Lab12-Approximation Algorithm II

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2018.

∗ If there is any problem, please contact TA Jiahao Fan.

∗ Name:Juncheng Wan   Student ID:516021910620   Email: 578177149@qq.com

1. Let us consider the special case of the Maximum Cut problem in which the required partition of the node set must have the same cardinality. Define a polynomial-time local search algorithm for this problem and evaluate its performance ratio.

   Solution. I provide with my idea of local search in this way: first divide the node set into two note sets A, B having the same cardinality randomly. Each time I test that if exchanging the one node in A and one node in B increases the cut. If increasing the cut, do this exchange otherwise do nothing. I do this until there is no increasing. Because the upper bound of the maximum cut is the number of edges $|E|$, this algorithm will at most loop $|E|$ times (each time the cut increases 1). Therefore, this is a polynomial-time approximation algorithm.

   First I give the pseudo code as follows and then evaluate its performance ratio.

   Algorithm 1: Maximum Cut-equally divided

   Input: Graph $G = (V, E)$ where $|V|$ is even;
   Output: Local optimal cut $(V1, V2)$ where $|V1| = |V2|$;

   ```
   Randomly and equally divide V into two sets
     V1 = {v_11, v_12, ··· v_1n}, V2 = {v_21, v_22, ··· v_2n} ;
   Increase ← True ;
   while Increase do
       Increase ← False ;
       for i = 1 to n do
           for j = 1 to n do
               if exchange(v_1i, v_2j) increases cut then
                   V1 ← V1 − {v_1i} + {v_2j} ;
                   V2 ← V2 + {v_1i} − {v_2j} ;
                   Increase ← True ;
   ;
   return (V1, V2);
   ```

   Set m_A the measure of this algorithm, m^* the optimal measure. I have:

   $$r = \frac{m^*}{m_A} \leq 2 \tag{1}$$

   Because at the last of the *while* loop *Increase* is false, exchange(v_{1i}, v_{2j}) can not increase cut. The situation is totally the same as the proof of the slide in the **Local Cut** and we can get the inequation above. □

2. **Minimum Weighted Vertex Cover:** Consider the weighted version of the Minimum Vertex Cover problem in which a non-negative weight c_i is associated with each vertex v_i and we look for a vertex cover having minimum total weight.

   (a) Given a weighted graph $G = (V, E)$ with a non-negative weight c_i associated with each vertex v_i, please formulate the Minimum Weighted Vertex Cover problem as an integer linear programming.

Solution. Assume that there are n vertices and m edges. Set $E = \{e_1 = (v_{11}, v_{12}), e_2 = (v_{21}, v_{22}), \cdots, e_m = (v_{m1}, v_{m2})\}$, where $v_{ij} \in V$.

$$\begin{aligned}
\textbf{min} \quad & z = \sum_{v_i \in V} c_i x_i \\
\textbf{s.t} \quad & v_{i1} + v_{i2} \geq 1, \quad \forall i = 1, 2, \cdots, m \\
& x_i \in \{0, 1\}, \quad \forall i = 1, 2, \cdots, n
\end{aligned}$$

□

(b) Prove that the following algorithm finds a feasible solution of the Minimum Weighted Vertex Cover problem with value $m_{LP}(G)$ such that $m_{LP}(G)/m^*(G) \leq 2$.

Algorithm 2: Rounding Weighted Vertex Cover

```
Input: Graph G = (V, E) with non-negative vertex weights;
Output: Vertex cover V' of G;
Let ILP_VC be the integer linear programming formulation of the
  problem;
Let LP_VC be the problem obtained from ILP_VC by LP-relaxation;
Let x*(G) be the optimal solution for LP_VC;
V' ← {v_i | x*_i(G) ≥ 0.5};
return V';
```

Proof. Feasible solution: For each $e_i \in E$, becasue $v_{i1} + v_{i2} \geq 1$, at least one of $\{v_{i1}, v_{i2}\}$ is greater than 0.5. Therefore, V' can cover each e_i.
Approximation Ratio: Set m_A the measure of the algorithm and m^* the optimal measure.

$$\begin{aligned}
m_A &= \sum_{v_i \in V'} c_i \\
&\leq 2 \sum_{v_i \in V'} c_i x^*_{LP}(G) \\
&\leq 2 \sum_{v_i \in V'} c_i x^*_{ILP}(G) \\
&\leq 2 \sum_{v_i \in V} c_i x^*_{ILP}(G) \\
&= 2 \sum_{v_i \in V} c_i x_i \\
&= 2m^*
\end{aligned}$$

Therefore, the approximation ratio is 2.

□

Remark: You need to include your .pdf and .tex files in your uploaded .rar or .zip file.