Solving QUBO by SAT #1

hysok2

2nd June 2020

Abstract

We present a technique that solves a quadratic unconstrained binary optimization problem (QUBO) by querying a SAT oracle in polynomial time. Specifically, we prove QUBO where its coefficients are integers is in $\mathsf{FP}^{\mathsf{NP}}$. This result implies that a problem that is higher than P^{NP} in polynomial hierarchy (e.g,, a Σ_2^{P}-compete problem) can hardly be reduced into QUBO.

1 Problem Definitions

In this memo, we present the technique that solves QUBO defined as follows.

Definition 1.1 (QUBO)**.** Let $x_1, x_2, ..., x_n$ be binary variables, and Q be a $n \times n$ matrix of integers. A quadratic unconstrained binary optimization problem (QUBO) is a problem of minimizing the following quadratic polynomial expression:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} Q_{ij}x_i x_j$$

It is worth noting that the original QUBO is a problem of minimizing quadratic polynomial expression where its coefficients are real numbers, while the problem above is the problem where its coefficients are integers. That is, the technique presented in this memo can only solve the subset of the original QUBO. However, we believe that the technique is useful in practical perspective. This is because, in many cases, QUBO are used when reducing combinatorial optimization problems, hence the obtained QUBO will only have integer coefficients.

We formulate the decision problem which is used in solving QUBO. And, we prove the complexity of solving the decision problem.

Definition 1.2 (DLEQUBO)**.** Let $x_1, x_2, ..., x_n$ be binary variables, Q be a $n \times n$ matrix of integers, and q be a integer number. DLEQUBO is the problem of deciding if the following inequality holds:

$$q \leq \min_{x_1,x_2,...,x_n} \sum_{i=1}^{n}\sum_{j=1}^{n} Q_{ij}x_i x_j$$

Lemma 1.3. DLEQUBO is in coNP.

Proof. We show the complement of DLEQUBO is in NP. We have

$$\begin{aligned}
&\neg \left(q \leq \min_{x_1,x_2,\ldots,x_n} \sum_{i=1}^{n}\sum_{j=1}^{n} Q_{ij}x_i x_j \right) \\
&\Leftrightarrow \neg \left(\bigwedge_{x_1,x_2,\ldots,x_n} q \leq \sum_{i=1}^{n}\sum_{j=1}^{n} Q_{ij}x_i x_j \right) \\
&\Leftrightarrow \bigvee_{x_1,x_2,\ldots,x_n} q > \sum_{i=1}^{n}\sum_{j=1}^{n} Q_{ij}x_i x_j
\end{aligned}$$

Trivially, the disjunction of the inequalities above is solvable by a non-deterministic Turing machine in polynomial time. Therefore, the complement of DLEQUBO is in NP. It follows that DLEQUBO is in coNP. □

2 Algorithm solving QUBO by SAT

We present Algorithm 1 which solves QUBO by querying an oracle that solves DLEQUBO.

Algorithm 1 Algorithm solving QUBO with DLEQUBO

```
1:  procedure SQUBO(Q)
2:      lb ← ∑{Q_ij | Q_ij ≤ 0}
3:      ub ← 1
4:      while lb + 1 < ub do
5:          q ← lb + ⌊(ub − lb)/2⌋
6:          if (Q, q) ∈ DLEQUBO then
7:              lb ← q
8:          else
9:              ub ← q
10:     return lb                          ▷ lb is the solution of QUBO
```

The algorithm uses a binary search to find the solution of a given QUBO in a set $[lb, ub]$ where lb is in the line 2, and ub is in the line 3 of Algorithm 1. Hence, the algorithm runs in logarithmic time in the worst case by querying the oracle $O(\log(ub - lb))$ times. Note that the solution of a given QUBO is in $[lb, ub]$. We have the solution is greater than or equal to lb, because lb is a sum of non-positive coefficients of the quadratic polynomial expression. Also, we have the solution is less than ub, because for any given QUBO the quadratic polynomial expression will be 0 when $x_1 = 0, x_2 = 0, ..., x_n = 0$.

Theorem 2.1. The output of SQUBO(Q) is the solution of QUBO.

Proof. Let m be the solution of a given QUBO. In the algorithm 1, the property $lb \leq m < ub$ is the loop invariant. Trivially, after the line 3, the invariant holds. At line 6, if we have $(Q, q) \in$ DLEQUBO, new lb satisfies $lb = q \leq m$. If we have $(Q, q) \notin$ DLEQUBO, new ub satisfies $m < q = ub$.

When the algorithm terminates, the while loop condition is false, hence we have $ub \leq lb + 1$. Therefore, it follows that $ub = lb + 1$ and $\mathrm{SQUBO}(Q) = lb = m$. □

Theorem 2.2. QUBO is in $\mathsf{FP}^{\mathsf{NP}}$.

Proof. Algorithm 1 terminates in polynomial time in the size of quadratic polynomial expression of a given QUBO. This is because ub and lb are bounded by 2^M where M is the size of the quadratic polynomial expression, and ub and lb are declared at line 3 and line 2 in Algorithm 1. Therefore, QUBO is in $\mathsf{FP}^{\mathrm{DLEQUBO}}$. By Lemma 1.3 and the fact $\mathsf{FP}^{\mathsf{coNP}} = \mathsf{FP}^{\mathsf{NP}}$, we have QUBO is in $\mathsf{FP}^{\mathsf{NP}}$. □