Quadratic programming on graphs without long odd cycles

Marcin Kamiński

Department of Computer Science
Université Libre de Bruxelles

March 2008
Graphs without long odd cycles

Let $\mathcal{G}(K)$ be the class of graphs without odd (not necessarily induced) cycles longer than $2K + 1$.

$\mathcal{G}(0) = \text{the class of bipartite graphs}$

$\mathcal{G}(1) = \text{the class consisting of all graphs whose line graphs are perfect}$
Let $\mathcal{G}(K)$ be the class of graphs without odd (not necessarily induced) cycles longer than $2K + 1$.

$\mathcal{G}(0) =$ the class of bipartite graphs

$\mathcal{G}(1) =$ the class consisting of all graphs whose line graphs are perfect
Graphs without long odd cycles

Let $\mathcal{G}(K)$ be the class of graphs without odd (not necessarily induced) cycles longer than $2K + 1$.

$\mathcal{G}(0)$ = the class of bipartite graphs

$\mathcal{G}(1)$ = the class consisting of all graphs whose line graphs are perfect
Motivation

Most combinatorial problems on graphs can be solved efficiently for trees and many can be solved efficiently over $G(0)$. Thus the absence of cycles (or odd cycles) seems to make graph optimization problems easy, and it is natural to ask the question if the absence of long cycles (or long odd cycles) also makes a graph optimization problem easier.

Max-Cut and Maximum-Stable-Set


**Max-Stable-Set**

A *stable set* is a subset of vertices such that no edge has both endpoints in this set.

The *value* of a stable set is the sum of weights of the vertices belonging to the set.

A stable set of maximum value is called a *maximum stable set*.

**Max-Stable-Set** problem is to find a maximum stable set.
A *stable set* is a subset of vertices such that no edge has both endpoints in this set.

The *value* of a stable set is the sum of weights of the vertices belonging to the set.

A stable set of maximum value is called a *maximum stable set*. 

**Max-Stable-Set** problem is to find a maximum stable set.
MAX-STABLE-SET

A *stable set* is a subset of vertices such that no edge has both endpoints in this set.

The *value* of a stable set is the sum of weights of the vertices belonging to the set.

A stable set of maximum value is called a *maximum stable set*.

MAX-STABLE-SET problem is to find a maximum stable set.
A stable set is a subset of vertices such that no edge has both endpoints in this set.

The value of a stable set is the sum of weights of the vertices belonging to the set.

A stable set of maximum value is called a maximum stable set.

Max-Stable-Set problem is to find a maximum stable set.
A cut in a graph is a partition of its vertex set into two disjoint parts.

The value of a cut is the sum of weights of the edges whose endpoints belong to two different parts of the cut.

A cut of maximum value is called a maximum cut.

Max-Cut problem is to find a maximum cut.
A cut in a graph is a partition of its vertex set into two disjoint parts.

The value of a cut is the sum of weights of the edges whose endpoints belong to two different parts of the cut.

A cut of maximum value is called a maximum cut.

Max-Cut problem is to find a maximum cut.
A cut in a graph is a partition of its vertex set into two disjoint parts.

The value of a cut is the sum of weights of the edges whose endpoints belong to two different parts of the cut.

A cut of maximum value is called a maximum cut.

Max-Cut problem is to find a maximum cut.
A cut in a graph is a partition of its vertex set into two disjoint parts.

The value of a cut is the sum of weights of the edges whose endpoints belong to two different parts of the cut.

A cut of maximum value is called a maximum cut.

Max-Cut problem is to find a maximum cut.
A graph $G = (V, E)$ with the vertex set $v_1, \ldots, v_n$.

An edge between $v_i$ and $v_j$ is denoted by $e_{ij}$.

Weights on vertices: $a_i \in \mathbb{R}$ for $v_i \in V$.

Weights on edges: $c_{ij} \geq 0$ for $e_{ij} \in E$. 
A graph $G = (V, E)$ with the vertex set $v_1, \ldots, v_n$.

An edge between $v_i$ and $v_j$ is denoted by $e_{ij}$.

Weights on vertices: $a_i \in \mathbb{R}$ for $v_i \in V$.

Weights on edges: $c_{ij} \geq 0$ for $e_{ij} \in E$. 

