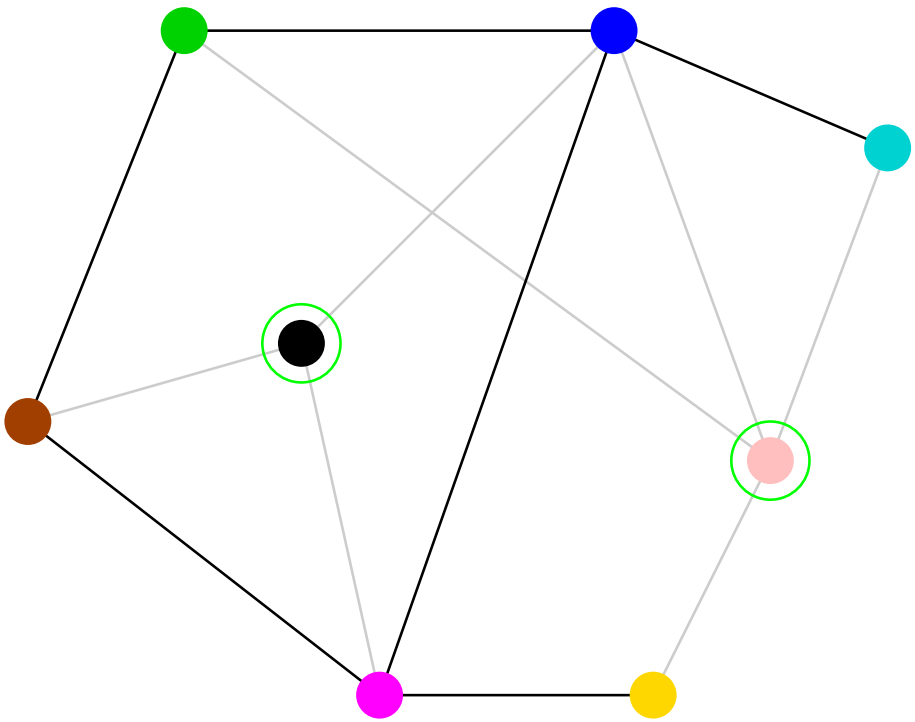
A *valid partition* divides the ○ vertices into input vertices →○ and output vertices ○→ such that for each pair one is input and one is output.

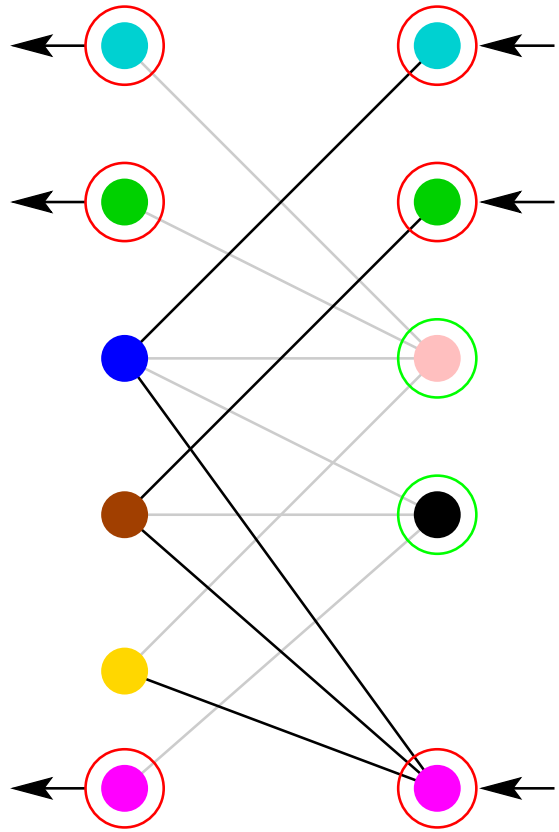
A cut between the input vertices and the output vertices of a valid partition provides a smaller bipartization solution.

Lemma ([Reed, Smith&Vetta 2004])

If there is a smaller bipartization solution, then there is a valid partition such that this solution is a cut between the input vertices and the output vertices.

