Algorithm Engineering for Optimal Graph Bipartization

Jiong Guo Jens Gramm Falk Hüffner Rolf Niedermeier
Sebastian Wernicke

Institut für Informatik
Friedrich-Schiller-Universität Jena

Dagstuhl Seminar N° 05301: Exact Algorithms and Fixed-Parameter Tractability

Outline

Introduction and Motivation

Iterative Compression for GRAPH BIPARTIZATION

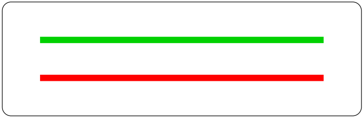
- An $O^*(2^k)$-time algorithm for EDGE BIPARTIZATION
- An $O^*(3^k)$-time algorithm for VERTEX BIPARTIZATION

Experimental Results for VERTEX BIPARTIZATION

- Runtime Improvements

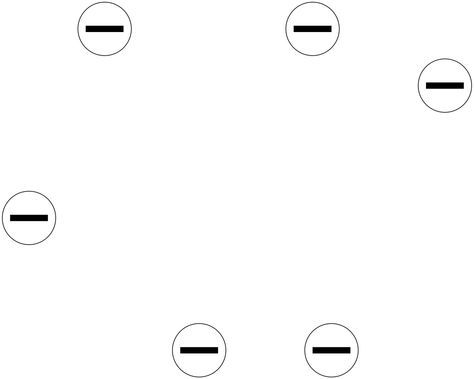
DNA Sequence Assembly

Cells have two slightly different copies of each chromosome



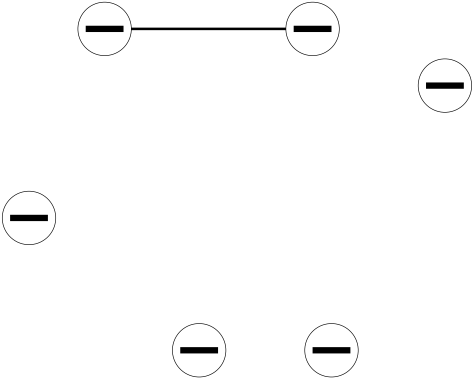
DNA Sequence Assembly

Assignments of the fragments to copies are initially unknown



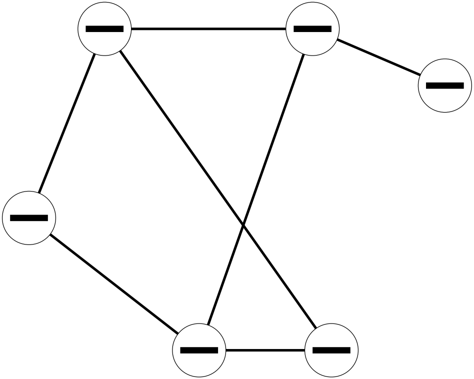
DNA Sequence Assembly

Pairwise conflicts indicate that two fragments are from different copies



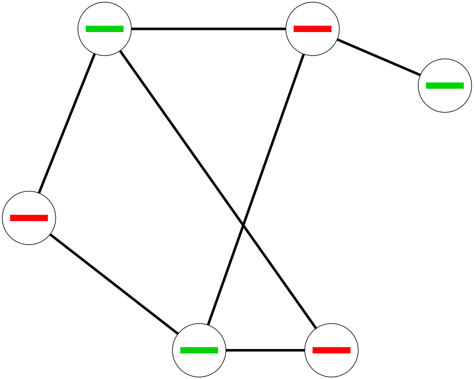
DNA Sequence Assembly

Pairwise conflicts indicate that two fragments are from different copies



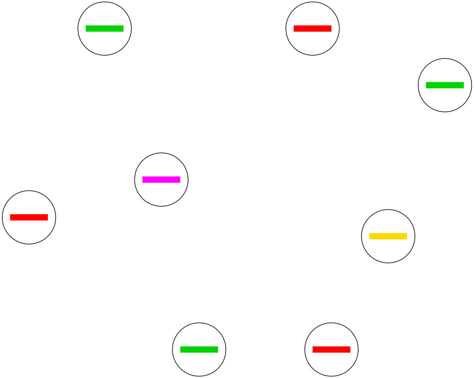
DNA Sequence Assembly

Reconstruction of assignment from the bipartite conflict graph



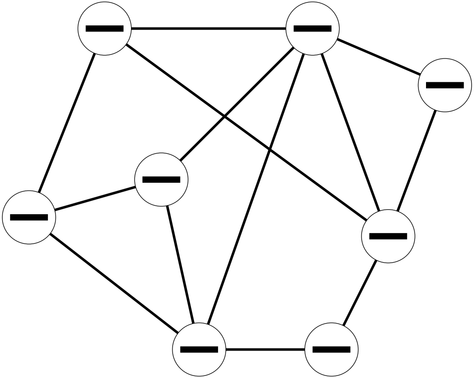
Minimum Fragment Removal

In practise, contaminations occur.



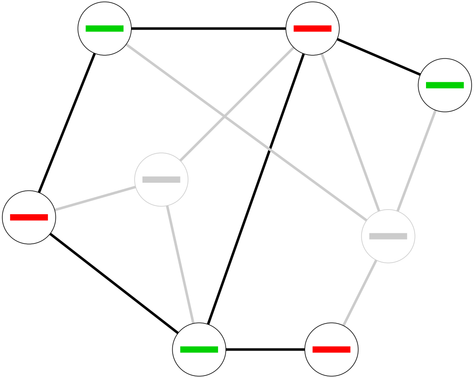
Minimum Fragment Removal

Contamination fragments will conflict with fragments from both copies.



Minimum Fragment Removal

The task is to recognize contamination fragments.



Formalization as Vertex Bipartization

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

Formalization as Vertex Bipartization

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

Equivalent formulation:

ODD CYCLE COVER