Quadratic Binary Programming on Graphs
A graph $G = (V, E)$ with the vertex set $v_1, \ldots, v_n$.

An edge between $v_i$ and $v_j$ is denoted by $e_{ij}$.

Weights on vertices: $a_i \in \mathbb{R}$ for $v_i \in V$.

Weights on edges: $c_{ij} \geq 0$ for $e_{ij} \in E$. 
A graph $G = (V, E)$ with the vertex set $v_1, \ldots, v_n$. 

An edge between $v_i$ and $v_j$ is denoted by $e_{ij}$. 

Weights on vertices: $a_i \in \mathbb{R}$ for $v_i \in V$. 

Weights on edges: $c_{ij} \geq 0$ for $e_{ij} \in E$. 
The **QBPG** problem is

\[
\text{max } \sum_i a_i x_i - \sum_{ij} c_{ij} x_i x_j \\
\text{s.t. } x_i \in \{0, 1\}
\]

The goal: To select a subset of vertices $X$ with maximum profit.

Gain: $a_i$ is the profit we get by including vertex $v_i$.

Loss: $c_{ij}$ is the penalty we pay if both endpoints of edge $e_{ij}$ are included.
The QBPG problem is

$$\begin{align*}
\text{max} & \quad \sum_i a_i x_i - \sum_{ij} c_{ij} x_i x_j \\
\text{s.t.} & \quad x_i \in \{0, 1\}
\end{align*}$$

The goal: To select a subset of vertices $X$ with maximum profit.

Gain: $a_i$ is the profit we get by including vertex $v_i$.

Loss: $c_{ij}$ is the penalty we pay if both endpoints of edge $e_{ij}$ are included.
Max-Stable-Set as QBPG

Max-Stable-Set is expressible in the QBPG framework:

\[
\max \sum_i a_i x_i - \sum_{ij} M x_i x_j \\
\text{s.t. } x_i \in \{0, 1\}
\]

\(M\) is "large"
Max-Stable-Set is expressible in the QBPG framework:

$$\max \sum_i a_i x_i - \sum_{ij} M x_i x_j$$

s.t. \( x_i \in \{0, 1\} \)

\(M\) is "large"
Max-Cut as QBPG

Max-Cut is expressible in the QBPG framework:

$$\max \sum_{ij} w_{ij} (x_i \overline{x_j} + \overline{x_i} x_j)$$

s.t. $x_i \in \{0, 1\}$

$$\max \sum_i W_i x_i - 2 \sum_{ij} w_{ij} x_i x_j$$

s.t. $x_i \in \{0, 1\}$

$$W_i = 2 \sum_{ij} w_{ij}$$
Max-Cut as QBPG

Max-Cut is expressible in the QBPG framework:

$$\max \sum_{ij} w_{ij} (x_i \overline{x}_j + \overline{x}_i x_j)$$

s.t. $x_i \in \{0, 1\}$

$$\max \sum_i W_i x_i - 2 \sum_{ij} \overline{w}_{ij} x_i x_j$$

s.t. $x_i \in \{0, 1\}$

$$W_i = 2 \sum_{ij} w_{ij}$$
**Max-Cut as QBPG**

Max-Cut is expressible in the QBPG framework:

\[
\begin{align*}
\text{max} & \quad \sum_{ij} w_{ij} (x_i \bar{x}_j + \bar{x}_i x_j) \\
\text{s.t.} & \quad x_i \in \{0, 1\}
\end{align*}
\]

\[
\begin{align*}
\text{max} & \quad \sum_i W_i x_i - 2 \sum_{ij} w_{ij} x_i x_j \\
\text{s.t.} & \quad x_i \in \{0, 1\}
\end{align*}
\]

\[W_i = 2 \sum_{ij} w_{ij}\]
**Theorem**

*QBPG is solvable in polynomial time in the class of bipartite graphs.*

**Sketch of Proof:**
- Consider a linearization of QBPG
- Consider a relaxation of the linearization
- Surprise: the constraint matrix is totally unimodular! (Assuming $G$ is bipartite.)

Note: True statement for MAX-CUT and MAX-STABLE-SET.
Theorem

QBPG is solvable in polynomial time in the class of bipartite graphs.

