CS 630, Fall 2024, Homework 3

Yifei Bao

September 29, 2024

Problem 1 ***Polling***

Algorithm. This problem can be converted to the Independent Set problem. Construct a conflict graph G with each group as nodes. If two groups share at least one member, then create an edge between the corresponding two nodes. Selecting groups not sharing numbers means that selected nodes in the graph share no edges.

It is a greedy algorithm based on selecting nodes (groups) with the minimum degree in the conflict graph. This algorithm initiates an empty set S for storing selected group nodes. Then iteratively choose node v with minimum degree, add corresponding group v.g to S and remove v and all its adjacent nodes from the graph until there are no nodes left in the graph.

Algorithm 1: SelectGroups($[g_1, g_2, ..., g_n]$)

```
  /* Input:  A list of groups [g_1, g_2, ..., g_n]                    */
  /* Output:  A list of selected groups S                             */
1 G ← CreateConflictGraph([g_1, g_2, ..., g_n])/* Construct a conflict graph with each
     group as nodes.  If two groups share members, create an edge between
     corresponding nodes                                              */
2 S ← [ ] /* Empty set of group nodes                                 */
3 while G is not empty do
4 |   v ← SelectMinDegreeNode(G)/* Select a node with minimum degree  */
5 |   S.append(v.g)/* Add the corresponding group to S                */
6 |   G ← RemoveNodeAndNeighbors(G, v)/* Remove the node and its neighbors */
7 return S
```

Running time and space complexity. Let n be the number of groups. Let m be the number of members. Let k be the maximum number of groups a member belongs to. Constructing n nodes in the graph costs $O(n)$.

Constructing all edges between k nodes for each member cost $O(k^2)$ and total is $O(mk^2)$. Removing nodes and neighbors costs $O(E) \leq O(mk^2)$. Hence the total running time is $O(n + mk^2)$.

For storing nodes and edges, the space complexity is $O(n + E) \leq O(n + mk^2)$.

Correctness. The conflict graph ensures that only between group nodes sharing members can an edge be added. When a node v is selected in the graph, all adjacent nodes and the node are removed. Thus no groups left could share member with group corresponding to node v. Therefore,

no two groups in S could share members. S is a valid solution for the problem.

Approximation Ratio. Let D be the maximum degree in G. The goal is to find a lower bound on $|S|$. Each v in S has at most D neighbors, so $|V - S| \leq D \cdot |S|$. Thus $|V| = |V - S| + |S| \leq D \cdot |S| + |S| = (D + 1)|S|$. In conclusion, $|OPT| \leq |V| \leq (D + 1)|S|$. The greedy algorithm is a $(D + 1)$-approximation.

Problem 2 ***Catering***

2.1

Algorithm. This problem can be converted to the Set Cover problem. For each pair of entree and drink, we find guests(set) that can be satisfied(covered) by the pair. Then we select the pair that can cover the most guests, and remove the covered guests from the set of uncovered guests. Repeat this process until all guests are covered and return the selected pairs.

Algorithm 2: PickPairs(P, G)

```
   /* P is a list of pairs (e, d)                                          */
   /* G is a list of guests, each g has list of entrees g.E and drinks g.D */
 1 X ← G/* Uncoverd guests in G                                            */
 2 S ← [ ]/* Empty set of selected pairs                                   */
 3 while X is not empty do
 4     p_best ← ( )/* Empty tuple for best pair                            */
 5     c_max ← 0/* Maximum count of guests satisfied                       */
 6     for (e, d) in P do
 7         G_satisfied ← {g ∈ U | e ∈ g.E and d ∈ g.D}/* All guests that can be satisfied
               by pair (e, d)                                             */
 8         if |G_satisfied| > c_max then
 9             p_best ← (e, d)/* Select p that covers the most guests      */
10             c_max ← |G_satisfied|/* Update maximum count               */
11     S.append(p_best);
12     X ← X − G_satisfied;
13 return S
```

Space complexity. Let m be the number of all unique entrees and n be the number of all unique drinks. Let k be the number of guests. For storing guests, the space complexity is $O(k(m + n))$ in the worst case. For storing pairs, the space complexity is $O(mn)$ in the worst case. In conclusion, the space complexity is $O(km + kn + mn)$.

Correctness. In this greedy algorithm, we select the pair (e,d) that can cover the most guests in each iteration. In each iteration, the chosen pair (e,d) is valid since it comes from P. Each time the guests covered by (e,d) are removed from the set of uncovered guests till all guests are covered. Thus, all guests are satisfied by the selected pairs from given list. The algorithm yields a valid solution.