Valid Partitions





Compression Routine Graph Bipartization

Compression Routine:

- Enumerate all 2^k valid partition
- For each, find a vertex cut in $k \cdot m$ time

Compression Routine Graph Bipartization

Compression Routine:

- Enumerate all 2^k valid partition
- For each, find a vertex cut in $k \cdot m$ time

Theorem

Graph Bipartization *can be solved in* $O(3^k \cdot kmn)$ *time.*

Experimental Results

Run time in seconds for some Minimum Site Removal instances

	n	m	k	ILP	Reed
A31	30	51	2	0.02	0.00
J24	142	387	4	0.97	0.00
A10	69	191	6	2.50	0.00
J18	71	296	9	47.86	0.05
A11	102	307	11	6248.12	0.79
A34	133	451	13		10.13
A22	167	641	16		350.00
A50	113	468	18		3072.82
A45	80	386	20		
A40	136	620	22		
A17	151	633	25		
A28	167	854	27		
A42	236	1110	30		
A41	296	1620	40		

[Data from Wernicke 2003]

Using Gray Codes to enumerate Valid Partitions

- The flow problems for different valid partitions are “similar” in such a way that we can “recycle” the flow networks for each problem

Using Gray Codes to enumerate Valid Partitions

- The flow problems for different valid partitions are “similar” in such a way that we can “recycle” the flow networks for each problem
- Using a Gray code, we can enumerate valid partitions such that adjacent partitions differ in only one element

Using Gray Codes to enumerate Valid Partitions

- The flow problems for different valid partitions are "similar" in such a way that we can "recycle" the flow networks for each problem
- Using a Gray code, we can enumerate valid partitions such that adjacent partitions differ in only one element
- Only $O(m)$ time, as opposed to $O(km)$ time for solving a flow problem from scratch

Using Gray Codes to enumerate Valid Partitions

- The flow problems for different valid partitions are "similar" in such a way that we can "recycle" the flow networks for each problem
- Using a Gray code, we can enumerate valid partitions such that adjacent partitions differ in only one element
- Only $O(m)$ time, as opposed to $O(km)$ time for solving a flow problem from scratch
- Worst-case speedup by a factor of k

Experimental Results

Run time in seconds for some Minimum Site Removal instances

	n	m	k	ILP	Reed	Gray
A31	30	51	2	0.02	0.00	0.00
J24	142	387	4	0.97	0.00	0.00
A10	69	191	6	2.50	0.00	0.00
J18	71	296	9	47.86	0.05	0.01
A11	102	307	11	6248.12	0.79	0.14
A34	133	451	13		10.13	1.04
A22	167	641	16		350.00	64.88
A50	113	468	18		3072.82	270.60
A45	80	386	20			2716.87
A40	136	620	22			
A17	151	633	25			
A28	167	854	27			
A42	236	1110	30			
A41	296	1620	40			

[Data from Wernicke 2003]

A Heuristic for Dense Graphs

- By examining the subgraph induced by the known odd cycle cover, we can omit many valid partitions from consideration

A Heuristic for Dense Graphs

- By examining the subgraph induced by the known odd cycle cover, we can omit many valid partitions from consideration
- No worst-case speedup for general graphs, but very effective in practice