Sketch of Proof:

- Consider a linearization of QBPG
- Consider a relaxation of the linearization
- Surprise: the constraint matrix is totally unimodular!
  (Assuming $G$ is bipartite.)

Note: True statement for MAX-CUT and MAX-STABLE-SET.
**Theorem**

*QBPG is solvable in polynomial time in the class of bipartite graphs.*

**Sketch of Proof:**

- Consider a linearization of QBPG
- Consider a relaxation of the linearization
- Surprise: the constraint matrix is totally unimodular! (Assuming $G$ is bipartite.)

Note: True statement for MAX-CUT and MAX-STABLE-SET.
QBPG in bipartite graphs

**Theorem**

*QBPG is solvable in polynomial time in the class of bipartite graphs.*

**Sketch of Proof:**

- Consider a linearization of QBPG
- Consider a relaxation of the linearization
- Surprise: the constraint matrix is totally unimodular!
  (Assuming $G$ is bipartite.)

*Note: True statement for MAX-CUT and MAX-STABLE-SET.*
Theorem

QBPG is solvable in polynomial time in the class of bipartite graphs.

Sketch of Proof:

- Consider a linearization of QBPG
- Consider a relaxation of the linearization
- Surprise: the constraint matrix is totally unimodular! (Assuming $G$ is bipartite.)

Note: True statement for Max-Cut and Max-Stable-Set.
Linearization of QBPG (proof)

(1) Formulation

\[
\begin{align*}
\text{max} & \quad \sum_i a_i x_i - \sum_{ij} c_{ij} x_i x_j \\
\text{s.t.} & \quad x_i \in \{0, 1\}
\end{align*}
\]

(2) Linearization

\[
\begin{align*}
\text{max} & \quad \sum_i a_i x_i - \sum_{ij} c_{ij} y_{ij} \\
\text{s.t.} & \quad y_{ij} \geq x_i + x_j - 1 \\
& \quad x_i, y_{ij} \in \{0, 1\}
\end{align*}
\]

Claim

The two above IPs are equivalent \((y_{ij} = x_i x_j)\)
Linearization of QBPG (proof)

(1) Formulation

\[
\begin{align*}
\max & \quad \sum_i a_i x_i - \sum_{ij} c_{ij} x_i x_j \\
\text{s.t.} & \quad x_i \in \{0, 1\}
\end{align*}
\]

(2) Linearization

\[
\begin{align*}
\max & \quad \sum_i a_i x_i - \sum_{ij} c_{ij} y_{ij} \\
\text{s.t.} & \quad y_{ij} \geq x_i + x_j - 1 \\
& \quad x_i, y_{ij} \in \{0, 1\}
\end{align*}
\]

Claim

*The two above IPs are equivalent \((y_{ij} = x_i x_j)\)*
Linearization of QBPG (proof)

(1) Formulation

\[
\max \sum_i a_i x_i - \sum_{ij} c_{ij} x_i x_j \\
\text{s.t. } x_i \in \{0, 1\}
\]

(2) Linearization

\[
\max \sum_i a_i x_i - \sum_{ij} c_{ij} y_{ij} \\
\text{s.t. } y_{ij} \geq x_i + x_j - 1 \\
x_i, y_{ij} \in \{0, 1\}
\]

Claim

*The two above IPs are equivalent* \((y_{ij} = x_i x_j)\)
(2) Linearization

\[
\begin{align*}
\text{max} & \quad \sum_i a_i x_i - \sum_{ij} c_{ij} y_{ij} \\
\text{s.t.} & \quad y_{ij} \geq x_i + x_j - 1 \\
& \quad x_i, y_{ij} \in \{0, 1\}
\end{align*}
\]

(3) Relaxation

\[
\begin{align*}
\text{max} & \quad \sum_i a_i x_i - \sum_{ij} c_{ij} y_{ij} \\
\text{s.t.} & \quad y_{ij} \geq x_i + x_j - 1 \\
& \quad x_i, y_{ij} \in [0, 1]
\end{align*}
\]
Relaxation of the linearization (proof)

