CSC373 Fall'19
Assignment 4 Solutions

Q1 [20 Points] Activity Selection

There are m students in a class, and a set of activities U happening in the class. Each student i is involved in a subset of activities $S_i \subseteq U$. We are told that for every activity in U, there are *exactly four* students involved in that activity.

We need to select some of the students as representatives. Our constraint is that for each activity in U, *at least three of the four students* involved in that activity must be selected. However, each student i already has some workload $w_i \geqslant 0$. So subject to that constraint, we want to minimize the total workload of the students selected as representatives.

(a) [5 Points] Write this problem as an *integer* program with 0-1 variables. Briefly explain what your variables are, and how an optimal solution to your program represents an optimal solution to our problem.

(b) [15 Points] Use LP relaxation and rounding to obtain a deterministic 2-approximation algorithm. Explain why your rounded solution is a feasible solution to the integer program and why it provides 2-approximation.

Solution to Q1

Let T_u denote the set of students involved in activity u. We are given that $|T_u| = 4$ for each $u \in U$.

(a) We use variable $x_i \in \{0, 1\}$ to indicate whether student i is selected as a representative. Then, the integer program we want to solve is:

$$\begin{aligned} &\text{Minimize } \textstyle\sum_i w_i \cdot x_i \\ &\text{Subject to} \\ &\textstyle\sum_{i \in T_u} x_i \geqslant 3, \forall u \in U \\ &x_i \in \{0,1\}, \forall i \in [m] \end{aligned}$$

The objective function clearly minimizes the total workload of the students selected, and the constraints ensure that at least three students participating in each activity are selected.

(b) Algorithm:

- Relax $x_i \in \{0, 1\}$ to $x_i \in [0, 1]$ for each $i \in [m]$.
- Solve the resulting LP to obtain an optimal fractional solution x^*.
- Round to a solution $\hat{x}$ of the original IP by setting $\hat{x}_i = 1$ if $x_i^* \geqslant 1/2$ and $\hat{x}_i = 0$ otherwise.

We need to show that (a) $\hat{x}$ is a feasible solution of the original IP, and (b) $\sum_i w_i \cdot \hat{x}_i \leqslant 2 \cdot \sum_i w_i \cdot x_i^*$ (given that the LP optimal objective value is at most the IP optimal objective value, this implies the desired 2-approximation).

Part (b) is trivial because we have $\hat{x}_i \leqslant 2 \cdot x_i^*$ for each i by the way we round.
For part (a), fix $u \in U$. The key observation is that x^* is a feasible solution of the LP. Hence, $\sum_{i \in T_u} x_i^* \geqslant 3$. If the sum of four real numbers from $[0, 1]$ is at least 3, then at least three of them must be at least 1/2: this is because if only two of them are at least 1/2, then the sum of all four is strictly less than $1 + 1 + 1/2 + 1/2 = 3$. Hence, in our rounding, at least three x_i^* will be rounded up to 1 for each u. Thus, $\hat{x}$ is a feasible solution of the IP.

Notes to graders:

- Part (a): Since the explanation is quite trivial, please give full marks if someone just writes the IP without explaining why it works. But they must write what x_i (or whatever variable they use) means.
- Part (b): 5 marks for the algorithm, 5 marks to show that rounding gives feasible IP solution, 5 marks to show 2-approximation.

Q2 [20 Points] Coffee Shop Dilemma

$$\begin{tikzcd}
(m,1) \arrow[r, -] \arrow[d, -] & (m,2) \arrow[r, -] \arrow[d, -] & \cdots \arrow[r, -] & (m,n) \arrow[d, -] \\
\vdots \arrow[d, -] & \vdots \arrow[d, -] & \cdots & \vdots \arrow[d, -] \\
(2,1) \arrow[r, -] \arrow[d, -] & (2,2) \arrow[r, -] \arrow[d, -] & \cdots \arrow[r, -] & (2,n) \arrow[d, -] \\
(1,1) \arrow[r, -] & (1,2) \arrow[r, -] & \cdots \arrow[r, -] & (1,n)
\end{tikzcd}$$

Your friends want to break into the lucrative coffee shop market by opening a new chain called *The Coffee Pot*. They have a map of the street corners in a neighbourhood of Toronto (shown above), and for each (i, j), an estimate $p_{i,j} \geqslant 0$ of the profit they can make if they open a shop on corner (i, j). If they open shops on multiple corners, their profits add up. So ideally, they would like to open a shop at every corner ("the dream").

