

# CS 138, Homework 2

Euiwoong Lee

April 23, 2009

## 1 3.6

Start with  $G_0 = G$ . For each  $i$ , find a minimum cost cycle cover  $C_i = \{c_1, \dots, c_q\}$  in  $G_i$ , using the polynomial algorithm mentioned in the problem. Let  $G_{i+1}$  be the complete graph with the vertex set  $C_i$  and  $cost(c_i, c_j) = \min_{v_i \in c_i, v_j \in c_j} cost(v_i, v_j)$ . Recurse until  $C_i$  consists of one cycle. Let  $k$  be the last step.

Starting from  $i = k$ , we find a TSP tour  $T_i$  of  $G_i$ .  $T_k = C_k$ . For  $i = k - 1, \dots, 0$ , we are given  $T_{i+1} = (c_1, \dots, c_q)$  and  $C_i = \{c_1, \dots, c_q | c_j = (v_{(j,1)}, \dots, v_{(j,p_i)})\}$ . Note that  $c_j \in V(G_{i+1})$  and  $v_{(j,l)} \in V(G_i)$ , which means that  $c_j$  is a cycle consisting of some  $v$ 's. For  $1 \leq j \leq q$ , let  $in_j$  and  $out_j$  be integers such that  $(v_{(j,out_j)}, v_{(j+1,in_{j+1})})$  is the edge used by  $T_{i+1}$  to go from  $c_j$  to  $c_{j+1}$  (suppose  $q + 1 = 1$ ). To obtain a tour of  $G_i$ , follow edges in  $T_{i+1}$ . When we arrive each vertex of  $G_{i+1}$ , which is a cycle of  $G_i$ , we visit every vertex in the cycle and follow the next edge in  $T_{i+1}$ . Since each edge used by  $T_{i+1}$  is the cheapest edge between two cycles, for each tour of cycle, we have to end up with the certain vertex to use the cheapest edge. Fortunately, we have to go around the cycle at most twice to satisfy these conditions.

Let  $c'_j = (v_{(j,in_j)}, v_{(j,in_j+1)}, \dots, v_{(j,p_j)}, v_{(j,1)}, \dots, v_{(j,out_j)})$  if  $in_j \leq out_j$ . Otherwise, let  $c'_j = (v_{(j,in_j)}, v_{(j,in_j+1)}, \dots, v_{(j,p_j)}, v_{(j,1)}, \dots, v_{(j,p_j)}, v_{(j,1)}, \dots, v_{(j,out_j)})$ .  $c'_j$  is a path starting from  $v_{(j,in_j)}$  to  $v_{(j,out_j)}$ , visiting each  $v_{(j,l)}$  at least once.  $c'_j$  uses an edge of  $c_j$  at most twice, so  $cost(c'_j) \leq 2cost(c_j)$ .

Finally, let  $T_i = c'_1 + \dots + c'_q$ , where  $+$  denotes the concatenation of paths.  $T_i$  is a tour visiting all  $v_i$ 's, and  $cost(T_i) \leq cost(T_{i+1}) + 2cost(C_i)$ , since  $cost(c'_j) \leq 2cost(c_j)$  and  $\sum_{j=1}^q cost(v_{(j,out_j)}, v_{(j+1,in_{j+1})}) = cost(T_{i+1})$ . Take shortcuts to make  $T_i$  a TSP tour. It does not increase  $cost(T_i)$ . After computing all  $T_k, \dots, T_0$ , return  $T_0$ .

$T_0$  is a TSP tour of  $G_0 = G$ , and

$$\begin{aligned} & cost(T_0) \\ & \leq 2cost(C_0) + cost(T_1) \\ & \leq 2cost(C_0) + 2cost(C_1) + cost(T_2) \\ & \dots \\ & \leq 2cost(C_0) + \dots + 2cost(C_{k-1}) + cost(C_k) \\ & \leq 2 \sum_{i=0}^k cost(C_i) \end{aligned}$$

Note that  $cost(C_i) \leq cost(\text{optimal TSP tour in } G_i) \leq cost(\text{optimal TSP tour in } G_{i-1}) \leq \dots \leq OPT$ , because every TSP tour in  $G_{i-1}$  implies a TSP tour in  $G_i$  of equal or lower cost ( $G_i$  has equal or lower edges costs, and we can take shortcuts). Furthermore, since each cycle contains  $\geq 2$  vertices,  $G_{i+1}$  has  $\leq \frac{1}{2}$  as many vertices as  $G_i$ , so  $k \leq \lceil \log_2 n \rceil$ . Therefore,  $cost(T_0) \leq 2k \cdot OPT \leq 2 \lceil \log_2 n \rceil \cdot OPT$ .

## 2 5.2

I found an example where the solution returned by this modified algorithm is 4 times the optimal solution. It is tight when the following conjecture is true: the cardinality of any minimal dominating set of  $H^4$  is at most the cardinality of any minimum dominating set of  $H$ .

If you are interested in looking at the counterexample, or proved the conjecture, please email me.

## 3 5.3

Given a simple, undirected graph  $G$  and a integer  $k$ , clique cover problem is to decide whether there exists a partition of  $V$  into  $k$  disjoint sets such that vertices in each set form a clique. This is known to be NP-Complete.

### 3.1

As in the textbook, let  $G_i = (V, E_i)$ , where  $E_i = \{e_1, \dots, e_i\}$  and  $cost(e_1) \leq \dots \leq cost(e_{|E|})$ . Our algorithm is the following.

1. Compute a maximal independent set,  $M_i$ , in each  $G_i$ .
2. Find the minimum index  $i$  such that  $|M_i| \leq k$ , say  $j$ .
3. For every  $u \in M_j$ , let  $V_u$  be the set of vertices which are  $u$  or adjacent to  $u$ . The union of  $V_u$ 's is  $V$  (will prove later). Remove some vertices from some sets to ensure that  $V_u$ 's are pairwise disjoint.
4. Return  $\{V_u | u \in M_j\}$ .

The proof of the algorithm is almost identical to Theorem 5.5 in the textbook. Use the technique of parametric pruning to see that the metric  $k$ -cluster problem is equivalent to finding the smallest index  $i$  such that  $G_i$  has a clique cover of size at most  $k$ .

Lemma 5.2 is replaced with the fact that the cardinality of any maximal independent set of  $H$  is at most the minimum number of cliques needed to cover  $H$ . Lemma 5.4 holds without any modification. For any vertex  $v$  in  $G_j$ , either  $[v \in M_j]$  or  $[\exists u \in M_j \text{ adjacent to } v]$ , since  $M_j$  is a maximal independent set of  $G_j$ . For each  $u \in M_j$ , the distance from  $u$  to any vertex in  $V_u$  is at most  $cost(e_j)$ , so the distance between any two vertices in  $V_u$  is at most  $2cost(e_j)$ . Therefore, the algorithm achieves an approximation factor of 2 for the metric  $k$ -cluster problem.

## 3.2

Reduce the clique cover problem to the metric  $k$ -cluster problem. The reduction is identical to Theorem 5.7 in the textbook.