(2) Linearization

\[
\begin{align*}
\text{max} & \quad \sum_i a_i x_i - \sum_{ij} c_{ij} y_{ij} \\
\text{s.t.} & \quad y_{ij} \geq x_i + x_j - 1 \\
& \quad x_i, y_{ij} \in \{0, 1\}
\end{align*}
\]

(3) Relaxation

\[
\begin{align*}
\text{max} & \quad \sum_i a_i x_i - \sum_{ij} c_{ij} y_{ij} \\
\text{s.t.} & \quad y_{ij} \geq x_i + x_j - 1 \\
& \quad x_i, y_{ij} \in [0, 1]
\end{align*}
\]
(3) Relaxation

\[
\begin{align*}
\text{max} & \quad \sum_i a_i x_i - \sum_{ij} c_{ij} y_{ij} \\
\text{s.t.} & \quad x_i + x_j - 1 \leq y_{ij} \\
& \quad x_i \geq 0 \\
& \quad x_i \leq 1 \\
& \quad y_{ij} \geq 0 \\
& \quad y_{ij} \leq 1
\end{align*}
\]

Lemma

If $G$ is bipartite, then the constraint matrix of the LP (3) is totally unimodular.
(3) Relaxation

\[ \begin{align*}
\text{max} & \quad \sum_i a_i x_i - \sum_{ij} c_{ij} y_{ij} \\
\text{s.t.} & \quad x_i + x_j - 1 \leq y_{ij} \\
& \quad x_i \geq 0 \\
& \quad x_i \leq 1 \\
& \quad y_{ij} \geq 0 \\
& \quad y_{ij} \leq 1
\end{align*} \]

**Lemma**

*If G is bipartite, then the constraint matrix of the LP (3) is totally unimodular.*
QBPG on graphs without long odd cycles

**Theorem**

*QBPG is solvable in polynomial time in the class of $\mathcal{G}(K)$ for any fixed $K \geq 0$.*

Inductive proof: we can solve QBPG in polynomial time in the class of bipartite graphs ($= \mathcal{G}(0)$).

We assume $K \geq 1$ and that we can solve QBPG in polynomial time in the class of $\mathcal{G}(K - 1)$.

Note: we are not concerned with optimality, just solvability in polynomial time. (Scarrrrrry factors around!)
Theorem

QBPG is solvable in polynomial time in the class of $\mathcal{G}(K)$ for any fixed $K \geq 0$.

Inductive proof: we can solve QBPG in polynomial time in the class of bipartite graphs ($= \mathcal{G}(0)$).

We assume $K \geq 1$ and that we can solve QBPG in polynomial time in the class of $\mathcal{G}(K - 1)$.

Note: we are not concerned with optimality, just solvability in polynomial time. (Scarrrrrry factors around!)
Theorem

QBPG is solvable in polynomial time in the class of $\mathcal{G}(K)$ for any fixed $K \geq 0$.

Inductive proof: we can solve QBPG in polynomial time in the class of biparite graphs ($= \mathcal{G}(0)$).

We assume $K \geq 1$ and that we can solve QBPG in polynomial time in the class of $\mathcal{G}(K - 1)$.

Note: we are not concerned with optimality, just solvability in polynomial time. (Scarrrrrry factors around!)
Theorem

QBPG is solvable in polynomial time in the class of $\mathcal{G}(K)$ for any fixed $K \geq 0$.

Inductive proof: we can solve QBPG in polynomial time in the class of bipartite graphs ($= \mathcal{G}(0)$).

We assume $K \geq 1$ and that we can solve QBPG in polynomial time in the class of $\mathcal{G}(K - 1)$.

Note: we are not concerned with optimality, just solvability in polynomial time. (Scarrrrrry factors around!)
Definition

A *cutvertex* is a vertex whose removal increases the number of connected components.

A *block* in a graph is a maximal connected subgraph without a cutvertex.

Every block of a connected graph is either a maximal 2-connected subgraph or a bridge.
A cutvertex is a vertex whose removal increases the number of connected components.

A block in a graph is a maximal connected subgraph without a cutvertex.

Every block of a connected graph is either a maximal 2-connected subgraph or a bridge.
Definitions

A cutvertex is a vertex whose removal increases the number of connected components.

A block in a graph is a maximal connected subgraph without a cutvertex.