Task: *Find a subset $C \subseteq V$ of vertices with $|C| \leq k$ such that C touches every odd length cycle in G.*

Formalization as Vertex Bipartization

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

Equivalent formulation:

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

Edge Bipartization: Equivalent problem for deleting edges (parametric dual of MaxCut)

Vertex Bipartization and Edge Bipartization

- Numerous applications in computational biology, VLSI, register allocation, ...

Vertex Bipartization and Edge Bipartization

- Numerous applications in computational biology, VLSI, register allocation, ...
- NP-complete

  [Lewis&Yannakakis, J. Comput. Syst. Sci. 1980]

Vertex Bipartization and Edge Bipartization

- Numerous applications in computational biology, VLSI, register allocation, . . .
- NP-complete

  [Lewis&Yannakakis, J. Comput. Syst. Sci. 1980]
- MaxSNP-hard

  [Papadimitriou&Yannakakis, J. Comput. Syst. Sci. 1991]

Vertex Bipartization and Edge Bipartization

- Numerous applications in computational biology, VLSI, register allocation, ...
- NP-complete

  [Lewis&Yannakakis, J. Comput. Syst. Sci. 1980]
- MaxSNP-hard

  [Papadimitriou&Yannakakis, J. Comput. Syst. Sci. 1991]
- Best known approximation is by a factor of $\log |V|$

  [Garg, Vazirani&Yannakakis, SIAM J. Comput. 1996]

VERTEX BIPARTIZATION and EDGE BIPARTIZATION

- Numerous applications in computational biology, VLSI, register allocation, ...
- NP-complete

  [LEWIS&YANNAKAKIS, J. COMPUT. SYST. SCI. 1980]
- MaxSNP-hard

  [PAPADIMITRIOU&YANNAKAKIS, J. COMPUT. SYST. SCI. 1991]
- Best known approximation is by a factor of $\log |V|$

  [GARG, VAZIRANI&YANNAKAKIS, SIAM J. COMPUT. 1996]
- Fixed-parameter tractable with respect to k

  [REED, SMITH&VETTA, OPER. RES. LETT. 2004]

Iterative Compression

Idea: 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.

