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 No 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
Assignments of the fragments to copies are initially unknown.
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.
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.

**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)
**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 $G$.
**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

Vertex Bipartization and Edge Bipartization

- Numerous applications in computational biology, VLSI, register allocation, ...  
- NP-complete  
- MaxSNP-hard  
Vertex Bipartization and Edge Bipartization

- Numerous applications in computational biology, VLSI, register allocation, ...  
- NP-complete  
- MaxSNP-hard  
- Best known approximation is by a factor of $\log |V|$  
Vertex Bipartization and Edge Bipartization

- Numerous applications in computational biology, VLSI, register allocation, ...
- NP-complete
- MaxSNP-hard
- Best known approximation is by a factor of $\log |V|$
- 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

\[
\Phi :=
\begin{align*}
&\{ \text{for (, ) or (, )} \\
&\{ \text{for (, ) or (, )} \\
\}
\end{align*}
\]

is an edge cut between \{\} and \{\}.
Comparing Disjoint Edge Bipartization Sets

$$\Phi := \begin{cases} \text{for (●, ○) or (○, ●)} & \text{or} \\
\text{blue for (●, ●) or (○, ○)} \end{cases}$$
Comparing Disjoint Edge Bipartization Sets

\[ \Phi := \{ \text{for } (\bullet, \circ) \text{ or } (\circ, \bullet) \} \]

\[ \{ \bullet, \circ \} \text{ is an edge cut between } \{ \circ \} \text{ and } \{ \bullet \} \]
Discovering a smaller edge bipartization set

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

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

- Guess $\Phi$ 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.

1. Guess \( \Phi \) at the endpoints of the edges in \( C \).
2. Find a minimum edge cut between \( \{ \text{site} \} \) and \( \{ \text{site} \} \) with the Edmonds–Karp MaxFlow algorithm.

Any such cut is a solution!
Discovering a smaller edge bipartization set

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

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

Any such cut is a solution!
Discovering a smaller edge bipartization set

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

▶ Guess \( \Phi \) at the endpoints of the edges in \( C \)
▶ Find a minimum edge cut between \{\( \bullet \)\} and \{\( \circ \)\} 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 $\Phi$
- 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

Φ := \{ (x, y), (z, w) \}
is a vertex cut between \{x, y\} and \{z, w\}.
Comparing Disjoint Vertex Bipartization Sets

\[ \Phi := \begin{cases} 
\text{for } (\bullet, \circ) \text{ or } (\circ, \bullet) \\
\text{for } (\bullet, \bullet) \text{ or } (\circ, \circ)
\end{cases} \]
Comparing Disjoint Vertex Bipartization Sets

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

\{\text{red, green}\} \text{ is a vertex cut between } \{\text{yellow}\} \text{ and } \{\text{blue}\}$
Discovering a smaller vertex bipartition 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

- Guess $\Phi$ around the vertices in $C$

- Find a minimum vertex cut between $\{\}$ and $\{\}$ with the Edmonds–Karp MaxFlow algorithm
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 $\Phi$ 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 $\Phi$ around the vertices in $C$
- Find a minimum vertex cut between $\{\bigcirc\}$ and $\{\bigotimes\}$ 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 $\Phi$ for the neighbors
  - second possible value of $\Phi$ 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

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

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

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

[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 $\Phi$ 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 $\Phi$ 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

<table>
<thead>
<tr>
<th></th>
<th>n</th>
<th>m</th>
<th>k</th>
<th>ILP</th>
<th>Reed</th>
<th>Gray</th>
<th>Enum2Col</th>
</tr>
</thead>
<tbody>
<tr>
<td>A31</td>
<td>30</td>
<td>51</td>
<td>2</td>
<td>0.02</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>J24</td>
<td>142</td>
<td>387</td>
<td>4</td>
<td>0.97</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>A10</td>
<td>69</td>
<td>191</td>
<td>6</td>
<td>2.50</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>J18</td>
<td>71</td>
<td>296</td>
<td>9</td>
<td>47.86</td>
<td>0.05</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>A11</td>
<td>102</td>
<td>307</td>
<td>11</td>
<td>6248.12</td>
<td>0.79</td>
<td>0.14</td>
<td>0.00</td>
</tr>
<tr>
<td>A34</td>
<td>133</td>
<td>451</td>
<td>13</td>
<td>10.13</td>
<td>1.04</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>A22</td>
<td>167</td>
<td>641</td>
<td>16</td>
<td>350.00</td>
<td>64.88</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>A50</td>
<td>113</td>
<td>468</td>
<td>18</td>
<td>3072.82</td>
<td>270.60</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>A45</td>
<td>80</td>
<td>386</td>
<td>20</td>
<td>2716.87</td>
<td>1.14</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>A40</td>
<td>136</td>
<td>620</td>
<td>22</td>
<td>0.80</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A17</td>
<td>151</td>
<td>633</td>
<td>25</td>
<td>5.68</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A28</td>
<td>167</td>
<td>854</td>
<td>27</td>
<td>1.02</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A42</td>
<td>236</td>
<td>1110</td>
<td>30</td>
<td>73.55</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A41</td>
<td>296</td>
<td>1620</td>
<td>40</td>
<td>236.26</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[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