However, if they open a shop on corner (i, j), municipal regulations forbid them from opening shops on *adjacent* corners $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, and $(i, j+1)$ (whichever exist). As you can guess, they would like to select street corners where to open shops in order to maximize their profits!

(a) [5 Points] Consider the following greedy algorithm to try and select street corners. Give a precise counter-example to show that this greedy algorithm does not always find an optimal solution. Clearly state your counter-example (values of m, n, and $p_{i,j}$ for $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$), the solution found by the greedy algorithm, and a different solution with larger profit.

```
$C \leftarrow \{(i,j) : 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n\}$   # $C$ is the set of every available corner
$S \leftarrow \varnothing$   # $S$ is the current selection of corners
while $C \neq \varnothing$:
    pick $(i,j) \in C$ with the maximum value of $p_{i,j}$
    # Add $(i,j)$ to the selection and remove it (as well as all corners adjacent to it) from $C$.
```

```
        $S \leftarrow S \cup \{(i,j)\}$
        $C \leftarrow C \setminus \{(i,j),(i-1,j),(i+1,j),(i,j-1),(i,j+1)\}$
    return $S$
```

(b) [15 Points] Prove that the greedy algorithm from part (a) gives 4-approximation.

[Hint: Let S be the selection returned by the greedy algorithm and let T be an optimal solution. Show that for all $(i,j) \in T$, either $(i,j) \in S$ or there is an adjacent $(i',j') \in S$ with $p_{i',j'} \geqslant p_{i,j}$. What does this means for all $(i,j) \in S$ and their adjacent corners?]

Solution to Q2

(a) For the input below (where we've indicated the profit of each corner), the algorithm returns the corners with profits 5 and 0, for a total of 5. However, picking both corners with profits 4 would give a total profit of 8 instead.

$$\begin{tikzcd} 4 \arrow[r, dash] \arrow[d, dash] & 5 \arrow[d, dash] \\ 0 \arrow[r, dash] & 4 \end{tikzcd}$$

(b) Note that the input to the problem can be represented as an undirected graph G, and valid selections of corners are the same as independent sets in G.

Let S be any independent set returned by the algorithm, and T be any independent set with the maximum profit in G. We want to show that $p(S) \geqslant p(T)/4$.

Take any $v \in T$. If $v \in S$, let us denote $f_v = v \in S$. If $v \notin S$, then there must exist a corresponding vertex $f_v \in S$ such that $(f_v, v) \in E$ and $p(f_v) \geqslant p(v)$. To see this, note that v must be removed by the greedy algorithm at some point (otherwise it would eventually be added to the solution S). Let f_v be the vertex whose addition to S caused the removal of v. By the greedy property of the algorithm, since the algorithm chose to add f_v instead of v at that point, it must be that $p(f_v) \geqslant p(v)$. If $v \in S$, we let $f_v = v$.

Note that $p(T) = \sum_{v \in T} p(v) \leqslant \sum_{v \in T} p(f_v) = \sum_{v' \in S} p(v') \cdot |\{v \in T : f_v = v'\}| \leqslant p(S) \cdot \max_{v' \in S} |\{v \in T : f_v = v'\}|$. To complete the proof of 4-approximation, we want to show that for each $v' \in S$, there exist at most four $v \in T$ such that $f_v = v'$.

If $v' \in T$, then in fact $v = v'$ is the unique vertex with $f_v = v'$. This is because any other vertex $v \in T$ is not adjacent to v' and thus cannot have $f_v = v'$. If $v' \notin T$, then only the (at most) four neighbours of v' can have this property. Thus, we have established that $p(T) \leqslant p(S) \cdot 4$, as required.

Notes to graders:

- Part (a): This can be more or less all-or-nothing.
- Part (b): -3 marks if the proof involves a statement that holds when vertex v belongs to one of S and T but not the other — but doesn't hold for the case where v belongs to both.

Q3 [20 Points] Randomized Algorithm

Recall that a 3CNF formula $\varphi = C_1 \wedge \ldots \wedge C_m$ consists of a conjunction of m clauses, where each clause is a disjunction of *exactly* 3 literals. In the standard (unweighted) Exact Max-3-SAT problem, our goal was to maximize the number of clauses that are "satisfied" (i.e. in which at least one literal is true). Recall that the naive randomized algorithm from class provides 7/8-approximation for this and can be derandomized.