Iterative Compression

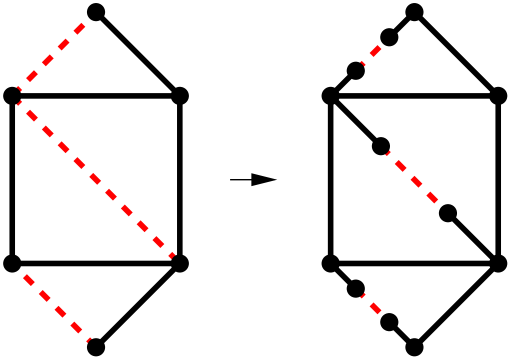
Idea: Use a *compression routine* iteratively.

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

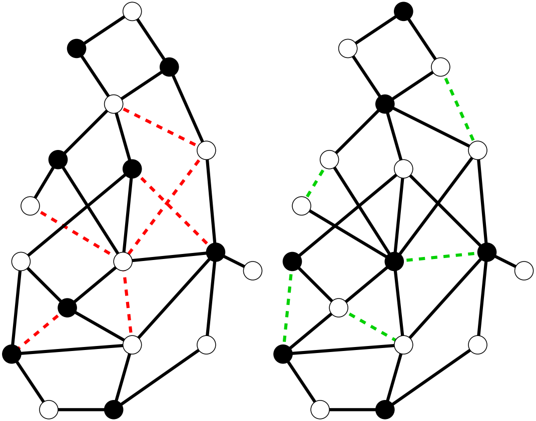
Algorithm:
Start with empty graph G' and empty edge bipartization set C
For each edge e in G:
  Add e to both G' and C
  Compress C using the compression routine

Iterative Compression for EDGE BIPARTIZATION

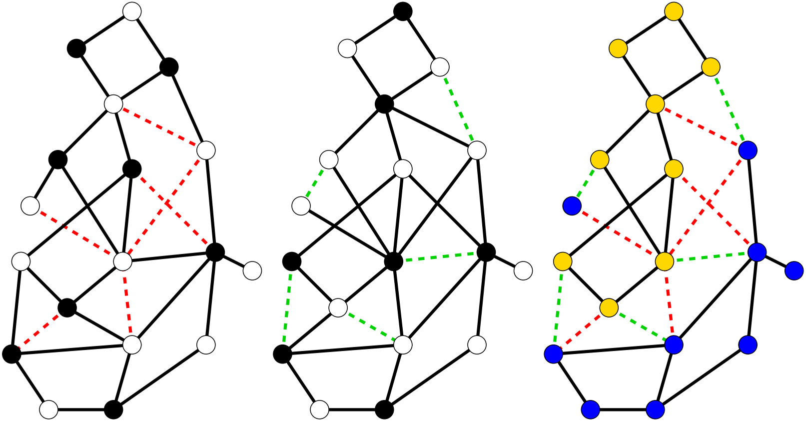
Preprocessing for the compression routine: Transform the input such that one can assume w.l.o.g. that the smaller solution is disjoint from the known one.