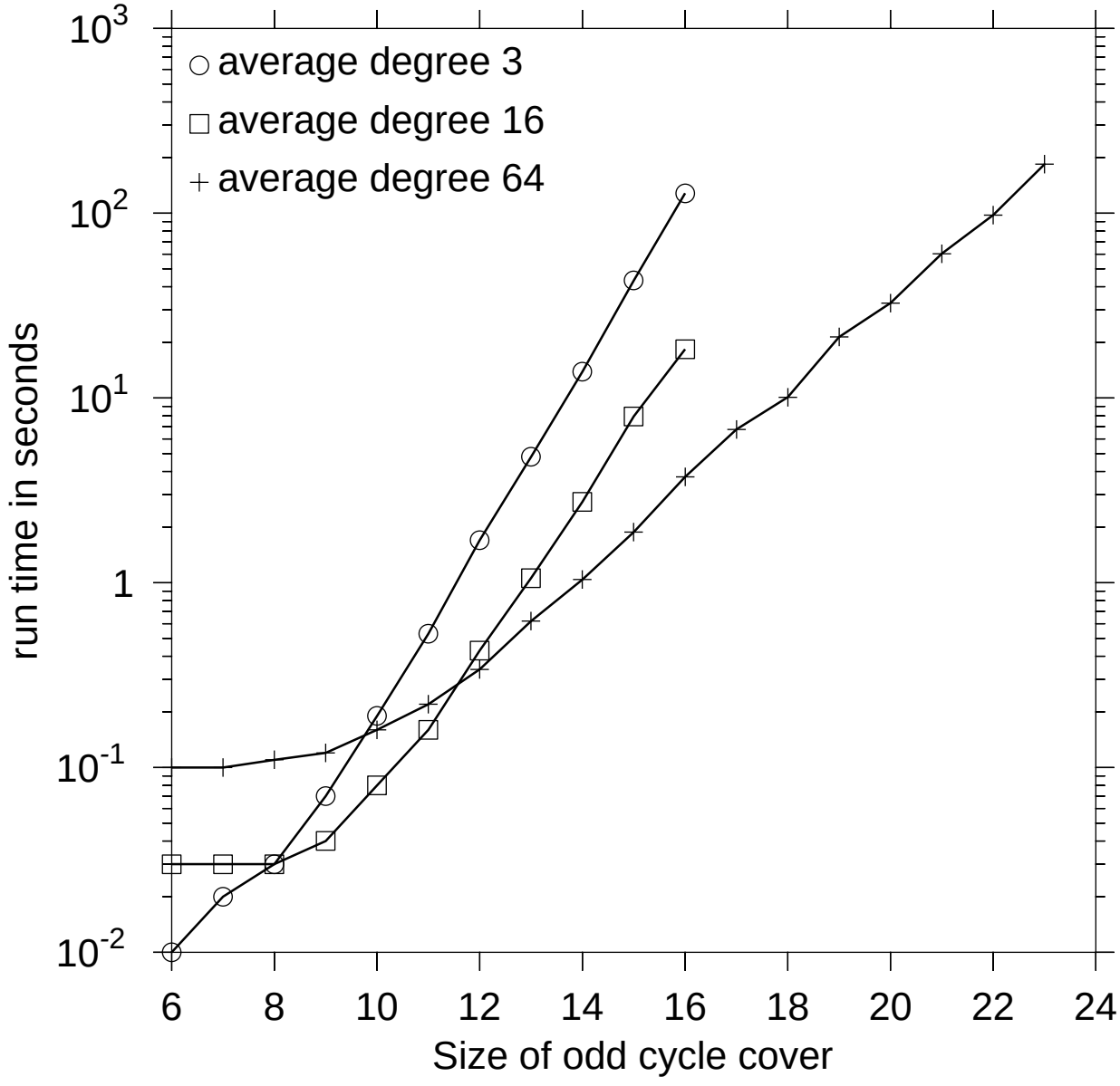
Experimental Results

Run time in seconds for some Minimum Site Removal instances

	n	m	k	ILP	Reed	Gray	Enum2Col
A31	30	51	2	0.02	0.00	0.00	0.00
J24	142	387	4	0.97	0.00	0.00	0.00
A10	69	191	6	2.50	0.00	0.00	0.00
J18	71	296	9	47.86	0.05	0.01	0.00
A11	102	307	11	6248.12	0.79	0.14	0.00
A34	133	451	13		10.13	1.04	0.04
A22	167	641	16		350.00	64.88	0.08
A50	113	468	18		3072.82	270.60	0.05
A45	80	386	20			2716.87	0.14
A40	136	620	22				0.80
A17	151	633	25				5.68
A28	167	854	27				1.02
A42	236	1110	30				73.55
A41	296	1620	40				236.26

[Data from Wernicke 2003]

Heuristic on Random Graphs



$n = 300$

Conclusions

- Iterative compression is a superior method for solving Graph Bipartization in practice
- This makes the practical evaluation of iterative compression for other applications (such as Feedback Vertex Set) appealing

Conclusions

- Iterative compression is a superior method for solving Graph Bipartization in practice
- This makes the practical evaluation of iterative compression for other applications (such as Feedback Vertex Set) appealing

Future work and open questions:

- Combination with data reductuction rules
- Application to Edge Bipartization
- Combination with heuristics