Every block of a connected graph is either a maximal 2-connected subgraph or a bridge.
Blocks

Lemma

Any two longest odd cycles of a 2-connected graph intersect.

Corollary

If $G \in \mathcal{G}(K)$ is 2-connected and $C$ is a longest odd cycle of $G$, then $G[V - C] \in \mathcal{G}(K - 1)$. 
Blocks

**Lemma**

*Any two longest odd cycles of a 2-connected graph intersect.*

**Corollary**

*If $G \in \mathcal{G}(K)$ is 2-connected and $C$ is a longest odd cycle of $G$, then $G[V - C] \in \mathcal{G}(K - 1)$.*
Solving QBPG on blocks

Solving QBPG on a 2-connected graph $G$:

- Find a longest odd cycle $C$ in $G$.
- Consider all 0,1 assignments of values to vertices of $C$.
- For each assignment solve the remaining instance.
Solving QBPG on blocks

Solving QBPG on a 2-connected graph $G$:

- Find a longest odd cycle $C$ in $G$.
- Consider all 0, 1 assignments of values to vertices of $C$.
- For each assignment solve the remaining instance.
Solving QBPG on a 2-connected graph $G$:

- Find a longest odd cycle $C$ in $G$.
- Consider all 0, 1 assignments of values to vertices of $C$.
- For each assignment solve the remaining instance.
Solving QBPG on a 2-connected graph $G$:

- Find a longest odd cycle $C$ in $G$.
- Consider all 0, 1 assignments of values to vertices of $C$.
- For each assignment solve the remaining instance.
Different blocks overlap in at most one vertex (which is a cutvertex of the graph).

The bipartite graph of incidence between cutvertices and blocks is a tree.

**Claim**

*Every connected graph with at least 2 blocks has a block containing exactly one cutvertex.*
Tree of blocks

Different blocks overlap in at most one vertex (which is a cutvertex of the graph).

The bipartite graph of incidence between cutvertices and blocks is a tree.

Claim

Every connected graph with at least 2 blocks has a block containing exactly one cutvertex.
Tree of blocks

Different blocks overlap in at most one vertex (which is a cutvertex of the graph).

The bipartite graph of incidence between cutvertices and blocks is a tree.

Claim

*Every connected graph with at least 2 blocks has a block containing exactly one cutvertex.*
Solving QBPG on the tree of blocks

Solving QBPG on a graph $G$:

- Find a block $B$ of $G$ with exactly one cutvertex $v_i$.
- Solve two instances of QBPG on $B$:
  one setting $x_i = 0$ and another with $x_i = 1$.
  Let $t_0$ and $t_1$ be the obtained values, respectively.
- Solve QBPG on $G - B + v_i$ setting $a_i = t_1 - t_0$. 
Solving QBPG on a graph $G$:

- Find a block $B$ of $G$ with exactly one cutvertex $v_i$.

- Solve two instances of QBPG on $B$:
  one setting $x_i = 0$ and another with $x_i = 1$.
  Let $t_0$ and $t_1$ be the obtained values, respectively.

- Solve QBPG on $G - B + v_i$ setting $a_i = t_1 - t_0$. 
Solving QBPG on a graph $G$:

- Find a block $B$ of $G$ with exactly one cutvertex $v_i$.
- Solve two instances of QBPG on $B$:
  one setting $x_i = 0$ and another with $x_i = 1$.
  Let $t_0$ and $t_1$ be the obtained values, respectively.
- Solve QBPG on $G - B + v_i$ setting $a_i = t_1 - t_0$. 
Solving QBPG on a graph $G$:

- Find a block $B$ of $G$ with exactly one cutvertex $v_i$.

- Solve two instances of QBPG on $B$:
  - one setting $x_i = 0$ and another with $x_i = 1$.
  - Let $t_0$ and $t_1$ be the obtained values, respectively.

- Solve QBPG on $G - B + v_i$ setting $a_i = t_1 - t_0$. 
Exact Max-Cut in sparse graphs


**Theorem**

There exists an algorithm which computes MAX-CUT in the class of graphs with maximum degree \( \Delta \) (\( \Delta \geq 3 \)) in time \( O^*(2^{(1-2/\Delta)n}) \) and polynomial space.
Thank you!