Comparing Disjoint Edge Bipartization Sets

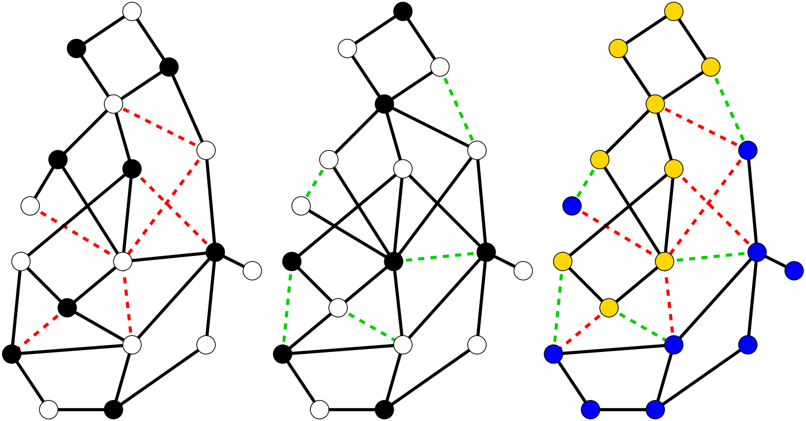


Comparing Disjoint Edge Bipartization Sets



$$\Phi := \begin{cases} \bullet \text{ for } (\bullet, \circ) \text{ or } (\circ, \bullet) \\ \bullet \text{ for } (\bullet, \bullet) \text{ or } (\circ, \circ) \end{cases}$$

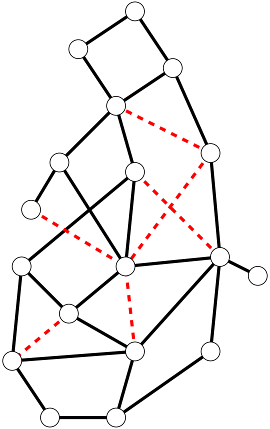
Comparing Disjoint Edge Bipartization Sets



$$\Phi := \begin{cases} {\color{gold}\bullet} \text{ for } (\bullet, \circ) \text{ or } (\circ, \bullet) \\ {\color{blue}\bullet} \text{ for } (\bullet, \bullet) \text{ or } (\circ, \circ) \end{cases}$$

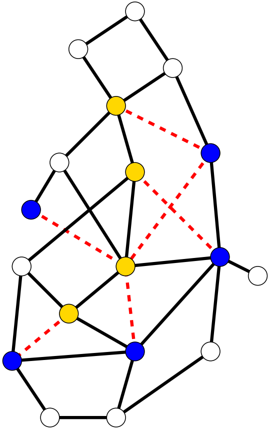
$\{{\color{red}\vdots}\ {\color{green}\vdots}\}$ is an edge cut between $\{{\color{gold}\bullet}\}$ and $\{{\color{blue}\bullet}\}$

Discovering a smaller edge bipartization set



Given: $G = (V, E)$ and an edge bipartization $C \subseteq E$ without redundant edges (⁞)

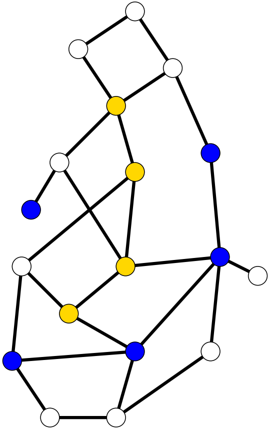
Discovering a smaller edge bipartization set



Given: $G = (V, E)$ and an edge bipartization $C \subseteq E$ without redundant edges (⁝)

- Guess Φ at the endpoints of the edges in C

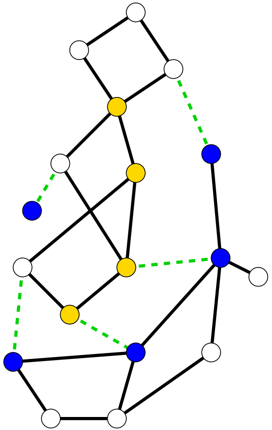
Discovering a smaller edge bipartization set



Given: $G = (V, E)$ and an edge bipartization $C \subseteq E$ without redundant edges (⁝)

- Guess Φ at the endpoints of the edges in C
- Find a minimum edge cut between {●} and {●} with the Edmonds–Karp MaxFlow algorithm

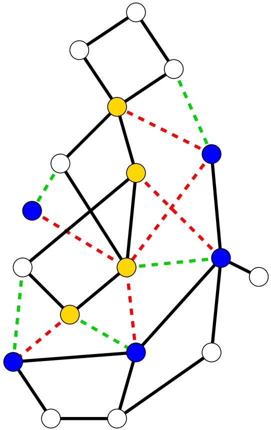
Discovering a smaller edge bipartization set



Given: $G = (V, E)$ and an edge bipartization $C \subseteq E$ without redundant edges (⁝)

- Guess Φ at the endpoints of the edges in C
- Find a minimum edge cut between {●} and {●} with the Edmonds–Karp MaxFlow algorithm

Discovering a smaller edge bipartization set



Given: $G = (V, E)$ and an edge bipartization $C \subseteq E$ without redundant edges ( )

- Guess Φ at the endpoints of the edges in C
- Find a minimum edge cut between $\{\bullet\}$ and $\{\bullet\}$ with the Edmonds–Karp MaxFlow algorithm
- Any such cut is a solution!

