Quadratic programming on graphs without long odd cycles

Marcin Kamiński
Department of Computer Science
Université Libre de Bruxelles, Belgium
Marcin.Kaminski@ulb.ac.be

Abstract
We introduce a quadratic programming framework on graphs (which incorporates MAXIMUM-CUT and MAXIMUM-INDEPENDENT-SET) and show that problems which are expressible in the framework can be solved in polynomial time on graphs without long odd cycles.

1 Introduction
In the 1980s George L. Nemhauser et al. began studying computational complexity of combinatorial problems on graphs without long odd cycles. They introduced the class $\mathcal{G}(K)$ which consists of all the graphs without odd cycles longer than $2K + 1$, for a fixed integer $K \geq 0$. ($\mathcal{G}(0)$ is the class of bipartite graphs.) In two papers they designed algorithms for the (weighted) MAXIMUM-CUT [4] and (weighted) MAXIMUM-INDEPENDENT-SET [5] problems. (Both MAXIMUM-CUT and MAXIMUM-INDEPENDENT-SET are NP-hard for general graphs [6].) Both of their algorithms run in polynomial time, if the input graph belongs to $\mathcal{G}(K)$, for some integer $K \geq 0$.

Here is a motivation for studying algorithmic problems on graphs without long (odd) cycles from [4]:

Most combinatorial problems on graphs can be solved efficiently for trees and many can be solved efficiently over $\mathcal{G}(0)$. 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.

In this note we present a framework which we call quadratic binary programming on graphs (QBPG) and show that the problems expressible in QBPG can be solved efficiently in $\mathcal{G}(K)$ for every fixed integer $K \geq 0$. QBPG contains (weighted) MAXIMUM-CUT and (weighted) MAXIMUM-INDEPENDENT-SET as special cases. In this sense our work generalizes [4] and [5]. Another problem expressible as QBPG is (weighted) DIRECTED-MAXIMUM-CUT. The complexity of this problem in the class of graphs without long odd cycles does not follow from [4] and to the best of our knowledge has not been known before.
2 QBPG

Let \( G = (V, E) \) be a finite, undirected, loopless graph without multiple edges with weights on vertices and edges. The vertex set \( V \) of \( G \) consists of vertices \( v_1, \ldots, v_n \). The weight of a vertex \( v_i \) is a real number denoted by \( a_i \). The weight of an edge (with endpoints \( v_i, v_j \)) is a nonnegative real number denoted by \( w_{ij} \). For graph-theoretic terms not defined here, we refer the reader to [3].

For each subset \( S \subseteq V \) of vertices, we define its value to be the sum of weights on the vertices in \( S \) minus the weights of the edges whose both endpoints are in \( S \). The QBPG problem is to find a subset of vertices with maximum value. It can be thought of a task of maximizing profit: we gain \( a_i \) by including vertex \( v_i \) but have to pay a penalty of \( c_{ij} \) if both endpoints of edge \( e_{ij} \) are included.

QBPG can be expressed as a quadratic binary program:

\[
\begin{align*}
\max & \quad \sum_{v_i \in V} a_i x_i - \sum_{v_i v_j \in E} w_{ij} x_i x_j \\
\text{s.t.} & \quad x_i \in \{0, 1\}
\end{align*}
\] (1)

Now we want to show that both weighted MAXIMUM-CUT and weighted MAXIMUM-INDEPENDENT-SET are expressible in QBPG. (The unweighted versions of both problems are obtained if vertex-weights are all equal.)

The weighted MAXIMUM-INDEPENDENT-SET problem is to select a subset \( S \) of pairwise non-adjacent vertices of a vertex-weighted graph in such a way that the sum of the weights of the vertices in \( S \) is maximized. If \( M \) is large enough, then MAXIMUM-INDEPENDENT-SET can be expressed as the following quadratic binary program (clearly in QBPG):

\[
\begin{align*}
\max & \quad \sum_{v_i \in V} a_i x_i - \sum_{v_i v_j \in E} w_{ij} x_i x_j \\
\text{s.t.} & \quad x_i \in \{0, 1\}
\end{align*}
\]

The weighted MAXIMUM-CUT problem is to partition the vertex set of an edge-weighted graph into two parts in such a way that the sum of the weights of the edges crossing the partition is maximized. (The weights on edges are required to be positive real numbers.) It can be stated as follows:

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

Rewriting the above program and taking \( W_i = 2 \sum_{j : v_j \in E} w_{ij} \), we obtain the following formulation (clearly in QBPG):

\[
\begin{align*}
\max & \quad \sum_{v_i} W_i x_i - 2 \sum_{v_i v_j \in E} w_{ij} x_i x_j \\
\text{s.t.} & \quad x_i \in \{0, 1\}
\end{align*}
\]
In the weighted directed-maximum-cut problem we are given an edge-weighted directed graph \( G = (V,A) \) and want to partition the vertex set into two parts \( V_1 \) and \( V_2 \) in such a way that the sum of the weights of the arcs from \( V_1 \) to \( V_2 \) is maximized. (The weights on edges are required to be positive real numbers.) The problem can be expressed in a similar way to maximum-cut:

