3
Related Work

There are several approaches to solve UBQP, we will now briefly discuss some of them:

Since some of them are actually algorithms for the Max Cut problem (MC), it is necessary to show their equivalence.

UBQP can model MC, as it was already shown in theorem 3, so here we show that MC can model UBQP as well.

Theorem 5 (Max Cut can model UBQP) *It is possible to model UBQP using MC.*

Proof. Let Q and b be as in definition 1, and let:

$$W = \begin{pmatrix} 0 & (-\frac{1}{2}Qe + b)^T \\ -\frac{1}{2}Qe + b & \frac{1}{2}Q \end{pmatrix}$$

(where e is the vector of all ones) be the adjacency matrix of a graph G with node set $V = \{0, 1, \ldots, n\}$, i.e., let the cost c_{ij} of the edge (i, j) be w_{ij}.

Since any cut V has the same value as its complement, we will assume without loss of generality that $0 \notin V$, whenever we define a cut of G.

Let V be a cut of G, and define $x(V) = (x_1(V), x_2(V), \ldots, x_n(V))$ where $x_i(V) = 1$ if $i \in V$ and $x_i(V) = 0$ if $i \notin V$.

It is easy to see that $x(.)$ is a bijective function from the cuts of G to $\{0, 1\}^n$. We will now show that if the cut V' has value K then $f(x(V')) = -K$. Let V be a cut of G. The value of the cut is:

$$
\begin{aligned}
K &= \sum_{i \in V} \sum_{j \notin V} W_{ij} \\
&= \sum_{i \in V} \left(W_{i0} + \sum_{j \notin V \cup \{0\}} W_{ij} \right) \\
&= \sum_{i \in V} \left(-\frac{1}{2} \sum_{j \in \{1,\ldots,n\}} Q_{ij} + b_i + \sum_{j \notin V \cup \{0\}} \frac{1}{2} Q_{ij} \right) \\
&= \sum_{i \in V} \left(-\frac{1}{2} \sum_{j \in V} Q_{ij} + b_i \right) \\
&= -\left(\frac{1}{2} \sum_{i \in V} \sum_{j \in V} Q_{ij} - \sum_{i \in V} b_i \right) \\
&= -\left(\frac{1}{2} x(V)^T Q x(V) - b^T x(V) \right)
\end{aligned}
$$

3.1 SDP relaxation

This relaxation was given in a classic paper by Goemans and Williamson [goemans1995].

Let $G = (V, E)$ with cost matrix $C = \{c_{ij}\}$ (if $(i, j) \notin E$ then $c_{ij} = 0$). Let $L = Diag(Ce) - C$, where $Diag(v)$ is the diagonal matrix with v in its diagonal. L is called the *Laplacian matrix* of G.

A simple exercise shows that the max cut can be formulated as $\max\{x^T Lx | x \in \{-1, 1\}^n\}$ (note that here we are defining the cut by means of a vector in $\{-1, 1\}^n$ and not $\{0, 1\}^n$), which can be reformulated as:

$$\begin{aligned} z = \max \quad & trLX \\ s.t.: \quad & diag(X) = e && (3.1) \\ & rank(X) = 1 && (3.2) \\ & X \text{ is positive semidefinite} && (3.3) \end{aligned}$$

Here X is thought as being $X = xx^T$ for some $x \in \{-1, 1\}^n$ which would define the cut.

Constraints (3.2) and (3.3) acts together to ensure $X = xx^T$ while (3.1) says that $x \in \{-1, 1\}^n$.

This kind of formulation, except for the rank constraint (3.2), is a SDP (Semidefinite Programming) formulation, which can be solved polynomially [lin1995, borchers1999, fujisawa2003].

By solving this relaxation (namely, by dropping constraint (3.2)), we get an upper bound whose value is not worse than $1.1382z$ [goemans1995].

See also [goemans1995] for random procedure to generate a solution with expected value at least $0.87856z$, which gives a lower bound.

These bounds can be combined in a branch and bound scheme.

3.2 Branch and Bound using SDP and the Bundle Method

This method was described at [rendl2010], and to our knowledge is the best available today.

The **triangular inequalities** for the Max Cut are as follows (here X is the matrix as in 3.1):

$$\forall i<j<k \begin{cases} X_{ij}+X_{ik}+X_{jk} & \geq -1 \\ X_{ij}-X_{ik}-X_{jk} & \geq -1 \\ -X_{ij}+X_{ik}-X_{jk} & \geq -1 \\ -X_{ij}-X_{ik}+X_{jk} & \geq -1 \end{cases} \tag{3.4}$$

which says that "for any triangle, zero or two edges are in the cut". These constraints can be viewed as $\mathcal{A}(X) \leq b$, where $\mathcal{A}$ is a linear operator from $\mathbb{R}^{n\times n}$ to $\mathbb{R}^{4\binom{n}{3}}$.