Now, consider a related problem, Exact Robust-Max-3-SAT, in which a clause is considered "satisfied" when *at least two literals* are true, and the goal is still to maximize the number of clauses that are "satisfied", but under the new definition of clause satisfaction.

(a) [5 Points] Give a randomized 1/2-approximation algorithm for Exact Robust-Max-3-SAT. Specifically, your algorithm should return a random truth assignment of variables such that the *expected* number of clauses "satisfied" is at least $m/2$. Argue correctness of your algorithm.

(b) [15 Points] Derandomize your algorithm to derive a deterministic algorithm which *always* returns a truth assignment of variables "satisfying" at least $m/2$ clauses. Write pseudocode for your derandomized algorithm, justify its correctness, and analyze its worst-case running time.

Solution to Q3

(a) Algorithm: Set each variable to TRUE with probability 1/2 and FALSE with probability 1/2, independently of the other variables.
In each clause, the probability of exactly k literals being TRUE is $\binom{3}{k} \cdot 1/2^3$. Hence, the probability of at least 2 literals being TRUE is $(\binom{3}{2} + \binom{3}{3}) \cdot 1/2^3 = 1/2$. Thus, each clause is "satisfied" with probability 1/2. By linearity of expectation, the expected number of clauses "satisfied" is $m/2$.

(b) We use the method of conditional expectation. Let $x_1, \ldots, x_n$ be the variables. For $z_1, \ldots, z_i \in \{\text{TRUE}, \text{FALSE}\}$, let $E[z_1, \ldots, z_i]$ denote the expected number of clauses satisfied conditioned on $x_1 = z_1, \ldots, x_i = z_i$ (with $x_{i+1}, \ldots, x_n$ still being random). Then, Algorithm 1 shows how to find a deterministic truth assignment using the method of conditional expectations. Algorithm 2 shows how to evaluate conditional expectations.

```
Algorithm 1: Derandomized algorithm for Exact Robust-Max-3-SAT
  Result: Truth assignment $z_1, \ldots, z_n$
  for $i = 1, \ldots, n$ do
      $z_i \leftarrow \arg\max_{z \in \{\text{TRUE,FALSE}\}} E[z_1, \ldots, z_{i-1}, z]$;
  end
```

Algorithm 2: Evaluating conditional expectations

```
Input: $z_1, \dots, z_i$
Result: $E[z_1, \dots, z_i]$
$T = 0$;
for $j = 1, \dots, m$ do
    $t \leftarrow$ number of literals in $C_j$ that evaluate to TRUE when we set $x_1 = z_1, \dots, x_i = z_i$;
    $f \leftarrow$ number of literals in $C_j$ that evaluate to FALSE when we set $x_1 = z_1, \dots, x_i = z_i$;
    if $t \geqslant 2$ then
        $T_j \leftarrow 1$;                  // Clause is already robustly satisfied
    else
        $T_j \leftarrow \sum_{k=2-t}^{3-t-f} \binom{3-t-f}{k} \cdot \frac{1}{2^{3-t-f}}$;     // The probability that, out of $3-t-f$
                                                       literals left, at least $2-t$
                                                       evaluate to TRUE
    end
    $T \leftarrow T + T_j$;
end
return $T$
```

Algorithm 1 is self-evident. In Algorithm 2, we note that by linearity of expectation, the expected number of clauses satisfied is the sum of probabilities of the different clauses being satisfied. T_j computes the probability of clause C_j being satisfied. This is trivially 1 if setting $z_1, \dots, z_i$ already makes at least two literals TRUE. Otherwise, the else condition evaluates the exact probability.

The running time of Algorithm 1 is $2n$ times the running time of Algorithm 2 because in each of n iterations, Algorithm 2 is called twice. The running time of Algorithm 2 is $O(m)$ because the time spent for each clause (which has a constant number of literals) is constant. Hence, the worst-case running time is $O(m \cdot n)$.

Notes to graders:

- Part (a): -2 marks if they write the algorithm but do not justify why it works. -1 mark if they justify by saying that each clause is satisfied with probability 1/2, but do not say why that is true.
- Part (b): -5 marks for not giving the algorithm to evaluate conditional expectations and -5 marks for missing justification or runtime analysis.