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

Rewriting the above program and taking \( W'_i = \sum_{j : v_i v_j \in A} w_{ij} \), and \( W''_i = \sum_{j : v_j v_i \in A} w_{ji} \), we obtain the following formulation (clearly in QBPG):

\[
\begin{align*}
\text{max} \quad & \sum_{v_i \in V} (W'_i + W''_i)x_i - 2\sum_{v_i v_j \in E} w_{ij} x_i x_j \\
\text{s.t.} \quad & x_i \in \{0,1\}
\end{align*}
\]

### 3 Bipartite graphs

In the class of bipartite graphs both \( \text{MAX-CUT} \) and \( \text{MAX-INDEPENDENT-SET} \) can be solved in polynomial time. The solution is trivial for the first problem, as \( \text{MAX-CUT} \) is equivalent to finding a maximum size bipartite subgraph of the input graph. \( \text{MAX-INDEPENDENT-SET} \) can be solved in bipartite graphs by a network flow technique. In this section we prove that QBPG can be solved in polynomial time in the class of bipartite graphs.

The following lemma was first proved in [2] in the context of maximum cut. We state it here in a more general form.

**Lemma 1.** The QBPG problem can be solved in polynomial time in the class of bipartite graphs.

**Proof.** Assume that \( G \) is a bipartite graph. Let us consider a linearization of (1). We introduce a new variable \( y_{ij} \) for each edge \( v_i v_j \in E \) and require that \( y_{ij} = x_i x_j \). Now (1) can be rewritten as

\[
\begin{align*}
\text{max} \quad & \sum_{v_i \in V} a_{ii} x_i - \sum_{v_i v_j \in E} w_{ij} y_{ij} \\
\text{s.t.} \quad & y_{ij} \geq x_i + x_j - 1 \\
& x_i \in \{0,1\} \\
& y_{ij} \in \{0,1\}
\end{align*}
\]

It is easy to see that (1) and (2) are equivalent. Their optimal values coincide and there is an easy correspondence between their optimal solutions, namely \( y_{ij} = x_i x_j \).

Now let us consider a continuous relaxation of (2).
max $\sum_{i \in V} a_i x_i - \sum_{(v_i,v_j) \in E} w_{ij} y_{ij}$ \hspace{1cm} (3)

s.t. $y_{ij} \geq x_i + x_j - 1$
$x_i \geq 0$
$x_j \leq 1$
$y_{ij} \geq 0$
$y_{ij} \leq 1$

Claim. The constraint matrix of the linear program (3) is totally unimodular.

Proof of Claim. Let $A$ be the constraint matrix of (3). It has $|V| + |E|$ columns and $2|V| + 3|E|$ rows and all its entries are either 0 or $\pm 1$. Let $B$ be an edge-vertex incidence matrix of $G$, with rows corresponding to edges and columns corresponding to vertices. Notice that $B$ is a submatrix of $A$. Moreover, any submatrix of $A$ that has two non-zero entries in every row and every column has to be a submatrix of $B$.

Take any square $k \times k$ submatrix of $A$. We will prove the claim by induction on $k$. Clearly, the result holds for $k = 1$.

Now assume that all $(k-1) \times (k-1)$ submatrices of $A$ are totally unimodular and consider a matrix $M$ which is a $k \times k$ submatrix of $A$.

If all entries of any row or column of $M$ are 0, then $\det(M) = 0$ and $M$ is totally unimodular. If any row or column of $M$ has a single non-zero element ($\pm 1$), then using the expansion method for calculating determinants and the induction hypothesis, it is easy to see that $\det(M)$ is either 0 or $\pm 1$, and $A$ is totally unimodular.

Suppose that each row and each column of $M$ has at least two non-zero entries. Hence, $M$ must be a submatrix of $B$ but, since $B$ is an incidence matrix of a bipartite graph, so is $M$. It is possible to partition the columns of $M$ into two parts, according to the partition of vertices of bipartite graph. The sum of the columns in each part yields a unit vector (each edge of the bipartite subgraph has one endpoint in each part) and that implies linear dependence of $M$, therefore $\det(M) = 0$ and $M$ is totally unimodular.

Since the constraint matrix of (3) is totally unimodular, then (3) has an optimal $0 - 1$ solution. Such solution is also optimal for (2) and, as we mentioned before, is an optimal solution to (1) as well.

An optimal solution to the linear program (3) can be computed in polynomial time (see for example [7]). This implies that the quadratic integer program (1) can also be solved in polynomial time.

4 Composing blocks

In this section we assume that QBPG can be solved in polynomial time on blocks (this will be proved in the next section) and show how to extend the solution on blocks to an algorithm running in polynomial-time for any graph in $\mathcal{G}(K)$. 

4
Lemma 2. Let \( \mathcal{G} \) be a class of graphs. If QBPG can be solved in polynomial time on 2-connected graphs from \( \mathcal{G} \), then it can be solved in polynomial time on all graphs from \( \mathcal{G} \).