Run Time for Edge Bipartization

- Compress m times
- Try 2^k values for Φ
- Find k times an augmenting path in time $O(m)$

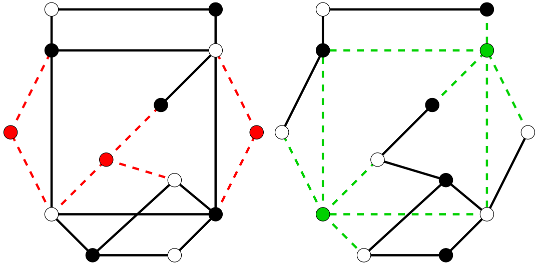
Theorem ([Guo et al., WADS'05])

Edge Bipartization *can be solved in* $O(2^k \cdot km^2)$ *time.*

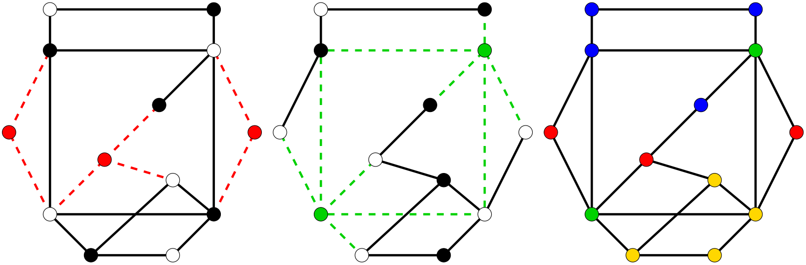
Adaption to Vertex Bipartization

- Input transformation to ensure solution disjointness no longer works
- Workaround: Try all 2^k bipartitions of the solution into vertices to keep and vertices to exchange.
- Additional cost: Factor of 2^k

Comparing Disjoint Vertex Bipartization Sets

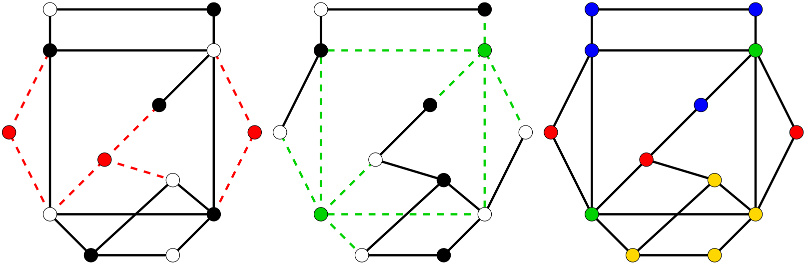


Comparing Disjoint Vertex Bipartization Sets



$$\Phi := \begin{cases} \textcolor{gold}{\bullet} \text{ for } (\bullet, \circ) \text{ or } (\circ, \bullet) \\ \textcolor{blue}{\bullet} \text{ for } (\bullet, \bullet) \text{ or } (\circ, \circ) \end{cases}$$

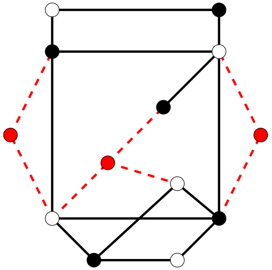
Comparing Disjoint Vertex Bipartization Sets



$$\Phi := \begin{cases} 🟡 \text{ for } (\bullet, \circ) \text{ or } (\circ, \bullet) \\ 🔵 \text{ for } (\bullet, \bullet) \text{ or } (\circ, \circ) \end{cases}$$