If we use the formulation in 3.1 with (3.4), we get a new problem whose Lagrangian is $\mathcal{L}(X,\gamma) = trLX + \gamma^T(b-\mathcal{A}(X))$

Then the dual function is $f(\gamma) = \max_X \mathcal{L}(X,\gamma)$, and the dual problem is $min_{\gamma\geq 0} f(\gamma)$.

For a fixed γ the evaluation of $f(\gamma)$ leads to a problem which can be easily solved by 3.1, so it uses the bundle method to solve the dual problem.

Since the number of triangles can be very big $(4\binom{n}{3})$, the dual is solved with only a portion of these triangles, and then it uses the solution (X^*, λ^*) to drop some of these constraints and add new ones.

The solution to the dual problem gives an upper bound for the Max Cut which is used in a branch and bound algorithm. The branching is done on the edges of the graph.

There is also an online solver using this method called Biq Mac (Binary Quadratic - Maximum Cut). It is available at http://biqmac.uni-klu.ac.at/. We compare the results from this solver to the results found by the new

method.

They have also collected a big number of instances for both UBQP and Max Cut called the Biq Mac Library (available at http://biqmac.uni-klu.ac.at/biqmaclib.html).

3.3 Relaxing equivalent problems to find a lower bound

This method is described at [billionnet2007], and uses a rather trivial (but really clever) observation: Since $x_i \in \{0,1\}$, then $x_i^2 = x_i$.

With this in mind, consider the following problem:

$$\begin{aligned} \min \quad & f_\lambda(x) = \frac{1}{2}x^T Q x - b^T x + \sum_{i=1}^{n} \lambda_i (x_i^2 - x_i) \\ \text{s.t.:} \quad & x \in \{0,1\}^n \end{aligned} \tag{3.5}$$

It is easy to see that $f(x) = f_\lambda(x)$ for all λ and for all feasible x, so the original problem can be replaced by the above problem for any λ.

Also, $f_\lambda(x) = \frac{1}{2}x^T(Q + 2Diag(\lambda))x - (b+\lambda)^T x$, where $Diag(\lambda)$ is the diagonal matrix with λ on its main diagonal. Let $Q' = Q + 2Diag(\lambda)$ and $b' = b + \lambda$. So the problem of optimizing $f_\lambda(x)$ is again UBQP.

The point of doing that is that even if the solution is the same, the solution to the relaxed problem (dropping the integrality constraint (3.5)) may vary, and for any λ it will be a lower bound to the original value.

If λ is such that Q' is positive definite, the relaxed problem is solvable directly by $x_\lambda = Q'^{-1}b'$. So the objective value of the relaxed problem will be:

$$\begin{aligned} x_\lambda^* &= Q'^{-1}b' \\ obj_\lambda &= f_\lambda(x_\lambda^*) \\ &= \frac{1}{2}x_\lambda^{*T} Q' x_\lambda^* - b'^T x_\lambda^* \\ &= \frac{1}{2}b'^T Q'^{-1} Q' Q'^{-1} b' - b'^T Q'^{-1} b' \\ &= -\frac{1}{2}b'^T Q'^{-1} b' \end{aligned}$$

So, we can find a family of lower bounds obj_λ.

One very natural question that arrives is: is it possible to get the best (maximum) lower bound? It actually is, by solving the following semidefinite program:

$$\begin{aligned} \min \quad & -K \\ \text{s.t.:} \quad & A = \begin{pmatrix} -2K & (b+\lambda)^T \\ b+\lambda & Q + 2Diag(\lambda) \end{pmatrix} \text{ is positive semidefinite.} \end{aligned}$$

Where K is the maximum lower bound.

It follows from the positive semidefiniteness of matrix A that:

- $K \leq 0$. This is always true in the original problem because the objective value for $x = 0$ is zero.
- $Q' = Q + 2Diag(\lambda)$ is positive semidefinite. Well, no surprise here neither.
- $K \leq -\frac{1}{2} b' Q'^{-1} b'$, by the Schur Complement of A (see, for instance, [boyd2009]).

From the fact that we are maximizing K (minimizing $-K$) and the last observation, we will have that ($K = -\frac{1}{2} b' Q'^{-1} b'$).

So, what is done in this work is: first the maximum lower bound is found (and the corresponding λ). Then they find $\min(f_\lambda(x) | x \in \{0,1\}^n)$ using a general UBQP solver. This is done due to the observation that the performance of existing UBQP solvers strongly depends on the lower bound achieved by the relaxation of the root problem.