Proof. Let \( G \in \mathcal{G} \) be an input for the algorithm. We proceed by induction on the number of blocks of \( G \). If \( G \) is 2-connected, then there is nothing to be done. Assume then that \( G \) is a graph with a cutvertex. Pick \( v_i \) to be a cutvertex of \( G \) such that at least one of the connected components of \( G - v_i \) is a block. Let us pick one of those blocks and call it \( B \). Notice that such a block always exists (Proposition 3.1.2 in [3]).

Let \( t_0 \) \((t_1)\) be the value of an optimal solution on \( B \), under the assumption that \( x_i = 0 \) \((x_i = 1\) respectively\). Also, let \( G' := G[V - V(B)] \) and \( t_0' \) \((t_1')\) be the value of an optimal solution on \( G \), under the assumption that \( x_i = 0 \) \((x_i = 1\) respectively\).

Now we are going to solve three instances of QBPG: two on \( B \) and one on \( G' = G[V - V(B)] \). We can solve all three in polynomial time: the instances on \( B \), by the assumption on existence of a polynomial-time algorithm for blocks, and the instance on \( G' \) by our inductive assumption.

- Solve QBPG on \( B \) setting \( x_i = 0 \). Let \( (x_j^0)_{v_j \in V(B)} \) be the optimal solution. Then, \( t_0 \) is the corresponding optimal value.
- Solve QBPG on \( B \) setting \( x_i = 1 \). Let \( (x_j^1)_{v_j \in V(B)} \) be the optimal solution. Then, \( t_1 \) is the corresponding optimal value.
- Solve QBPG on \( G' + v \) setting \( a_i = t_1 - t_0 \). Let \( (x_j^v)_{v_j \in V(G')} \) be the optimal solution and \( y = x_i' \). Then, \( t_v' \) is the corresponding optimal value.

Now we combine these three solutions into one solution \( (x_j)_{v_j \in V(G)} \) on \( G \). Let \( x_j = x_j^0 \) for all \( v_j \in V(G') \), and let \( x_j = x_j^v \) for all \( v_j \in V(B) \). Notice that the value corresponding to \( (x_j)_{v_j \in V(G)} \) is \( T := t_v + t_v' \). Also, since the value of \( x_j^v \) was chosen to be \( y \), then \( t_v + t_v' \geq t_v + t_v' \) \((y = 1 - y)\).

Claim. Let \( x^* \) be an optimal solution of QBPG on \( G \) and \( t^* \) the corresponding optimal value. Then, \( T = t^* \).

Proof of Claim. Since \( (x_j)_{v_j \in V(G)} \) is a feasible solution, then \( T \leq t^* \). Suppose \( t^* > T \) and \( z = x_j^v \). Recall that \( t_z \) is the value of the optimal solution on \( B \) under the assumption that \( x_j = z \), and \( t_z' \) is the value of the optimal solution on \( G' \) under the assumption that \( x_j = z \). Then, \( t_z + t_z' > t^* \). We consider two cases: \( y = z \) and \( y = \overline{z} \).

If \( t^* > T \) and \( y = z \), then \( t_z + t_z' = t^* > T = t_y + t_y' \). We consider two cases: \( y = z \) and \( y = \overline{z} \).

If \( t^* > T \) and \( y = \overline{z} \), then \( t_z + t_z' = t^* > T = t_y + t_y' \). Recall that \( t_y + t_y' \geq t_f + t_f' \). Hence, \( t_z + t_z' = t^* > T \geq t_f + t_f' = t_z + t_z' \); a contradiction.

This completes the proof of the lemma.
5 Solving QBPG on blocks

In this section we show how an instance of QBPG in the class $G(K)$ can be reduced to a number of instances on graphs from $G(K-1)$. Techniques used here are generalization of these in [4, 5]. We need an elementary fact from graph theory.

**Lemma 3** (Lemma 1 in [5]). *Let $G$ be a 2-connected graph containing two distinct longest odd cycles $C'$ and $C''$. Then, vertex sets of $C'$ and $C''$ are not disjoint.*

The following theorem relies on an observation that once a longest odd cycle is found and the variables corresponding to its

**Theorem 4.** *The QBPG problem can be solved on graphs from $G(K)$ in polynomial time.*

**Proof.** We proceed by induction on $K$. Lemma 1 shows that the QBPG problem can be solved in polynomial time on graphs from $G(0)$. Assume that $K \geq 1$ and QBPG can be solved in polynomial time on graphs from $G(K-1)$.

Let $G$ belong to $G(K) \setminus G(K-1)$. Its longest odd cycle $C$ can be found in time $O(n^{2K+1})$. A consequence of Lemma 3 is that $G[V - C] \in G(K-1)$. Now we consider all possible 0-1 assignments of values to the variables associated with the vertices of $C$, and for each assignment we create one instance. There are $O(2^{2K+1})$ such instances and each of them belongs to $G(K-1)$. By the inductive assumption, the problem can be solved in polynomial time.

**References**