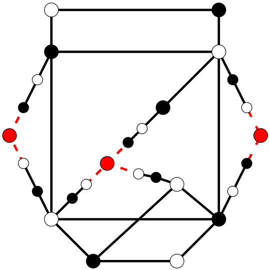
{🔴 🟢} is a vertex cut between {🟡} and {🔵}

Discovering a smaller vertex bipartization set



Given: $G = (V, E)$ and a vertex bipartization $C \subseteq V$ without redundant vertices

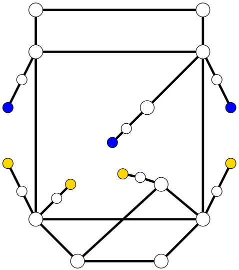
Discovering a smaller vertex bipartization set



Given: $G = (V, E)$ and a vertex bipartization $C \subseteq V$ without redundant vertices

- Subdivide edges around vertices in C by two vertices

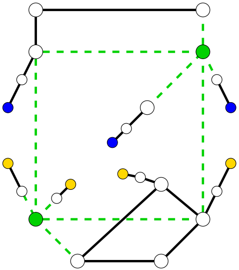
Discovering a smaller vertex bipartization set



Given: $G = (V, E)$ and a vertex bipartization $C \subseteq V$ without redundant vertices

- Subdivide edges around vertices in C by two vertices
- Guess Φ around the vertices in C

Discovering a smaller vertex bipartization set



Given: $G = (V, E)$ and a vertex bipartization $C \subseteq V$ without redundant vertices

- Subdivide edges around vertices in C by two vertices
- Guess Φ around the vertices in C
- Find a minimum vertex cut between $\{\bullet\}$ and $\{\bullet\}$ with the Edmonds–Karp MaxFlow algorithm

Run Time for Vertex Bipartization

- Compress n times
- Try 3 roles for each vertex from the vertex bipartization set:
  - remains in vertex bipartization set
  - first possible value of Φ for the neighbors
  - second possible value of Φ for the neighbors
- Find k times an augmenting path in time $O(m)$

Theorem ([Reed, Smith&Vetta, Oper. Res. Lett. 2004])

Vertex 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		

[H., WEA'05; 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			

[H., WEA'05; data from Wernicke 2003]

A Heuristic for Dense Graphs

- If two vertices in the vertex bipartization set are connected by an edge, then the guess of Φ for them is coupled

A Heuristic for Dense Graphs

- If two vertices in the vertex bipartization set are connected by an edge, then the guess of Φ for them is coupled
- No worst-case speedup for general graphs, but very effective in practice

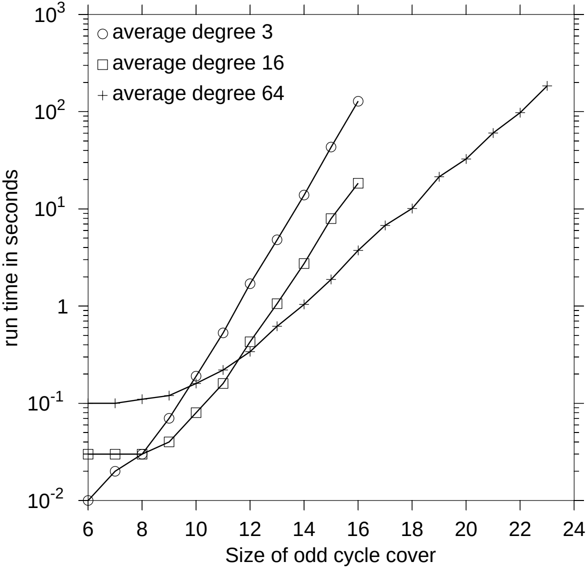
Experimental Results

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

[H., WEA'05; data from Wernicke 2003]

Run time for random planted bipartitions ($n = 300$)



Conclusions and Outlook

- 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:

- Reduction rules and kernel
- Combination with heuristics