Approximation Ratio. The algorithm is a SC greedy approximation which is correlated with the algorithm in class. Suppose the optimal solution to SC uses k pairs, and there are total n guests. In this problem, pairs correspond to sets, and guests correspond to items. Since the optimal solution uses k sets, there is at least one set in the optimal solution that covers $\frac{1}{k}$ fraction of all items. The GreedySC algorithm selects the largest set, so it also covers at least $\frac{n}{k}$ items. After the first iteration, at most $n\left(1-\frac{1}{k}\right)$ items remain uncovered. After $k\ln n$ rounds, there are at most $n\left(1-\frac{1}{k}\right)^{k\ln n}$ uncovered items left. We know that:

$$n\left(1-\frac{1}{k}\right)^{k\ln n} < n\cdot\left(\frac{1}{e}\right)^{\ln n} = 1$$

Thus, there are at most $k\ln n$ sets returned by the greedy algorithm. The greedy algorithm is a $\ln n$-approximation.

2.2

Algorithm. Unlike the first problem, where guests must be satisfied by both the entree and drink in a specific pair, in this problem guests can mix and match. For each pair, we find guests whose entree or drink preferences can be satisfied by that pair, even if they prefer to mix the entree from one pair with the drink from another. Then select the pair that satisfies the most guests, updating their entree or drink preferences as covered. Repeat the process until all guests have both their entree and drink preferences satisfied, and returns the selected pairs.

Space complexity. The space complexity is also $O(km+kn+mn)$.

Correctness. In this greedy algorithm, we select the pair (e,d) that can satisfy the most uncovered guests in each iteration. Unlike the first problem, guests can mix and match, so the algorithm tracks entree and drink preferences separately. In each iteration, the chosen pair (e,d) is valid since it comes from P. Each time, the algorithm updates the satisfaction status of guests for both entree and drink independently. Guests are removed from the set of uncovered guests only when both their entree and drink preferences are satisfied. Repeat this process until all guests have both preferences satisfied. Thus, all guests are eventually satisfied by the selected pairs from the given list. The algorithm yields a valid solution.

Approximation Ratio. This algorithm is a variant of the SC greedy approximation algorithm, adapted to allow mixing of entrees and drinks. Let the optimal solution use k pairs to satisfy all guests, and suppose there are n guests in total. In this problem, each pair corresponds to a set, and guests correspond to items, similar to the standard Set Cover problem. In each iteration, the algorithm selects the pair that satisfies the largest number of remaining guests' entree or drink preferences. Since the optimal solution uses k sets (pairs), there is at least one pair in the optimal solution that covers at least $\frac{1}{k}$ fraction of all remaining uncovered preferences (entree or drink). The greedy algorithm always selects the pair that covers the most uncovered preferences, so it covers at least as many preferences as the best pair in the optimal solution. After the first iteration, at most $n\left(1-\frac{1}{k}\right)$ preferences remain uncovered. After $k\ln n$ rounds, at most $n\left(1-\frac{1}{k}\right)^{k\ln n}$ uncovered preferences remain. We know that:

$$n\left(1-\frac{1}{k}\right)^{k\ln n} < n\cdot\left(\frac{1}{e}\right)^{\ln n} = 1$$

Thus, the algorithm returns at most $k \ln n$ pairs, similar to the standard Set Cover problem. Therefore, the greedy algorithm is a $\ln n$-approximation.

Algorithm 3: PickMixablePairs(P, G)

```
   /* P is a list of pairs (e, d)                                                    */
   /* G is a list of guests, each g has list of entrees g.E and drinks g.D           */
 1 X ← G/* Uncovered guests in G                                                     */
 2 S ← [ ]/* Empty set of selected pairs                                            */
 3 E_satisfied ← {g : False for g ∈ G}/* Store if a guest's E is satisfied          */
 4 D_satisfied ← {g : False for g ∈ G}/* Store if a guest's D is satisfied          */
 5 while X is not empty do
 6     p_best ← ( )/* Empty tuple for best pair                                     */
 7     c_max ← 0/* Maximum count of guests satisfied                               */
 8     for (e, d) in P do
 9         G_e_satisfied ← {g ∈ X | e ∈ g.E and not E_satisfied[g]}/* Guests whose entree
               preference is satisfied by e but not yet covered                   */
10         G_d_satisfied ← {g ∈ X | d ∈ g.D and not D_satisfied[g]}/* Guests whose drink
               preference is satisfied by d but not yet covered                   */
11         G_satisfied ← G_e_satisfied ∪ G_d_satisfied/* Guests satisfied by either e or d */
12         if |G_satisfied| > c_max then
13             p_best ← (e, d)/* Select the pair that satisfies the most guests  */
14             c_max ← |G_satisfied|/* Update maximum count                        */
15     S.append(p_best);
16     for g in G_e_satisfied do
17         E_satisfied[g] ← True/* The guest's entree preference is now satisfied */
18     for g in G_d_satisfied do
19         D_satisfied[g] ← True/* The guest's drink preference is now satisfied  */
20     X ← {g ∈ X | not(E_satisfied[g] and D_satisfied[g])}/* Remove guests whose both
           entree and drink preferences are satisfied                             */
21 return S
```